# Circuit Design Rules for Mixed Static and Dynamic CMOS Logic Circuits

by

Rolando Ramírez Ortiz, B.Sc., M. Eng.

A Ph.D. thesis submitted to the
Faculty of Graduate Studies and Research
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

Ottawa-Carleton Institute for Electrical Engineering,
Department of Electronics,
Carleton University,
Ottawa, Ontario, Canada

January, 1999

# abstract

This research is an improvement to the available techniques for logic circuit design for the mixture of *single-clock-phase* dynamic-logic gates and static gates in CMOS technology. The analysis consists of a timing framework, a generalized method for finding the waveform response by each gate circuit, and a set of principles which identify when a gate is operating correctly. A new technique is found and is simple enough to attempt a manual optimization, or to implement it into a Computer Aided Design tool. The technique covers most single-phase clock dynamic-logic techniques such as Domino, TSPC and All-N logic. A new power estimating method based on the analysis is also introduced. As an example of the new technique two benchmark Static circuits are modified into a mixture with Dynamic logic where the power consumption is reduced.

## Acknowledgements

The completion of this thesis would not have been possible without the support and encouragement from various people. These people include staff and students at Carleton University, as well as friends and family. I thank you all.

My friends, both recent and properly aged, they all encouraged me. They are all wonderful people with warm human characteristics which made this achievement possible.

In particular I'd like to mention the following: my supervisor Dr. John P. Knight; the properly aged "boys" from the DRAB cycling club - Dr. Darrell Makarenko (Jose D), Señor Alan Dean (Jose A), Señor Bob Stout (Jose B); my parents without whom this would never have been possible - Roberto R. Ramírez Gonzalez and Silvia E. Ortiz de Ramírez, and my close friends Tara L.Hennessy and Walt T. Bax.

I would also like to thank Micronet, NSERC and Dr. Martin Lefebvre for their support.

... and now let's have a cigar!


Rolando Ramírez Ortiz (Jose R), January 1999

# Contents

# List of Figures

# Chapter 1

# Introduction

This thesis is about the analysis of various methods for CMOS Dynamic Logic gates and CMOS Static gates. The objective of this analysis is to generalize the behavior and usage of these gates with the purpose of finding a set of circuit design rules that will become part of a Computer Aided Design tool.

### Synthesis and logic gates

The synthesis of logic circuits is a process which facilitates bringing the various descriptions of a circuit from an abstract logic representation to a polygon layout. Such processes, in the form of computer tools, have been successfully applied to large logic circuits.

However stringent demands towards better performance have driven engineers to consider alternative circuit techniques. The case considered here is when different structures for logic gate circuits are introduced to the synthesis process. The computer tools that circuit designers use reach a limit at this stage because they were aimed primarily to operate on a fixed set of rules for Static Logic circuit design. Therefore the automation gained by the synthesis process is partially lost and the engineer is now faced with a new problem, learning new methods and the manual optimization of logic circuits.

## Dynamic Logic

The use of different types of gate circuits, such as the Domino Logic gates[KLL82] and the True-Single-Phase-Clock(TSPC)[JS89] latches, have demonstrated that there are alternative gate circuits providing certain advantages.

Domino Logic gates and the TSPC latches belong to a family of circuits named Dynamic Logic. This family of circuits covers many methods and concepts fundamental to this type of gates and in many cases their basic concepts overlap. In fact the existing information to make a Dynamic Logic circuit is available in the form of optimized gates and circuit design techniques. Such techniques are sometimes presented as alternative solutions to each other. A consequence of these alternative solutions are circuits which have a guaranteed compatibility with a restricted group of gates.

Dynamic Logic gates have in general a complex behavior when compared to the more common Static Logic gates. This complexity is approached ingenuously by many techniques and they provide working circuits. Unfortunately the lack of compatibility between techniques and their intrinsic complexity have often prompted engineers to abandon some of these techniques.

A literature survey can reveal the existence of a constant research effort to exploit the intrinsic advantages found in Dynamic Logic gates. For example the following techniques were presented as improved versions to each other. Starting with the gates from the Domino Logic technique, the rules set by the No-Race technique [GM83] provided a link between Domino gates and $C^2MOS$ latches. These latches require two clock-phase signals and introduce many structural constraints to make a working logic circuit. These structural constraints reduced the popularity of Dynamic Logic for a number of years until the True-Single-Phase Clock latches, introduced by the TSPC technique, renewed an interest in Dynamic Logic.

### A basic technique based on common concepts

The research presented here is aimed to bring commonality between a group of Dynamic Logic techniques. Where the gate circuits in these techniques have some structural similarities. This commonality is found by means of the behavioral analysis of the basic Dynamic Logic gates. This type of analysis is required because it is the first step for understanding our current wealth of techniques and gates, and it provides the ability of including future developments. New Dynamic Logic developments are likely to appear as it continues to be commercially used. Therefore its analysis is also required to have a level of generalization appropriate for this type of circuits.

The concept of commonality leads to circuit design rules and these are the result of a better understanding of Dynamic Logic. The circuit design rules are intended to be as simple as possible to such extent that a manual effort can produce working circuits. Because the simpler is our understanding of Dynamic gates, and their behavior, the bigger is the possibility of including future developments.

This common set of circuit design rules is also a pointer towards a computer aided design tool. Some important performance issues such as noise, power and clock feedthrough are of primordial concern. However they involve principles independent to the timing considerations of this thesis and are left for future work.

### Hinted Benefits from using Dynamic Logic Gates

The number of commercial applications using Dynamic Logic is often an insufficient justification to include it in a product. A common question about Dynamic Logic is what are its benefits, and if they will justify deviating from current methods where reliable circuits are being made with Static Logic.

A first trial to answer this question is to bring more evidence. For example, when the Alpha chip [D+92] was first introduced it contained many CMOS Dynamic latches and replaced a few Static Logic gates with Dynamic gates. Other manufacturers would claim a similar performance without using Dynamic Logic, however the most recent development release indicates that the best results are obtained when Dynamic Logic gates are part of the

design strategy from the very beginning [S+98] [FB98].

Another form of justifying Dynamic Logic is gained from a structural point of view. For instance, if a circuit is made entirely of Static Logic gates then this circuit is one possible solution within the search space created by this type of circuits. If on the other hand the same circuit is made entirely of Dynamic Logic gates then this is a possible solution, from a separate search space, that exploits those conditions which are favorable to Dynamic Logic. A common set of rules will effectively merge both Static and Dynamic logic circuits and will subsequently expand the universe of possible solutions. This larger number of solutions does not guarantee a better circuit but it has a better probability of finding a better circuit. Because the best tradeoff between both types of circuits is also part of the universe of solutions.

Additional evidence is found from the performance measurement of independent Dynamic and Static logic gates. Circuit simulations of individual gates show that techniques such as Domino logic are able to perform better than their Static logic counterparts[SL94], but only under certain circumstances. This type of simulations show the potential in Dynamic Logic. However without the knowledge of how these simulated gates relate to other gate types a circuit designer is unable to recreate the right conditions. The skeptic will remain unconvinced.

The characteristics found in Dynamic Logic gates make them a desirable option to have in addition to Static Logic gates.

The following points state some cases where the propagation delay in Dynamic Logic gates can be **faster** than their Static Logic gate counterparts:

- Precharged Dynamic gates have a single logic evaluation block instead of two. The reduced load allows for a faster charge/discharge of the gate nodes.

- Dynamic latches are compact circuits combining complex logic and latch functions [D+92] [JS89]. This dual purpose reduces the number of stages for the data to propagate through.

- A fixed precharge voltage level in Dynamic gates eliminates one type of transition at the output. The propagation delay for this type of transition is effectively zero.

The following items indicate some characteristics that can be exploited in dynamic logic, under the right circumstances, to **consume less power**:

- Precharged Dynamic gates provide a lower capacitive load to its driving gates, because precharged Dynamic gates have one logic evaluation block instead of two.
- Static logic gates are subject to transient short circuit currents contributing 10% to 60% [Yea98] of the power. Dynamic logic eliminates this type of current.
- Dynamic latches can include complex logic in their structure. Fewer devices in a gate or latch leads to smaller capacitive loads and a subsequent reduction of dynamic power consumption.
- Dynamic latches are simpler. Dynamic latches, such as TSPC, require fewer devices than conventional circuits.
- Dynamic logic gates induce a reduced number of power consuming glitches [Yea98]. Because only one voltage transition is propagated during the evaluate state.

## 1.1 Objectives

Finding a common analysis framework and gaining the better understanding of Dynamic Logic gates, and its techniques, is the objective of this research.

The analysis presented here provides the study for a restricted group of Dynamic Logic gates which have a *single-phase clock* mode of operation and do not include *pass-transistor* gates. The common analysis framework is used to derive the behavioral relationships between Dynamic gates and other circuits. These relationships are the circuit design rules for a specific group of gates and they guarantee their logic functionality.

In addition two benchmark circuits are considered. These benchmark circuits are methodically transformed from a full Static Logic circuit to a new version where Dynamic Logic gates are utilized. The purpose of these

benchmarks is to demonstrate the structural flexibility gained from the better understanding of Dynamic Logic.

The analysis provides a form of behavior information and it is used here to estimate the power consumption of Dynamic circuits. This power estimator is introduced as an example to illustrate the benefits from using the information gained during the analysis and to demonstrate the flexibility of the rules.

The following is a summary of the objectives set for this research to help accomplish the analysis and obtain a new technique with circuit design rules for Dynamic Logic,

- First: Analyze dynamic logic gates (including static) and latches found in a group of Dynamic Logic techniques. This involves:
  - Reducing the number of gate types described in the for these techniques to a set of *fundamental gate types.*
  - Setting up a simplified waveform description model.
  - Defining the characteristics of a properly working gate in terms of the waveform model.
- Second: Find the Input/Output node interconnection rules for the fundamental gates. The analysis performed for the first objective identifies which gate interconnections operate correctly. Searching for the rules involves:
  - Outlining a search algorithm which yields all possible Input/Output node interconnections for gates.
  - Summarizing the results from the search algorithm into a simplified set of rules.

  These rules augmented by the accumulated wisdom from the involved dynamic logic techniques is what constitutes the new technique for dynamic logic gates. Specific gate circuits made with the fundamental types can then be chosen to be optimized for the desired properties.
- Third: Provide an application of the new concepts. The analysis performed for the first objective is able to forecast the output waveforms

of a circuit. This information is used for developing the following applications:

- A power estimation method based on waveform probabilities and their propagation in a Dynamic Logic circuit.
- Two benchmark circuits are converted from a full Static Logic implementation to a new version mixing Static and Dynamic Logic gates. Their logic correctness is guaranteed by the rules found in the second objective.

## 1.2 Organization

The organization of this thesis is as follows:

- Chapter 2, Background: An introduction to static and dynamic logic gates is presented. First are introduced the principles for Static gates, followed by Dynamic logic gates and by the techniques for larger dynamic logic circuits.
- Chapter 3, Analysis of Dynamic Logic: The fundamental gates are extracted from a group of techniques and both their behavior and interaction is analyzed. The results from this chapter are the interconnection rules for the fundamental gates.
- Chapter 4, Examples using the Circuit Technique: The examples in this chapter are aimed to illustrate how dynamic gates can be interconnected as well as to predict their output response.
- Chapter 5, Performance Measurement Model: A basic power model based on the analysis in Chapter 3 is introduced. This model is merely an example illustrating the potential from the additional information gained about the behavior of Dynamic Logic gates.
- Chapter 6, Examples using the Probability Model: Two benchmark circuits are modified to illustrate the flexibility gained from the new concepts. The performance measurement model in Chapter 5 is utilized

with the purpose of granting an objective decision between alternative circuits.

- Chapter 7, Conclusion: A summary of contributions is presented together with a description of future work.

# Chapter 2

# Background

## 2.1 Introduction

This chapter is a survey on the circuit design techniques for dynamic logic which lead to a unified technique whose gates have non-complementary inputs and operate under a single-clock phase. The techniques presented in this chapter are specific to CMOS technology and consequently two types of devices are available: the *n channel* (NMOS) and the *p channel* (PMOS). Their properties are well known in digital electronics [Sho88] [Rab96] and they are the only devices to be considered.

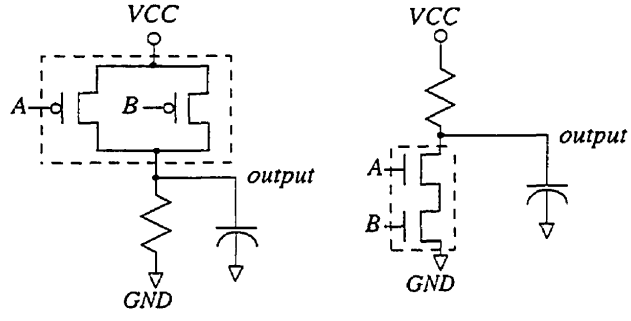
This chapter begins with the circuit design of the *logic evaluation block* and the structure of static logic gates. The techniques for dynamic logic are next and they are presented with the objective of providing a meaningful chronology of its developments towards a unified single-phase technique with static logic.

## 2.2 Logic Evaluation Block

The *logic evaluation block* is a structure of transistor devices of the same type which is part of the circuitry in a logic gate. The *logic evaluation block* is connected to a logic gate by two terminals, the *source* and the *drain* of the block, and its operation is as follows.

The input signals will charge or discharge the gate capacitances of the transistors in the evaluation block. These devices will provide a conduction path thru it, or none at all, connecting the *drain* to the *source* node. A *logic evaluation block* is seen as a switch which is conditionally open or closed.

**Example 1** *Two logic gates implement the same logic function in Figure 2.1, however their evaluation blocks contain different devices. A resistor has been added to illustrate the functionality of each block separately.*




*logic gate with P type evaluation block (PBLOCK)*

*logic gate with N type evaluation block (NBLOCK)*

*A single evaluation block is used on each circuit to implement the same logic function* $f = \overline{AB}$

*Figure 2.1: Logic Evaluation Blocks*

### 2.2.1 N-type evaluation block (Nblock)

First consider the transistor devices in a logic evaluation block that have their source and drain nodes arranged in either a *series* or a *parallel* connection.

The NBLOCK is a logic evaluation block whose transistor devices are all NMOS and are connected to provide a conductive path to GND, therefore, a path through it generates a logic zero at the output.

The *parallel* arrangement of devices in the NBLOCK generates a conductive path, from the common source and drain nodes of the devices, if at least one of their gate nodes receives a high voltage level (VH). The evaluated logic function by an NBLOCK made of two devices in parallel is the NOR function $\bar{f} = A + B$.

The logic function NAND is evaluated for a *series* arrangement of the NMOS transistors in Figure 2.1, because it requires that all the devices involved be activated to provide a conductive path to the GND node. The evaluated logic function by an NBLOCK with two devices in series is the NAND function $\bar{f} = A\,B$.

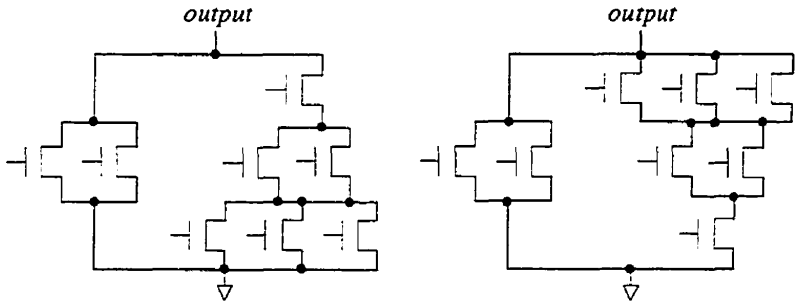
### 2.2.2 P-type evaluation block (Pblock)

The PBLOCK has an evaluation block implemented entirely with PMOS devices and is connected to provide a conductive path towards the VH supply. The potential from this supply generates a logic one, therefore a PBLOCK with two devices in *series* implements the function $f = \bar{A}\,\bar{B}$ (or the NOR function $f = \overline{A + B}$), whereas the *parallel* connection of two PMOS devices evaluates the NAND logic function $f = \bar{A} + \bar{B} = \overline{A\,B}$.

### 2.2.3 Complex Functions

Complex gates with a logic depth of more than one logic operator have some devices arranged in series and others in parallel leaving, for some instances, many alternatives for the evaluation block. The selection of the best arrangement is based on a structural sort [Man89] of the number of devices in parallel. The sorting-priority is chosen according to the type of devices in the evaluation-block.

For example, if the devices in the logic-block are NMOS then the top sorting priority of Figure 2.2 will provide the smallest capacitance load at the output. Similarly for a logic-block made of PMOS devices a bottom sorting priority will provide the least load to the output.




The arrangement of devices in these evaluation blocks maintain the same logic function. These two options allow to either minimize or maximize the output capacitance due to the device's intrinsic loads connected at the output node.

Figure 2.2: Structural sort options for the NBLOCK.

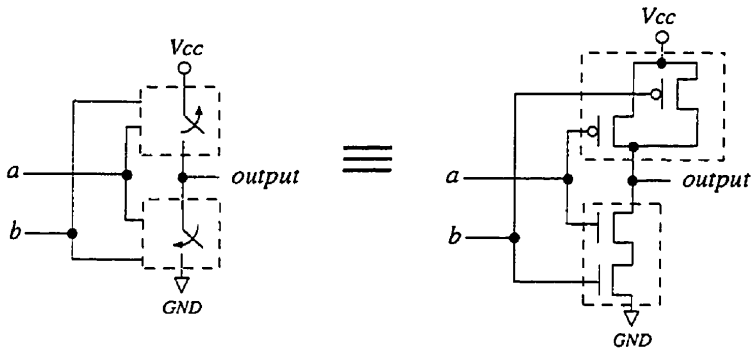
### Other arrangements

*Bridge* combinations of devices provides alternative arrangements which may simplify the structure of an evaluation. Bridge combinations should be considered whenever the structure of the logic evaluation block may require some improvements [Dag91] [ZAEB93].

## 2.3 Static Logic Gates

The first technique to consider is static logic. The output of a gate is connected to two logic evaluation blocks, one is an NBLOCK and the other one is a PBLOCK, these blocks have a *complementary* mode of operation and are directly connected to the output node and each evaluation block is connected to the GND and VCC supply nodes respectively.

**Example 2** *The logic evaluation blocks have, in general, the behavior of a switch, Figure 2.3a shows the complementary behavior of the logic blocks within a static logic gate. The example in Figure 2.3b shows a* NAND *logic gate implemented in static logic.*




*The complementary mode of operation in a static logic gate gives a path from the output to either supply, but never both at the same time.*

*Figure 2.3: NAND gate in Static Logic*

## 2.4 Dynamic Logic Gates

*Dynamic logic* is based on the ability of MOS devices to hold their charge at the gate node without a direct path to a voltage supply. This mode of operation is named dynamic and numerous techniques have been developed to create *dynamic logic* gates whose performance is comparable and sometimes better than their static logic counterparts [SL94].

### 2.4.1 N and P Type Dynamic Logic Gates

The basic structure for a dynamic logic gate consists of a single evaluation block connected by two clock devices as illustrated in Figure 2.4. The type of transistors used in the evaluation block (N channel or P channel) will determine the type of this dynamic gate.

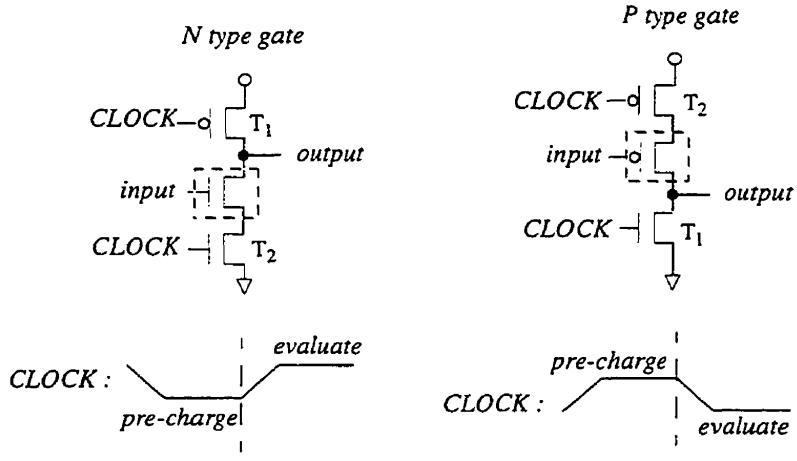
The evaluation of logic is performed by two states of operation: the *precharge* and the *evaluate* state. The *precharge* state begins when the clock signal activates device $T_1$, which is connected to the output, and loads a known voltage level, such as VH for an N-type gate. The $T_2$ clock device is of an opposite type and it is off, therefore the logic evaluation block has no effect over the output voltage.

The *evaluate* state begins when the clock signal changes to a voltage level that activates the $T_2$ clock device. During the *evaluate* state the NMOS clock device of an N-type gate will help discharge the output node if there is a conduction path thru the logic evaluation block, otherwise the output will hold its charge dynamically until the next precharge.

### 2.4.2 Domino Logic Technique

Domino logic is a technique that uses a single type of dynamic gate. This technique originated from the fact that N-type dynamic gates cannot be connected to other N-type gates directly because there is a race condition to avoid: the propagation time to the output, during the evaluate state, has to be faster than the clock signal.

For example in Figure 2.5, if an N type dynamic gate is precharging then the output from this gate is a VH level. The evaluate state uses a VH level




The sequence of events for both gates has the same precharge-evaluate order. However these gates have complementary CLOCK signals.

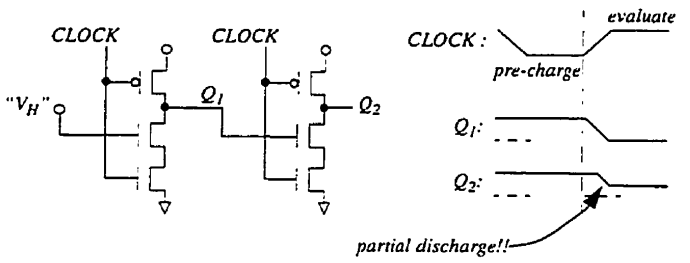
Figure 2.4: N and P type Dynamic Gates.

for the clock signal and if there's a second N-type gate that is driven, by the output of the first gate, then the output of the second gate should stay at the VH level too until the first gate has finished propagating the right output.

However both gates are controlled by the same clock signal. By the time the output of the first gate has propagated, the second gate will be discharged because it evaluated the VH precharged level from the first gate. The second gate is unable to recover its last charge and the clock signal has propagated faster than the data. As a result of this race, the clock signal has given a path to GND thru the evaluation block of the second gate before the arrival of a valid input level.

### Racefree Domino Logic

Domino logic solves this race problem between clock and data by inverting the output level with a static gate as illustrated in Figure 2.6. Domino logic operates with the same clock signal as dynamic gates; first the clock signal initiates the precharge state and loads all the output nodes. During the

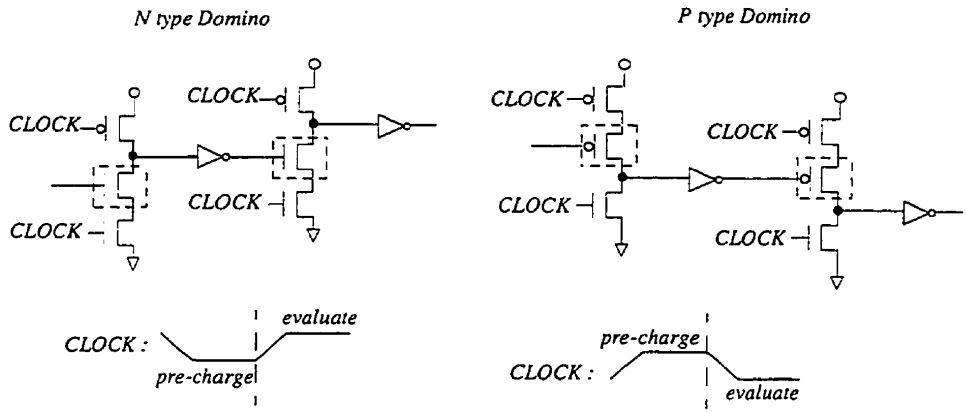



This figure illustrates the race condition between data and clock signal. The clock signal starts the evaluate state in both gates faster than the gates are able to propagate their data.

Figure 2.5: Race in a dynamic circuit made of N-type gates only

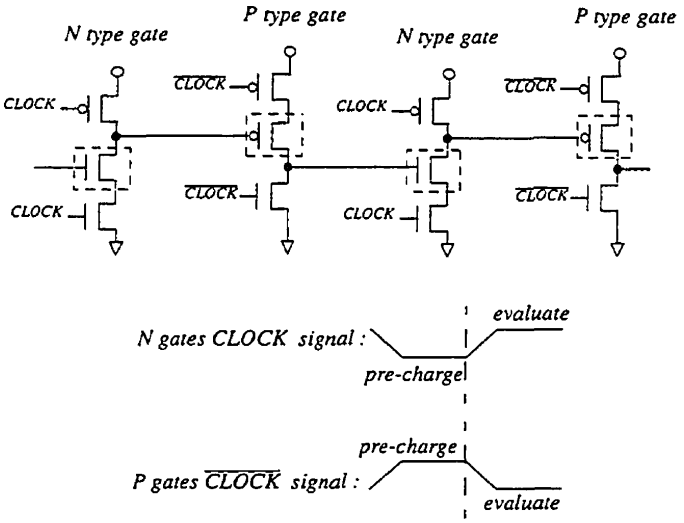
evaluate state these outputs will either stay at the precharged level or will discharge depending on the existence of a path thru the evaluation block.

In general, *a dynamic gate is able to drive another dynamic gate of the same type if there is an odd number of inversions between them.*




Odd numbers of static inverters between same types of dynamic gates create an inverted precharge state. The CLOCK signal is no longer in a race against data propagation.

Figure 2.6: Domino logic technique




Alternating N and P dynamic logic gates under complementary CLOCK signals is another technique where races are avoided. The logic functions implemented by this technique are easier than those from the DOMINO technique since no extra inverters are required.

Figure 2.7: N–P structure technique

### 2.4.3 N–P Structure Technique

The *n–p structure* technique [FL84] is an alternative solution to the race problem. This technique builds circuits with N and P type dynamic gates alternating along paths and are driven by both phases of the clock signal. Figure 2.7 illustrates the clock signals which control the precharge and evaluate states for N and P-type gates.

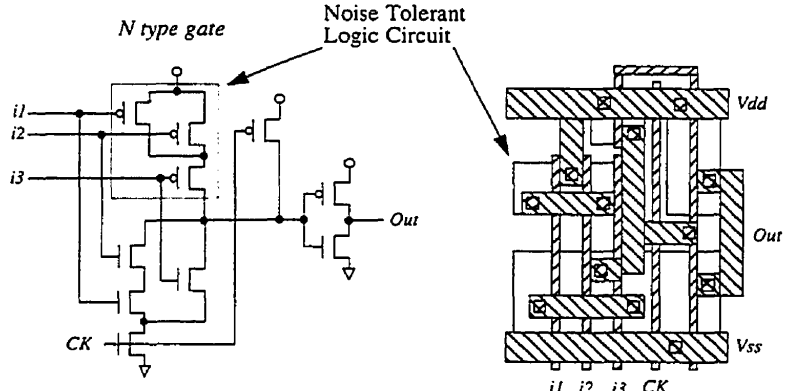
Similarly to the static inverter in the Domino logic technique, the race problem is solved because the devices, of the next gate, hold a precharged voltage that doesn't create a conduction path in the evaluation block.

### 2.4.4 Noise Tolerant Precharge (NTP) dynamic logic

The circuit in the *Noise Tolerant Precharge* logic (NTP) [M+96] provides an improved version of the N-type dynamic gates. The gates in this technique

have an additional evaluation block of minimum device size whose purpose is to pull up a partially discharged output. Subsequently this additional block helps prevent charge sharing and leakage (Section 2.4.5).

For example, the NTP gate of Figure 2.8a is a dynamic N-type gate with an additional P-type evaluation block. The evaluation block is a feed-forward structure which provides a conduction path from the output to the $V_H$ supply node. The devices for this P-type block are kept to a minimum and as shown in Figure 2.8b there is not a great impact on the total area of the gate. This effect is more beneficial for the larger gates such as bus drivers.

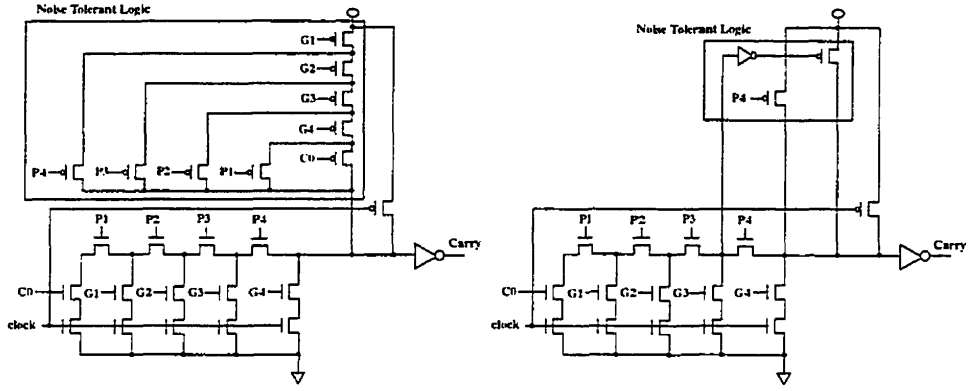



Example of an AND-OR NTP, figure extracted from [M+96].

Figure 2.8: Noise Tolerant Precharge circuit

Furthermore, because the purpose of this P-type block is to provide a current source for some outputs it can be further optimized. The Manchester carry generating gate of Figure 2.9a is optimized to a smaller version in Figure 2.9b at the expense of diminishing the noise tolerance for some input signals. Simplified structures such as the carry Manchester are possible because the behavior of the incoming signals is known and it intrinsically avoids a hazardous sequence at the input.

In summary, NTP logic circuits are useful to replace dynamic precharged gates in techniques such as Domino logic. These gates are useful for non-minimal gates and for gates where the floating node picks up noise easily,




**(a)** **(b)**

a) NTP with full N block b) NTP with simplified noise tolerant N block, figure extracted from [M+96].

Figure 2.9: 4 bit carry NTP circuit

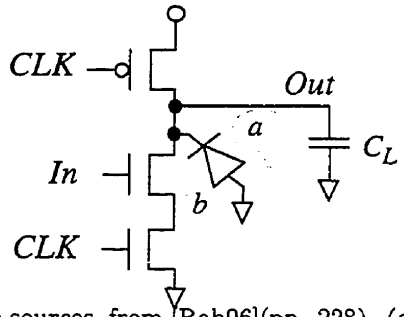
and finally the P-type block allows to have a slower slew rate in the clock signal.

### 2.4.5 Improvements for N and P Gates

N and P type Dynamic gates are susceptible to *charge sharing* and *charge leakage*. Charge sharing originates from a redistribution of charge whenever the structure of the evaluation block has one or more discharged internal nodes that become in contact with the output load. Charge leakage is due to subthreshold currents flowing from drain to source, and by leaked currents thru the reversed biased diode of the diffused area, Figure 2.10 from [Rab96](pp. 227).

These deficiencies are diminished by three different methods, and are applicable to both N and P type gates:

- The addition of extra precharge devices in Figure 2.11a precharges internal nodes. This extra device alleviates charge sharing but slows down the discharge of the output, and the circuit still suffers from




Leakage sources, from [Rab96](pp. 228). (a) from the reversed biased diode of the diffused area, and (b) from subthreshold currents flowing from drain to source.

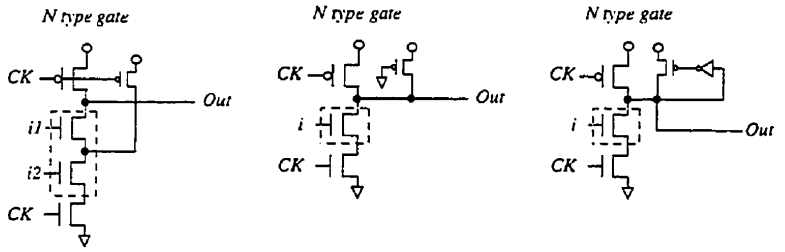
Figure 2.10: Leakage in dynamic circuits

charge leakage. The sizing for this device is small because of the smaller capacitance it is charging.

- The addition of a constant current source or a *static bleeder device.* The current source is a small device in parallel with the precharge device that provides a weak current, and creating with it a pseudo-NMOS gate. Charge sharing and leakage are diminished at the cost of some static power consumption. This device should have a long and narrow geometry [Rab96].

- A feedback device restores the output level of the dynamic gate. A small inverter in Figure 2.11c drives the feedback device while a larger static buffer, or a complex gate, drives the output load [KLL82].

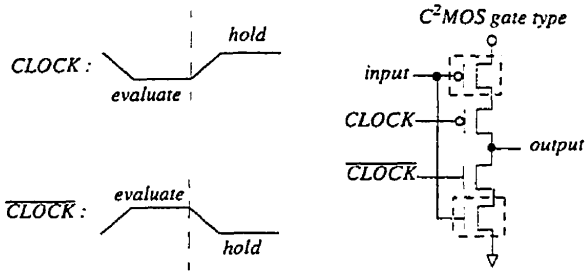
- The addition of a *noise tolerant precharge* block made of complementary devices provides a selective current source to the output [M+96] (Section 2.4.4).




(a) (b) (c)

(a) Precharge of internal nodes, (b) Constant current source, and (c) feedback device.

Figure 2.11: Simple solutions to charge sharing and leakage




The levels from both CLOCK signals are synchronized to simultaneously connect the output to the evaluation blocks. The output node is either in direct contact with the evaluation blocks or is dynamically held for half the clock cycle.

Figure 2.12: C²MOS dynamic latch

## 2.5 Dynamic Logic Latches

### 2.5.1 C²MOS Dynamic Latches

The C²MOS gate is the dynamic latch of Figure 2.12 [SOA73]. This circuit is a latch because it dynamically holds the output during the *hold* state and sets a new level during the *evaluate* state.

The *hold state* occurs when the output capacitance is isolated from both evaluation blocks. For example in the C²MOS latch of Figure 2.12, when the CLOCK signal is set to a VL level and the $\overline{\text{CLOCK}}$ into a VH level then both clock devices will have no conduction channel and the output node is isolated.

The *evaluate state* occurs when the clock devices in the C²MOS latch have conduction channels and allow the output node to be charged by either the NBLOCK or the PBLOCK. The structure of the logic evaluation blocks is complementary, and in the example of Figure 2.12 the voltage levels for the CLOCK and $\overline{\text{CLOCK}}$ signals during the *evaluate state* are VL and VH respectively.

### 2.5.2 NO–RAce (NORA) Technique

The *No–Race* technique of Figure 2.13 [GM83] is a set of rules which guarantee the correct functionality of a circuit when mixing DOMINO logic gates, Static gates, N-P structures and $C^2$MOS latches.

In the NORA technique two sections of logic gates are separated by a $C^2$MOS latch, where each section operates with an opposite assignment of the clock signal to the dynamic gates. These sections are classified with the clock phase that controls the evaluate state of an N-type dynamic gate. The name for each section are the CLOCK pipeline and the $\overline{\text{CLOCK}}$ pipeline.

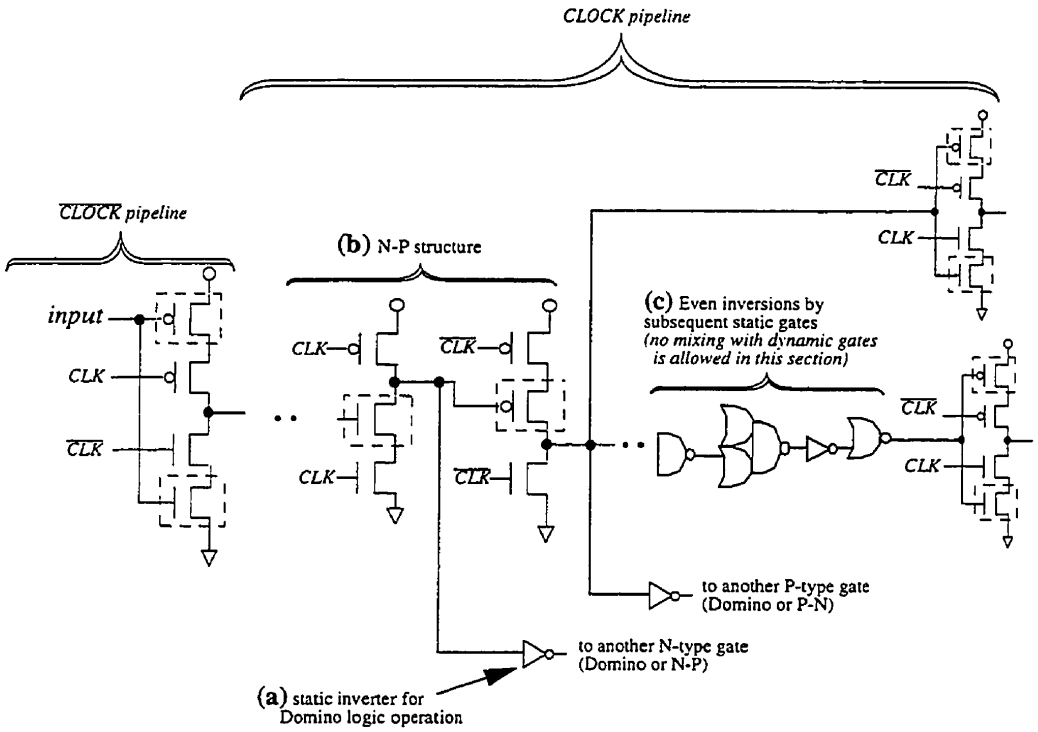
A CLOCK *pipeline* is where an N–type gate is controlled by the $CLK$ clock signal and the P–type gate uses the $\overline{CLK}$ clock signal. Similarly, the $\overline{\text{CLOCK}}$ *section* requires a $\overline{CLK}$ signal for the N–type gates and a $CLK$ signal for the P–type gates.

The assignment of different clock phases between pipelines gives them a complementary mode of operation. When the CLOCK pipeline is propagating data and its $C^2$MOS latch is transparent the $\overline{\text{CLOCK}}$ pipeline is in precharge and its $C^2$MOS latch is holding the output data.

#### NO–RAce (NORA) Technique, constraints

The circuit structure between two latches will operate without error as long as the rules set by the Domino Logic technique are met, as illustrated in Figure 2.13a, and as well as the constraints for building a circuit of the N-P-structure technique, and illustrated in Figure 2.13b. In addition, the utilization of static gates is restricted in Figure 2.13c to exist only after the dynamic gates. The number of gates between latches must be even and the number of static gates before a latch must be even too. Finally, no mixture between static and dynamic gates is allowed once static gates are utilized because a circuit with a liberal use of static gates is known to induce glitches [GM83].

The mixture of precharged N and P dynamic gates with $C^2$MOS latches is guaranteed a racefree mode of operation by the NORA technique. The even number of static gates between a dynamic gate and the output $C^2$MOS latch has the effect of propagating the same transient from node N1, in Figure 2.14,




- The number of invertions between latches is always even.

The topology constraints for the NORA technique are specified by the correct utilization of a) Domino logic gates, b) N-P-structure gates and c) and even number of static gates after the dynamic gates and before a $C^2MOS$ latch.

Figure 2.13: No–Race Technique

to node N2. A dynamic gate connected to this C$^2$MOS latch is immune to the overlap of the clock phases because "the alteration of the output information is controlled by only one of the phases $CLK$ or $\overline{CLK}$"[GM83].

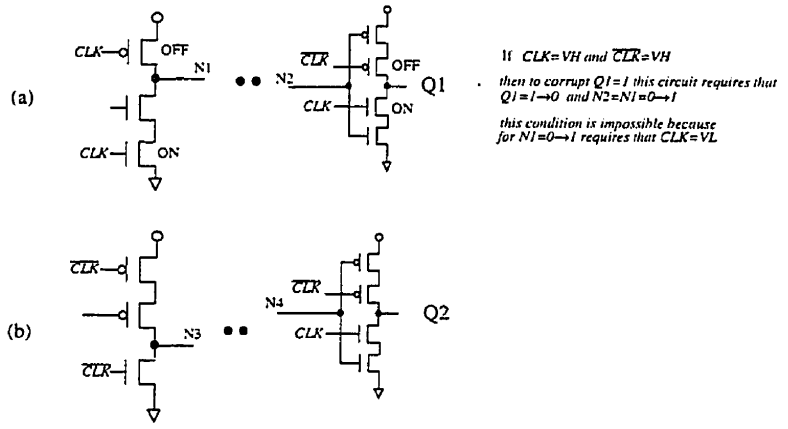



Figure 2.14: Precharge racefree in NORA

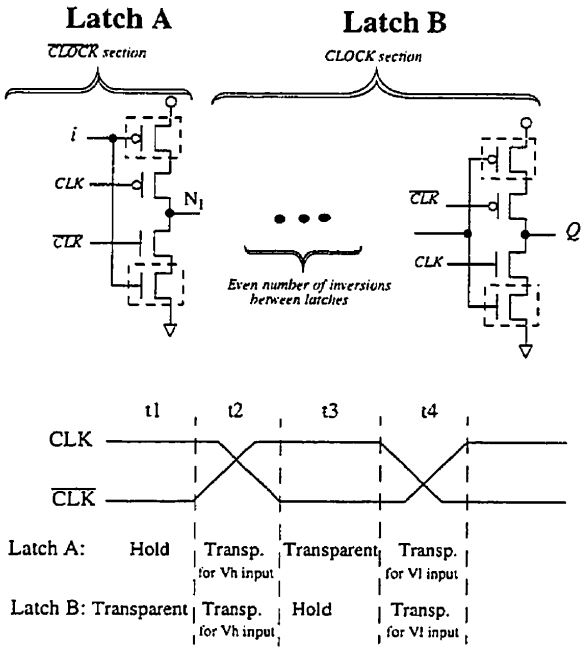
For example in Figure 2.14a, if the N-type dynamic gate is entering the precharge state, then the C$^2$MOS latch is expected to hold the evaluated data. When the clock signal $\overline{CLK}$ changes to a $V_H$ level while $CLK$ is still at a $V_H$ level, then node N1 is unable to corrupt node $Q1$ because the only transient that could affect $Q1$ requires that the clock signal $CLK$ be at a $V_L$ level.

### Racefree operation between latches

The even number of inversions between latches has the additional purpose of preventing the clock race between two latches. For example, in Figure 2.15 the CLOCK pipeline is switching from the evaluate to the precharge/hold state.

- During the time period $t1$ latch $A$ is in a hold state and therefore any levels can be present at nodes $i$ and $N1$.
- At the end of the period $t1$ latch $B$ has finished loading the right value at $Q$.

- During the transition period $t2$ both clock signals $CLK$ and $\overline{CLK}$ are at the same $V_H$ level making both latches, $A$ and $B$, transparent for input rising transients or $V_H$ levels.
- If the rule of even number of inversions between latches is maintained then an input high on latch $A$ appears as an input low on latch $B$, which cannot propagate. Therefore $B$ will hold its old desired value. The race between data and the clock signal is non-existent by means of structural constraints.
- Similarly during the time period $t4$ both latches are transparent for only the falling transient or $V_L$ levels.




The even number of inversions is a structural constraint that eliminates the race between clock and data. The race originates from the overlap on the clock phase which enables the latches to propagate a single type of transient.

Figure 2.15: Protection Against Glitch Propagation.

### 2.5.3 N/P–C$^2$MOS Dynamic Latches

The N–C$^2$MOS and P–C$^2$MOS gates [JKS87] are a variation of the C$^2$MOS latch. The objective of these gates is tc eliminate one clock device and to avoid constraining the structure of its surrounding gates. These gates require a single clock phase and if driven by the N or P dynamic gates of Figure 2.16 they efficiently operate as latches under a true single-phase mode of operation of the clock signal.

#### Operation of an N–C$^2$MOS latch

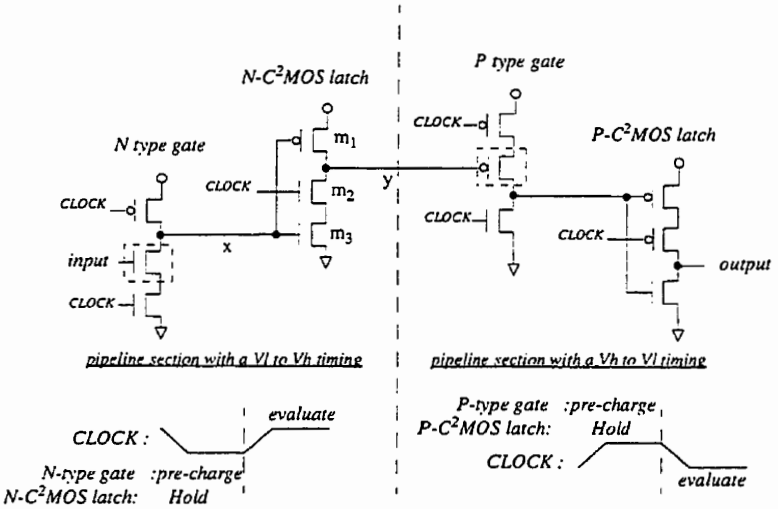
The operation of the N-C$^2$MOS latch is as follows. In Figure 2.16, During the precharge/hold state node $x$ has a VH level and device $m_1$ is off. Thus there is no need for the clocked PMOS transistor from the C$^2$MOS gate. Node $y$ has no conduction path to a supply node and it therefore holds the result from the previous evaluation.

The evaluate state is initiated when the clock signal switches from a VL to a VH level. The dynamic N gate is able to set the final value at node $x$ and, because device $m_2$ is on, the output node $y$ can load either voltage level.

#### Operation of a P–C$^2$MOS latch

The P-C$^2$MOS latch operates under an opposite timing for the precharge/hold and evaluate states. In Figure 2.16, a VH level in the clock signal marks the precharge of the P dynamic gate and the holding of the *output* level. The VL level in the clock signal controls the evaluate state of the input signals.

The opposite timing of events between both latches makes it possible for an N–C$^2$MOS latch to evaluate the level being held by a P–C$^2$MOS latch and vice versa.




Modified C²MOS latches eliminate one clock device since the precharge levels from the dynamic gates disable any conductive path from one of the evaluation blocks and can be operated with a single phase of the clock.

Figure 2.16: N/P–C²MOS latches

### 2.5.4 Single-Phase Clock techniques

The correct use of Dynamic logic gates with more than one clock phase has been addressed by techniques such as NORA. However their structural constraints are sometimes insufficient to protect a dynamic circuit against secondary effects such as Charge Feedthrough. This effect can be easily prevented but it further constraints the optimization of a circuit. Single-Phase clock techniques avoid Charge Feedthrough and all the structural complications of its various solutions. The single-phase clock technique are presented in the following sections, and the effects of Charge Feedthrough are explained in this section.

#### Charge Feedthrough

A problem to consider at the output of a gate when it has a fanout larger than one. *Charge Feedthrough* is a problem caused when the output node of a circuit is left floating as well as the devices driven by this node. Leaving the gate nodes of these transistors susceptible to charge changes in their capacitances, which can affect the potential on the gate nodes, and modify the operating conditions for other gates.

**Example 3** *The circuit in Figure 2.17 has the waveforms described in Figure 2.18. In this circuit node* N1 *is left floating for the period of time between 30.0ns and 34.75ns. During this period of time there are two major capacitors, the channel of the* N*-type gate provides a capacitor from* N1 *to* GND *($C_1$) and the channel of the* P*-type gate provides a capacitor from* N1 *to* VCC *($C_2$). The sequence of events for this failure are indicated in Figure 2.18.*

#### Solutions for Charge Feedthrough

A solution to this problem would either have to avoid the conditions over which a changed potential may corrupt the output of gates, or simply to eliminate those gates which create the problem:

- A solution can be implemented by eliminating all N and P type gates at the output of a pipe, or

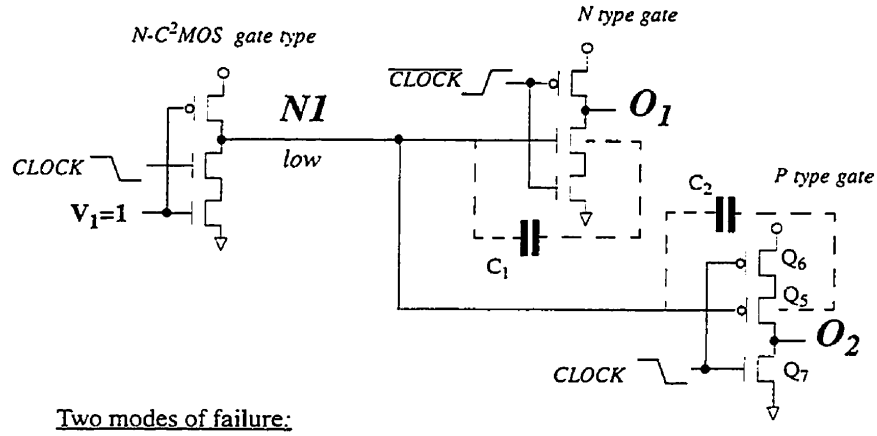



*Figure 2.17: Charge Feedthrough*

- use as input gates the N and N–C²MOS gate types (or only the P and P–C²MOS type gates)
- Another solution solves this problem by driving the floating nodes (O1 or O2) with static gates.

### 2.5.5 True Single Phase Clock (TSPC) technique

The TSPC [JS89] technique consists of a group of improved versions for the N/P–C²MOS gates driven by a gate of either the same type, N/P–C²MOS gates, or by precharged dynamic gates. In addition the N–P structures of Figure 2.7 are permitted between these latches and are not subject to structural constraints other than the alternation of the N and P dynamic gates.

For example, if the precharged N–type TSPC latch in Figure 2.19 is used, and there is an odd number of inversions between TSPC latches. The dynamic gate types before this N–type latch is a sequence of gates of the form **{"p-tspc"...–p–n–p–"n-tspc"}**. Whereas the use of an even number of inversions before the N–type latch will require that the sequence of gates to have the form along paths of **{"p-tspc"...–n–p–"n-tspc"}**.

The P–type TSPC latch has a similar constraint, where an odd number of inversions is required if the input sequence of gates is of the form **{"n-tspc"...–n–p–n–"p-tspc"}**, and an even number of inversions if the input sequence is of the form **{"n-tspc"...–p–n–"p-tspc"}**.

### Improved TSPC Dynamic Latches

The latches of Figure 2.19 are prone to generate glitches because of the propagation time from the first stage to the second during the transition period from the hold to the evaluate state [JS89]. For example, the following sequence of events are derived from driving the TSPC-N dynamic latch in Figure 2.20a with the waveforms of Figure 2.20b. The initial conditions are:

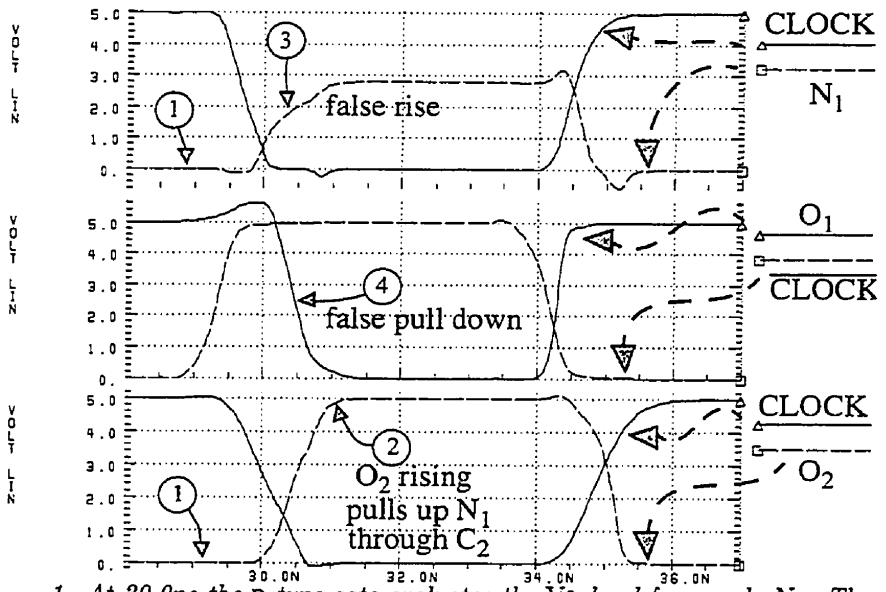
- The clock signal is at a $V_L$ level and the input at a constant $V_H$ level,
- node-$B$ is precharged at a $V_H$ level which discharges node-$C$, and
- the output is a dynamic node holding a $V_H$ initial voltage.

The transition state from 1ns to 4ns is given a slow clock signal to accentuate glitches. The events during this transition time are as follows:

- The rising clock signal drains node-$A$ first and then node-$B$,
- this same clock signal activates device $m_1$, and since $m_2$ still has a conduction channel, the output node discharges thru devices $m_1$ and $m_2$. The output node has been drained of its dynamic charge,
- as the clock rises and node-$B$ discharges, device $m_3$ becomes active and it charges the output with the true output level.

An improvement to prevent such glitches are the TSPC–2 latches in Figure 2.21. The difference is that the ground node of the second stage has been connected to node-$A$ to prevent node-$C$ from losing all its charge during the hold state.

The simulation of the N-TSPC-2 latch in Figure 2.22 shows that the glitch has been eliminated. Nodes A, B and C are precharged to the VH level and, during the transition state from 1ns to 4ns, nodes A, B and C are being discharged until node-B turns on the $m_3$ transistor. The charge at the output node is discharged but at a slower rate because node C is discharged first.



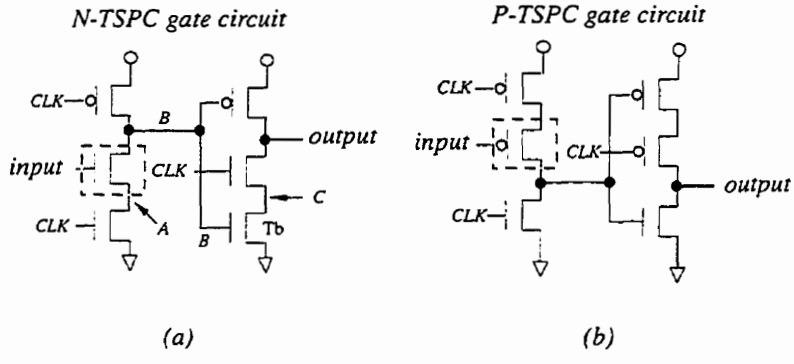

1. *At 30.0ns the* P*-type gate evaluates the* VL *level from node* $N_1$. *The channel of* $Q_5$ *is initially discharged.*

2. *At 31.0ns the gate charges the* $O_2$ *node from a* VL *to a* VH *level.*

3. *Between 30.0ns and 31.0ns* $Q_5$ *and* $Q_6$ *are on. The channel of* $Q_5$ *rises toward 5v pulling up* $N_1$ *vis capacitor* $C_2$.

4. *The increased potential at* $N_1$ *is approximately proportional to* $C_2/(C_1 + C_2)$. *This voltage is enough to corrupt the output* $O_1$ *of the* N*-type gate.*

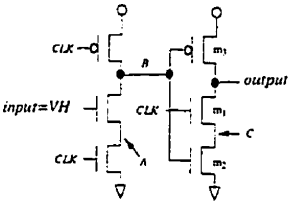
*Figure 2.18: Waveforms demonstrating the Charge Feedthrough effect*



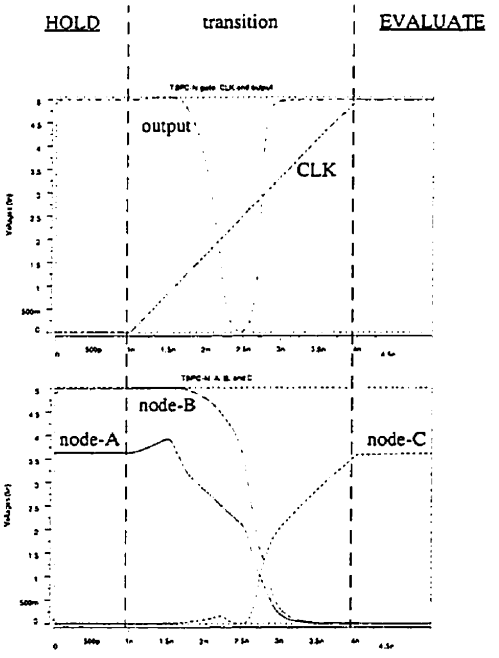

Single phase operation is achieved by combining latches and gates.

Figure 2.19: True–Single Phase Clock Technique

a)

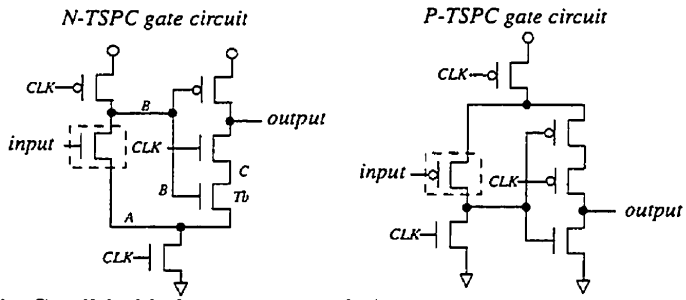



b)




As the clock rises the output will discharge through $m1$ and $m2$ before $B$ discharges. This is because $B$ is still partially pulled up until the clock is over half risen.

Figure 2.20: Glitches generated by TSPC gates.




Node $C$ will hold charge even with $B$ at a $V_H$ level until the clock rises.

Figure 2.21: Optimized TSPC-2 as suggested by Jiren and Svensson

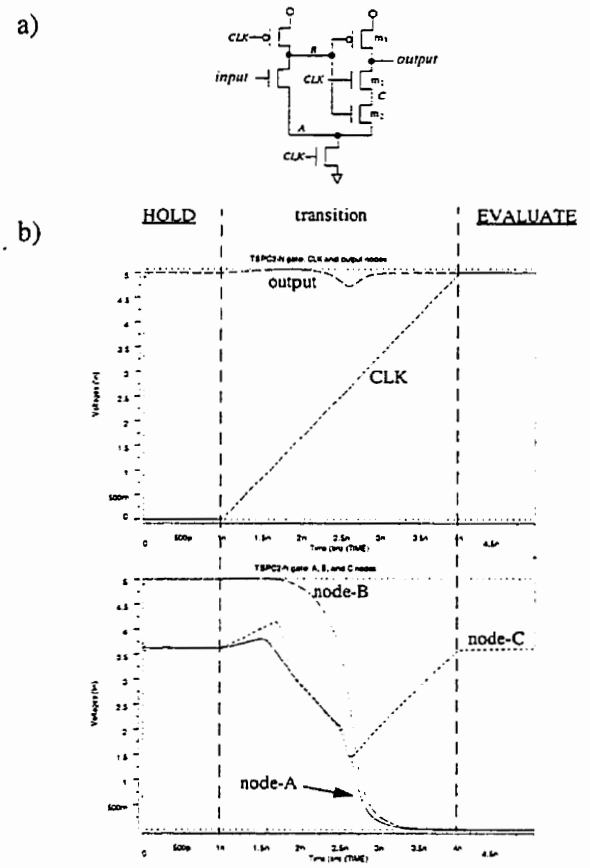
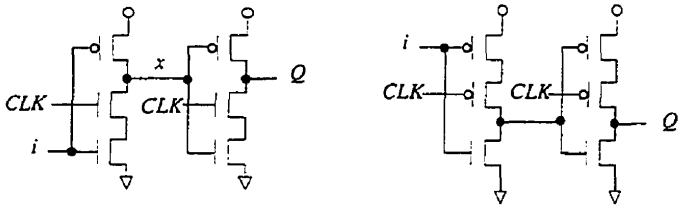



Figure 2.22: Waveforms for the optimized TSPC-2 latch

### Split-Output TSPC Dynamic Latches

The TSPC technique provides another alternative to the dynamic precharged latches when using the doubled N-C$^2$MOS and P-C$^2$MOS gates in Figure 2.23. A doubled N-C$^2$MOS gate replaces an N-TSPC latch, and a doubled P-C$^2$MOS gate replaces the P-TSPC latch.




*(a)* *(b)*

a) doubled N-C$^2$MOS and b) doubled P-C$^2$MOS

Figure 2.23: Doubled N/P-C$^2$MOS latches

The states of operation for these latches are *evaluate* and *hold.* The hold state is used instead of precharge because, when doubled, the modified N/P-C$^2$MOS latches require no fixed input precharge levels.

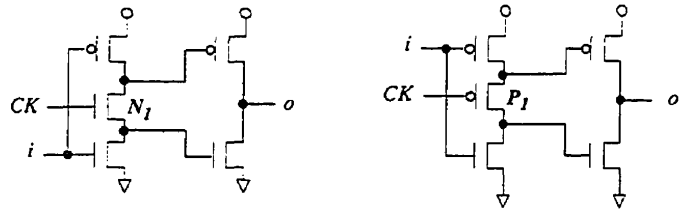
For example, the latch in Figure 2.23a is a doubled N-C$^2$MOS latch:

- When the clock $CLK = V_H$ the latch is transparent, similar to two static inverters in series.
- When the clock $CLK = V_L$ the latch is holding the output level. All the NMOS devices are disconnected from $x$ and $Q$:
  - If $i = V_H$ then both nodes $x$ and $Q$ are floating, and
  - if $i = V_L$ then node $x = V_H$ and node $Q$ is still floating.

Therefore, if node $i$ has a transition from $i = V_H$ to $i = V_L$ then node $x$ will switch from $x = V_L$ to $x = V_H$ but node $Q$ remains unchanged. The implication from this mode of operation is that there are no constraints on the kind of gates between latches. The only exception is the trivial case

where the mode of operation for the dynamic gates before the latch and that of the latch should be the same.

The circuits in Figure 2.23 are optimized into the *Split-Output* latches of Figure 2.24, where only one transistor is controlled by the clock signal. A drawback from using the Split-Output latches is that no node will have a full voltage swing because there is a threshold drop in devices $N_1$ and $P_1$ of Figure 2.24.




*(a)* *(b)*

a) N-type TSPC Split-Output latch and b) P-type TSPC Split-Output latch

Figure 2.24: Split-Output latches

### Improved TSPC latches in the Alpha Microprocessor

The importance of the TSPC technique has been demonstrated by commercial products such as the *Alpha Microprocessor* [D+92]. The reduced number of constraints in TSPC latches when compared to the NORA technique allows for an easier interaction with non-dynamic circuits and are still able to benefit from using the smaller latches.

Figure 2.25 summarizes the optimizations to the TSPC latches as used in the *Alpha Microprocessor*, where the purpose of this optimization is to provide a "weak feedback device".

For example, when the active-high latch in Figure 2.25a has a clock signal at a VL level, the input at a VH level and the output at a VL level; then the internal node $x$ will be holding a VH level and the output remains at a VL level. If node $x$ has a glitch, from either capacitive couplings or charge

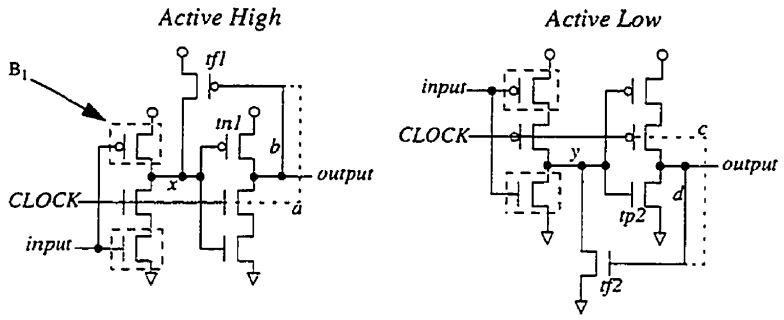



Figure 2.25: Optimized TSPC latches as implemented in the Alpha microprocessor

redistribution in the PBLOCK $B_1$, then device $tn1$ will charge the output and it will stay there, with the wrong output, until the clock initiates the next evaluate state.

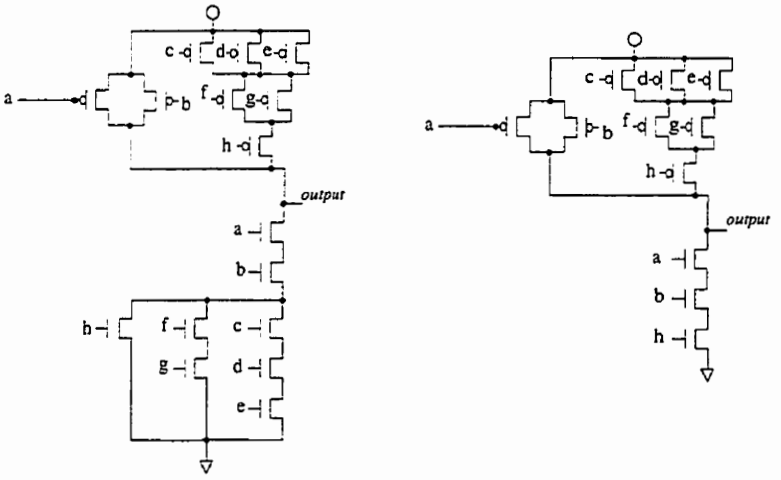
The feedback device $tf1$ provides better noise immunity and the signal driving its gate node is provided according to the structure of the $B_1$ PBLOCK. If the devices in $B_1$ are not in series then $tf1$ is driven by connecting wire $b$, whereas the existence of devices in series within $B_1$ requires driving $tf1$ with wire $a$ [D+92]. In general wire $b$ is used unless more current is required.

### 2.5.6 Clock and Data Precharged Dynamic (CDPD) technique

The Clock and Data Precharged Dynamic circuit (CDPD) technique [YSL93] evolved from the Domino logic technique. The gates from this technique replace portions of a Domino logic circuit where static inverters are deemed redundant whenever their purpose is only to provide the right precharge levels to a dynamic gate.

The L/H and H/L gates are the optimized versions for static gates. The base for this optimization is the knowledge that all input signals will have the same voltage level during precharge, therefore one logic block will precharge

the output. The output is expected during the evaluate state to either stay at the precharged level or perform a single transition, which can only be done by a complementary logic block as illustrated in Figure 2.26.

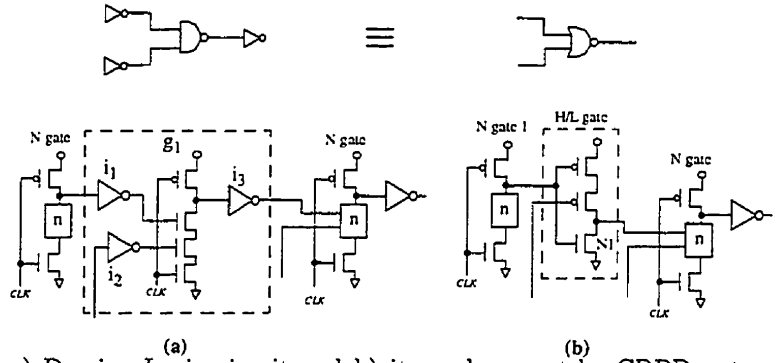



(a) A static gate can always be utilized instead of L/H and H/L gates. However, if the input waveforms are of the right type then this static gate can be optimized to the circuit in (b) without affecting the shape of the output waveforms.

Figure 2.26: H/L gate alternative to a static gate

For example, the Domino logic circuit of Figure 2.27a shows a dynamic gate and the inverters which are redundant. The inverters $i_1, i_2$ and $i_3$ are eliminated and the logic in gate $g_1$ is modified by applying DeMorgan's theorem. The replacement for the inverters and gate $g_1$ is the H/L gate of Figure 2.27b. The inputs to this H/L gate precharge to a $V_H$ voltage and, when propagated, the output is precharged to a $V_L$ voltage.

A longer chain of CDPD gates is built by alternating H/L and L/H gates to optimize both Domino logic and TSPC circuits, the reason being is that both techniques use N-type dynamic gates. However, a limitation exists for




a) Domino Logic circuit and b) its replacement by CDPD gates. The precharge on the H/L gate is supplied by $N1$ when the N-type gate-1 is precharged.

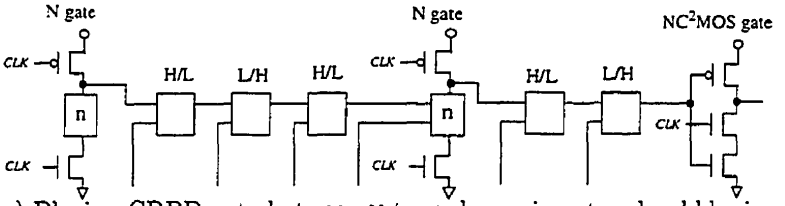
Figure 2.27: Replacement of Domino gates by CDPD gates

some logic gates where, in the worst case, a CDPD gate is identical to its static logic counterpart and may not reduce the total number of devices.

The requirements for using the replacement H/L and L/H gates are derived from those in Domino logic and TSPC:

- An odd number of CDPD gates are required between N-type dynamic gates. Each CDPD gate replaces three gates, and so, if an odd number of CDPD gates between two dynamic gates is maintained then the precharged level from the first gate, a $V_H$ level, is propagated as an input $V_L$ precharged level to the other dynamic gate. Therefore for the circuit in Figure 2.28a no race exists between data and the clock signal.

- An even number of CDPD gates are required between an N-type dynamic gate and an N-C$^2$MOS gate. The dynamic gate precharges to a $V_H$ level, therefore it can only drive a H/L gate. The N-C$^2$MOS gate needs a precharge input level of $V_H$ and it can only be obtained from a CDPD L/H gate. Therefore, the correct operation of the N-C$^2$MOS gate is only possible by using an even number of CDPD gates between the two.




a) Placing CDPD gate between N-type dynamic gates should be in odd numbers, and b) even numbers of CDPD gates between N-type gates and a N-C²MOS gate.

Figure 2.28: Cascaded H/L and L/H gates.

### 2.5.7 All-N-Logic technique

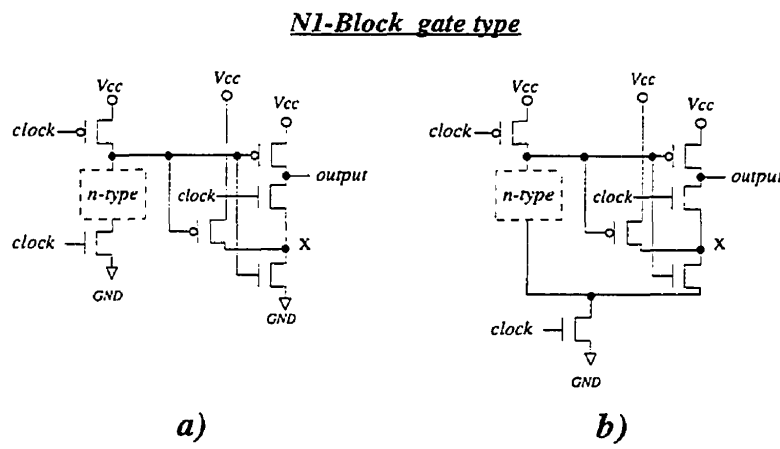
The *All-N-Logic* technique [GE96] provides a modified version of the TSPC-N latch and an efficient alternative for the TSPC-P latch. This technique introduces two types of latches named the *N1-block* and the *N2-block*.

The N1-block is an improved version of the N-type TSPC latches, where a single P-type device is added to speedup charging the internal node $x$ of the circuits in Figure 2.29. The circuits in Figure 2.29a and Figure 2.29b are the alternatives to the N-type TSPC-1 and TSPC-2 latches respectively, and are operated under the same timing of events which is controlled by the clock signal.

#### N2-Block latch

The N2-block is an alternative circuit for the P-type TSPC latch. The objective of this latch is to use the NMOS devices for the evaluation block instead of PMOS devices. The type of transistors used for the logic evaluation block indicates, in most dynamic gates, which node is the output. This node would be located where both NMOS and PMOS devices meet. This is node $a$ in Figure 2.30a.

However, the N2-block is designed differently because the node where the NMOS clock device meets the N-block is the output, which is node $b$ in Figure 2.30a. As a consequence the output levels from this N-type block are




a) N1-block replacement for N-type TSPC-1 latches, and b) for TSPC-2 latches. The feedforward device at node $x$ is added to speed up a rising transient at the output node.

Figure 2.29: Circuit schematics of the all-N logic N1-blocks

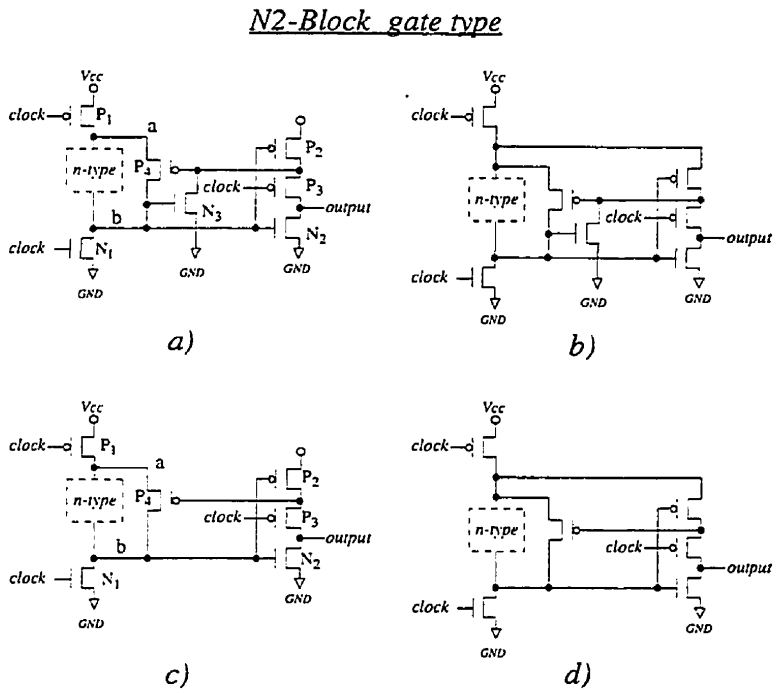
unable to reach the $V_H$ voltage level because of the threshold drop.

The N2-block latch solves this threshold drop problem with the positive feedback device $p4$ of Figure 2.30a. The operation of this latch is as follows:

- When the clock is $V_H$ the precharge state takes place and node $b$ is precharged to $V_L$, device $N_3$ is off and $P_2$ is on, device $P_4$ is off and the output holds the previous evaluation.
- When the clock switches to $V_H$ the evaluate state takes place and, if the N-type logic block is providing a conduction path then node $b$ will start charging with the current coming from the N-block.
- When node $b$ reaches the threshold voltage it will change to a full $V_H$ level mostly by the current from the bypass transistor $P_4$ because now device $N_3$ is conducting.

Device $N_3$ of the N2-block latch in Figure 2.30a can be eliminated if the clock signal has sharp slew rates. The improved versions are illustrated in

Figure 2.30c and Figure 2.30d and in these circuits the devices triggering $P_4$ are the $N_2$ and $P_3$ devices.



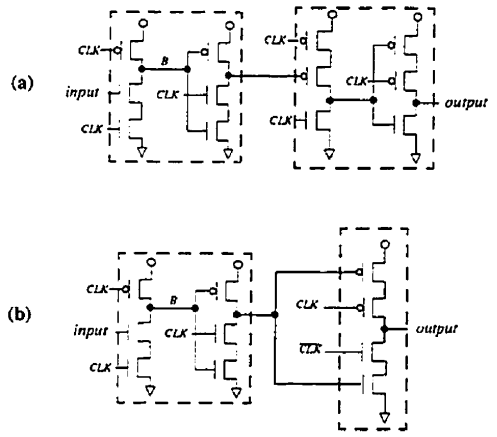

a) N2-block replacement for P-type TSPC-1 gates, and b) the N2-block replacement for P-type TSPC-2 gates. The optimized versions when the clock has a sharp slew rate are c) replacements for P-type TSPC-1 gates, and d) the replacement for P-type TSPC-2 gates.

Figure 2.30: Circuit schematics of the all-N logic N2-blocks

A very important characteristic of the N2-block latches is that the first stage is evaluating a non-inverting logic function, therefore, the N2-latch is inverting. An extra degree of freedom is gained from this technique because both the TSPC and the All-N logic latches are compatible, and the P-type latch can either be inverting (N2-latch) or non-inverting (TSPC and N2-block) without additional penalties.

### 2.5.8 TSPC Full Latches (Flip-Flops)

The TSPC full-latches contain a single stage $C^2$MOS like gate that replaces the P-type TSPC latch [YS97]. The master slave TSPC flip-flop of Figure 2.31 is transformed into a TSPC full-latch by replacing P-TSPC latch with a $C^2$MOS latch. The $C^2$MOS latch has two clock devices, and is made into a single-phase clock latch when one device receives the clock signal and the other device receives the output from the dynamic gate in the N-TSPC latch, this is node $x$ in Figure 2.32a. The mode of operation for the circuit in Figure 2.32a is as follows:




a) A master slave flip-flop using TSPC latches and b) equivalent using a $C^2$MOS latch.

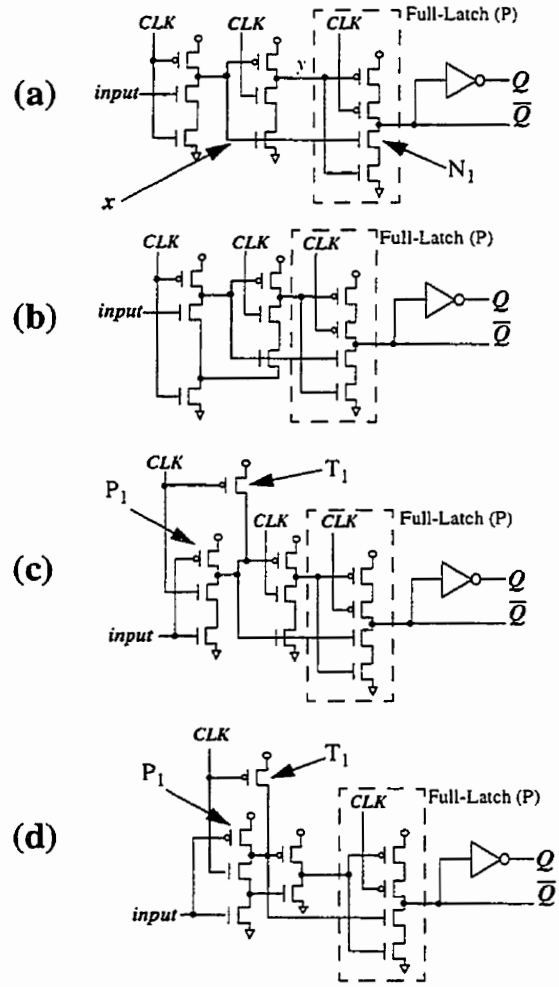
Figure 2.31: TSPC flip-flop

- When the clock is at a $V_L$ level node $x$ is set to $V_H$ and node $y$ holds its voltage level, therefore the TSPC Full-Latch is in the evaluate state and the N-TSPC latch is in precharge.
- The clock switches to a $V_H$ level and the N-TSPC latch will turn off device $N_1$ if node $x$ is evaluated to a $V_L$ level and, since the N-$C^2$MOS gate is just an inverter, node $y$ will be the opposite level of node $x$.

- Complementary levels at nodes $x$ and $y$ imply that the output voltage at $\overline{D}$ has no conduction path to a supply node and, subsequently, the C$^2$MOS latch is holding its voltage during the VH level of the clock signal.

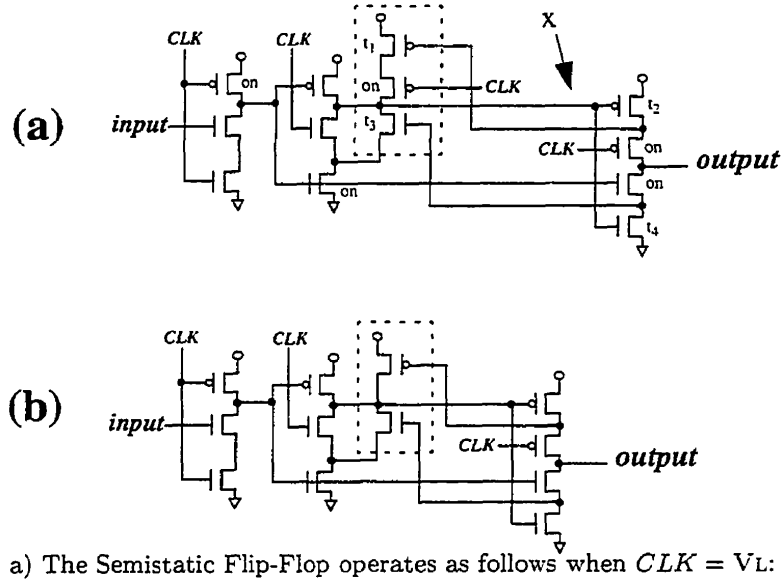
The replacements for the various P-TSPC latches and their interconnection to the N-type TSPC latches is illustrated by the full-latches (flip-flops) in Figure 2.32. The devices $T_1$ in Figure 2.32c and Figure 2.32d are required to set the C$^2$MOS latch in three-state during precharge. The $T_1$ device makes the logic block in $P_1$ redundant, but it is left and kept to a minimum size since it prevents charge sharing and maintains the noise levels.

The TSPC full-latch is made into a "semi-static" version by using the arrangement in Figure 2.33a. This latch is semi-static because it provides a conduction path for the output of the N-TSPC latch when the clock is at the VL level. A simplified version is illustrated in Figure 2.33b. This conduction path allows to stop the clock signal during the hold state of the TSPC master latch however it still contains the same constraints as before regarding the slew rate of the clock signal.




a) N-TSPC-1 to FL(P) b) N-TSPC-2 to FL(P) c) doubled N-C$^2$MOS to FL(P) and d) Split-Output to FL(P)

Figure 2.32: Dynamic Full-Latches




a) The Semistatic Flip-Flop operates as follows when $CLK = V_L$:

- If node $X = V_L$ then this node has a conduction path to the GND because $t_2 = on, t_4 = off, t_1 = off$ and $t_3 = on$
- If node $X = V_H$ then this node has a conduction path to the supply because $t_2 = off, t_4 = on, t_1 = on$ and $t_3 = off$

b) is the simplified version. In both circuits the size of the devices in the dashed box should be kept to a minimum to reduce the output load of the N-TSPC latch.

Figure 2.33: Semistatic Full-Latches

# Chapter 3

# Analysis of Dynamic Logic

This chapter presents the analysis for the dynamic logic techniques in Chapter 2. The first step for understanding how the various techniques interact is by establishing their fundamental components and by defining a common framework. The first section of this chapter presents both, the fundamental gate methods extracted from the gate circuits in Chapter 2, and a binary-signal notation that is appropriate for analyzing dynamic logic.

The next step is to determine the output response of the fundamental gates, and the definitions which indicate when is this output response providing the correct logic values. The last step in the analysis explores, with the fundamental gates, the interconnections leading to all possible waveform behaviors.

The results after searching for feasible interconnections is summarized by the general description of the waveform behaviors and their relationship to the fundamental gates. This information is the core of the new circuit design technique. The last section describes the problems solved by the new technique.

## 3.1 Fundamental Gates and Timing Framework

The fundamental methods for designing gate circuits in dynamic logic are extracted from the techniques of Chapter 2 and are explained in this section. These methods do not specifically include the replacement circuits utilizing the CDPD, and the TSPC-Full-Latches nor the optimizations found in the TSPC-2, Split-Output and NTP circuits. The reason is because these circuits are all custom optimizations of the same fundamental gates and will have the same behavior when analyzed under the timing framework of Section 3.1.2.

The timing framework is defined for a *single-phase clock* mode of operation. Therefore a gate such as the $C^2$MOS, which is a fundamental gate, is not considered in this chapter because of its double-phase clock requirement.

### 3.1.1 Fundamental Design Methods for Gates

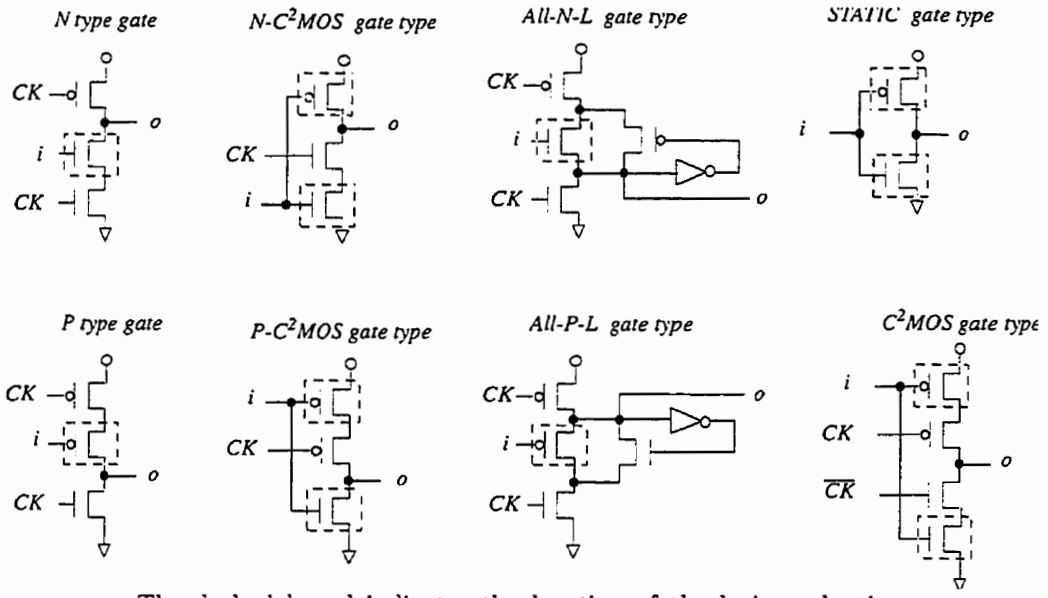
The fundamental circuit design methods for CMOS logic gates are illustrated in Figure 3.1. Static gates, N type and P type dynamic gates are used by most techniques and need no further justification.

Similarly, the N–$C^2$MOS and P–$C^2$MOS gates are fundamental gates and are included without optimizations.

#### Gates from the ALL-N logic technique

The All–N–L gate is a non–inverting gate extracted from the N2–Block inverting latch of the All-N-Logic technique. The extracted All–N–L gate is found when the schematic for the N2-Block, in Figure 3.2a, is simplified to the equivalent circuit in Figure 3.2b and further modified in Figure 3.2c. The circuit found in Figure 3.2c is the same N2–block from the functional point of view. The modified schematic shows that two gates are found in the N2–Block, the All–N–L gate and a new latch.

The N1–Block in Figure 3.3a is also optimized and is simplified to the schematic in Figure 3.3b. This simplified circuit also reveals the existence of the new latch. However the circuit for this ubiquitous latch does not provide a new functional structure. When the new latch is compared to the




The dashed boxed indicates the location of the logic evaluation block.

Figure 3.1: Fundamental Gate Design Methods

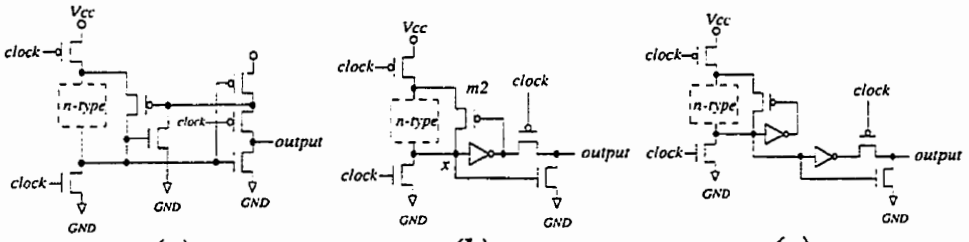
N–C$^2$MOS gate in Figure 3.4 it is clear that both circuits drive the output node in exactly the same way.

**Example 4** *Figure 3.4a shows the netlist for the new dynamic register extracted from Figure 3.2c, together with the equivalent circuit for different values of the clock signal.*

*The netlist for the* N–C$^2$MOS *gate from Figure 3.1 is also illustrated in Figure 3.4b together with the equivalent circuit for different values on the clock signal.*

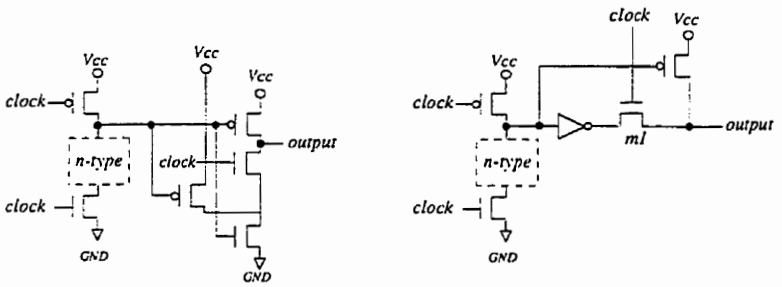
*The equivalent circuits for each gate type are self explanatory and they clearly illustrate that both schematics have the same behavior.*

The simplified N2-block of Figure 3.2a is an optimized version of the interconnection between the All–N–L gate, in Figure 3.1, to the N–C$^2$MOS gate, as a result the contribution from the All-N logic technique are the non-inverting All–N–L gate and its P-type counterpart the All–P–L gate.




Alternative schematic representations of the same circuit, a) N2-block, b) simplified schematic and c) equivalent circuit.

Figure 3.2: N2-Block inverting latch




a) N1-Block and b) simplified schematic

Figure 3.3: N1-Block non-inverting latch

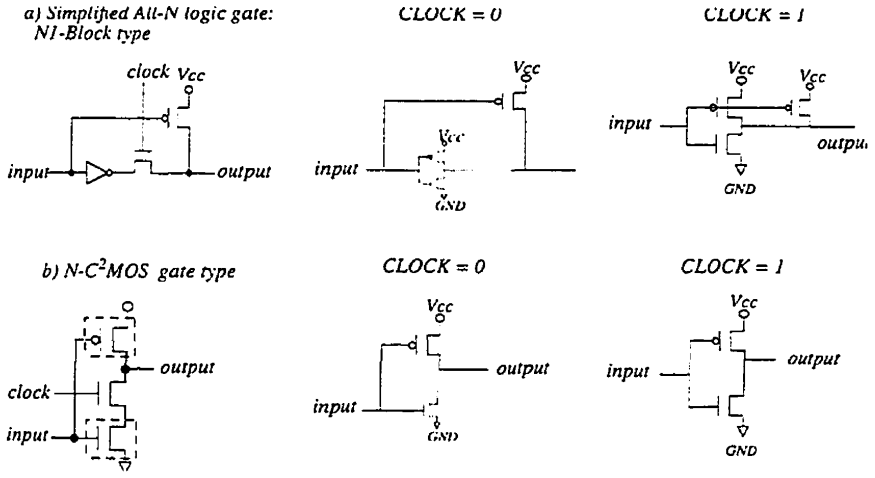



*Figure 3.4: Comparison between equivalent schematics*

### 3.1.2 Timing Framework

This section explains the timing framework for single-phase clock circuits and the symbols introduced to describe the shape of a waveform signal.

The following definitions are used to describe a signal:

- A *transient* is a signal described in the time domain whose plot of time vs. voltage can have any shape. The time period of such plot is defined by the regions of the timing framework which is introduced in this section. The timing framework identifies these transients with the symbols $\nearrow$, $\searrow$, 0, and 1 as defined in this section. The extended timing framework of Chapter 5 adds the $\perp$ and $\top$ symbols to this set of transients.

- A *transition* is a transient signal which has a full voltage swing from the lowest voltage ($V_L$) to the highest ($V_H$), or vice versa. The timing framework identifies these transitions with the symbols $\nearrow$ and $\searrow$.

- A *level* is a stable transient signal either at the $V_L$ or the $V_H$ voltage. The timing framework identifies these levels with the symbols 0 and 1.

- A *glitch* is any transient signal whose shape is anomalous and it cannot be identified as a level or a transition. The extended timing framework of Chapter 5 identifies these glitches with the symbols $\perp$ and $\top$.

The voltage levels VL and VH are used to describe binary values, therefore their definition is not a fixed voltage level but a center value with a tolerance of usually $Vt$ volts from the full voltage swing.

### Timing Types

A transition in the clock signal followed by a stable level, 0 or 1, corresponds to the first state which is *precharge*, and the next transition followed by an opposite level corresponds to the *evaluate* state.

If the stable level during precharge is VL then the stable level for the evaluate state is VH. This VL to VH timing defines a CKLH(N) *timing type* and is illustrated in Figure 3.5. Similarly, if the VH level of the clock signal is used for the precharging and a VL level to evaluate then this VH to VL timing defines a CKHL(P) *timing type.*

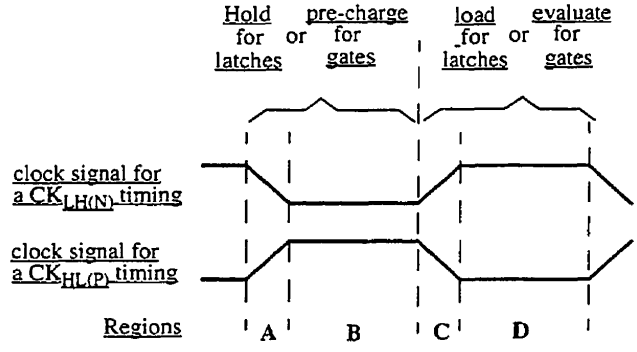



Figure 3.5: Timing Scheme Types and Region Identification

### Transient Regions and Waveform Shapes

A waveform is defined to have four transients during a single clock cycle, one for each of the four regions in the clock signal. The first region is "A"

and in this region the clock signal has a full swing transient that ends in the clock's precharge voltage level. The second region is "B" where the clock signal remains stable at the precharge voltage level.

Similarly for regions "C" and "D", the clock signal will have a full voltage swing in region C and it will remain at that level for the duration of region D. Both regions, A and B, define the time period for the precharge state, and regions C and D the evaluate state.

Inside these regions only one type of transient is allowed. The symbols which define these transients are:

- The "0" symbol defines a signal which remains at the $V_L$ level
- The "1" symbol defines a signal which remains at the $V_H$ level
- The "/" symbol defines a transition signal that has a full voltage swing from $V_L$ to $V_H$
- The "\" symbol defines a transition signal that has a full voltage swing from $V_H$ to $V_L$

The shape of a feasible waveform is not allowed to switch from one logic level to the opposite without the appropriate transient such as \ or /. Therefore, the four regions of activity and four possible transients per region describe the 32 waveform shapes of Figure 3.6.

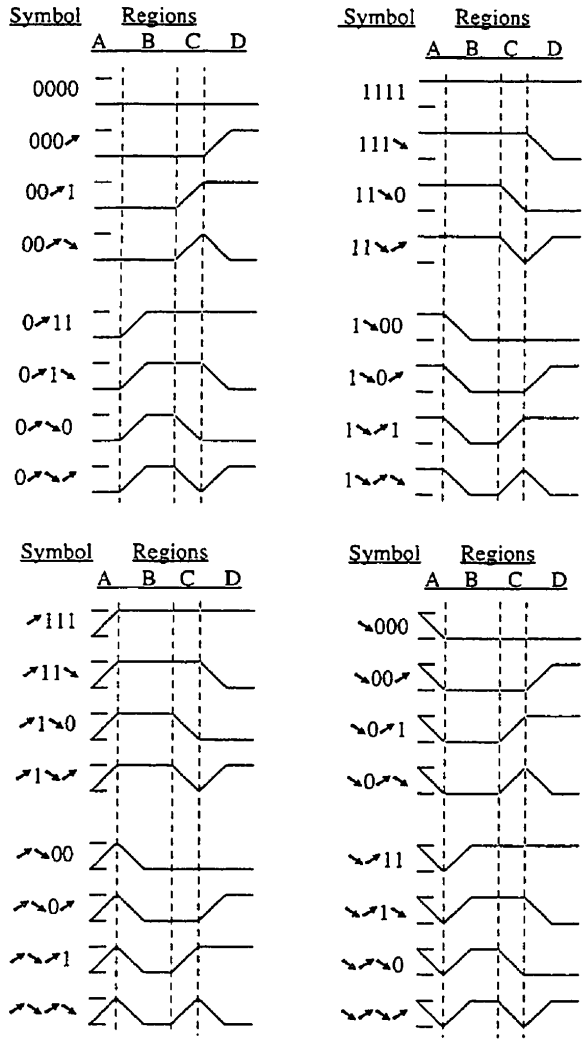
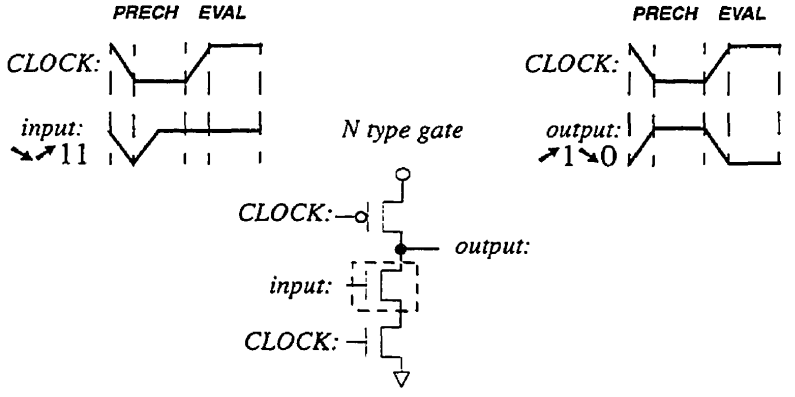



Figure 3.6: All 32 possible waveforms described by the timing framework

**Example 5** *The* N*-type dynamic gate of Figure 3.7 is controlled by a* CKLH(N) *timing, therefore the clock signal has the "\0/1" waveform. The given input signal has a \/11 waveform and the generated output signal has a /1\0 waveform.*




*A* CKLH *timing for an N type gate uses a ↘0↗1 waveform for its clock. When an input signal with a ↘↗11 waveform is applied the result is a ↗1↘0 waveform at the output.*

*Figure 3.7: Example Using Transient Symbols for Waveforms*

## 3.2 Optimized Gates and Interconnections

The techniques introduced in "Chapter 2 background" presented various optimizations for their gates and their interconnection. These optimizations can still be used with the fundamental gates, the following is a summary:

- Optimized N and P gates against *charge sharing* and *charge leakage.* A full description is given in Section 2.4.5 and Section 2.4.4.
- Optimized N–C$^2$MOS and P–C$^2$MOS gates against *charge sharing.* N–C$^2$MOS and P–C$^2$MOS gates can have a weak feedback device to improve their noise immunity as implemented in the Alpha chip, see Section 2.5.5. In addition, from the All-N-Logic technique in Section 2.5.7 a device can

be introduced to improve the noise margin and increase the switching speed. This is device $p_1$ in Figure 3.8.

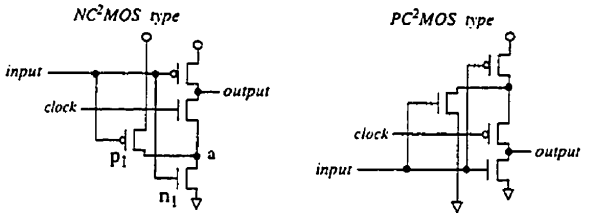
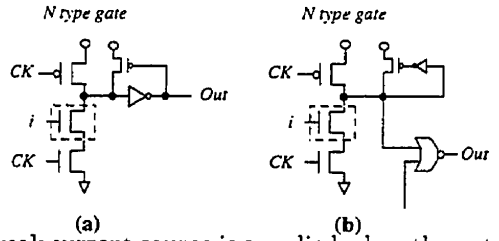



Figure 3.8: Dynamic latch improvements for charge redistribution

- Optimized N to Static (or P to Static) gate interconnections. This type of interconnection is the Domino logic technique described in Section 2.4.2 for which there are additional optimizations found in the CDPD technique of Section 2.5.6. Preventing charge sharing and leakage is generalized in Figure 3.9. When the static gate at the output is a buffer then a single device can supply the current. However if the static gate is more complex than a buffer then the current is provided with a device that is driven with a small static inverter.




A weak current source is supplied when the output gate is a (a) static buffer, or (b) a static complex gate.

Figure 3.9: Improvements specific to Domino Logic circuits

- Optimized N to N–C$^2$MOS (or P to P–C$^2$MOS) gate interconnections. The True-Single-Phase-Clock technique (Section 2.5.5) introduces the TSPC-

2 circuit to optimize this type of interconnections. The All-N-Logic technique has an optimized version and is described in Section 2.5.7.

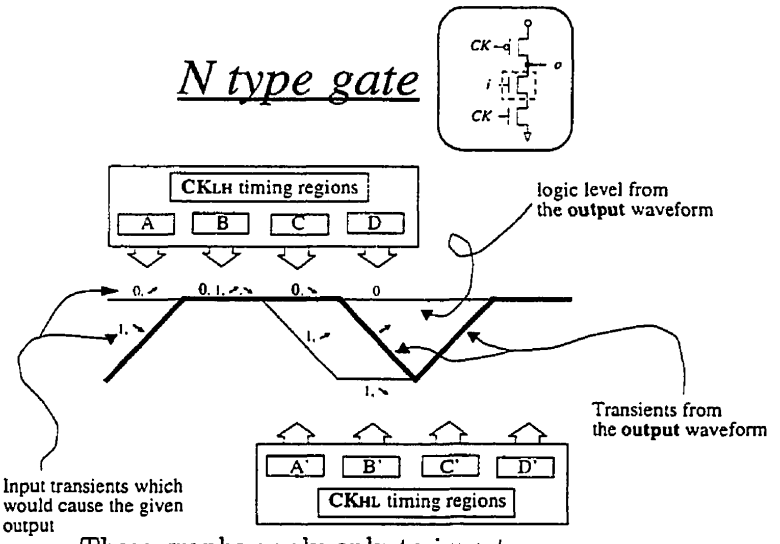
- Optimized N–C$^2$MOS to N–C$^2$MOS (or P–C$^2$MOS to P–C$^2$MOS) gate interconnections. The True-Single-Phase-Clock technique (Section 2.5.5) introduces the Split-Output circuit to optimize this type of interconnections. The other alternative to this type of interconnection is the TSPC full-latch which entirely replaces these two gates with the FL(N) and the FL(P) latches Section 2.5.8.

- Optimized All–N–L to P–C$^2$MOS (or All–P–L to N–C$^2$MOS) gate interconnections. The All-N-Logic technique (Section 2.5.7) introduces the "N2-block" and the "revised N2-block" circuits to optimize this type of interconnections.

## 3.3 Waveform Response Graphs (WRGs)

This section describes the *waveform response graphs* for each gate type. The components of this *waveform response graphs* are edges and labels, where the labels indicate which transient symbols of the input signal are matched to generate the output transient depicted by an edge in the graph.

Figure 3.10 shows the shape of the output signal when an N-type gate has an input described by the \00/ waveform. The additional regions, C´ and D´ are a copy of the contents in the A and B regions.




- These graphs apply only to inverters.
- The nodes and edges indicate the shape of the **output waveform**.
- The labels 0,1,/, and \ indicate the shape of the **input waveform** on the specified region.

Testing for all 32 possible input waveforms and the recording of the output response from an N-type gate is graphically summarized by this Waveform Response Graph.

Figure 3.10: Example of a Waveform Response Graph

### Compatibility with Gates of Opposite Timing

The clock waveform that is used to study an N–type dynamic gate is the CKLH(N) timing and a P–type dynamic gate uses the CKHL(P) timing. These are opposite timings and to analyze their interaction is necessary to have their signals defined within the same timing.

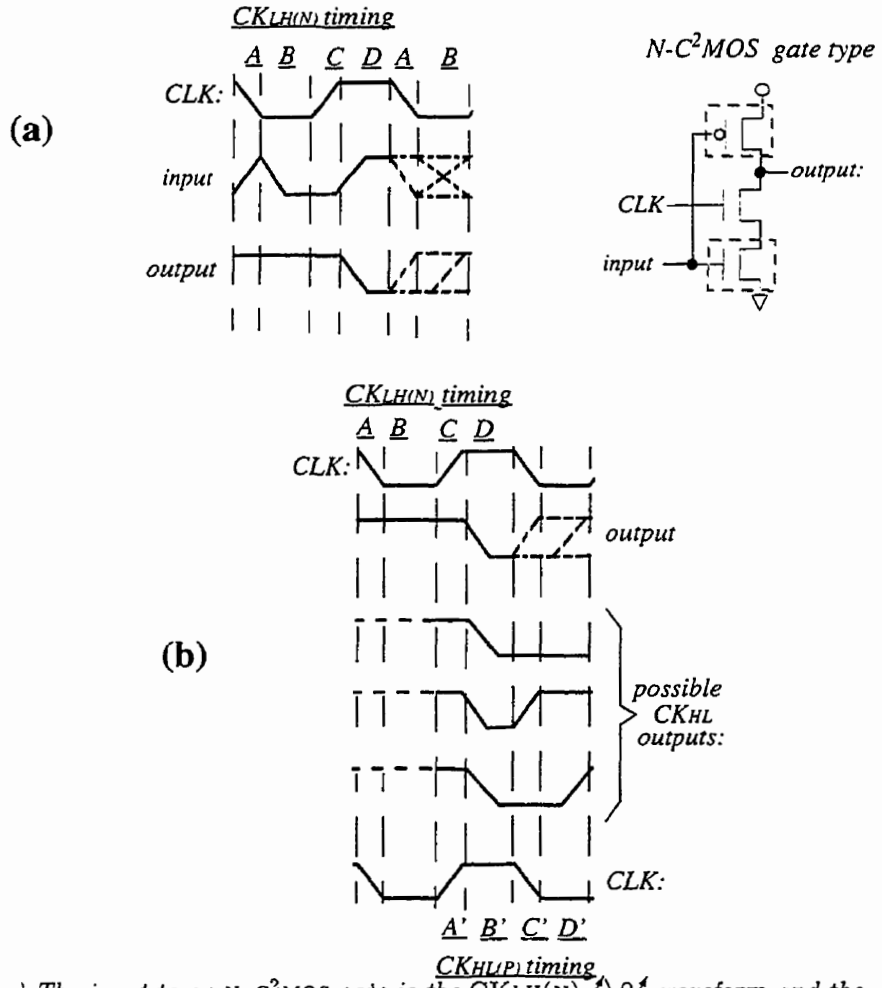
The output response of a buffer, under an opposite timing, is found from the *waveforms response graphs* by copying the contents of regions C and D into regions A´ and B´ of the opposite timing, and the transients in regions C´ and D´ are obtained from all the input possibilities for regions A and B

**Example 6** *Figure 3.11 shows an* N–$C^2$MOS *gate with an input waveform of* /\0/ *, the output is the* 111\ *waveform. The additional transients marked with dashed lines are all the alternatives which would follow the input waveform. The possible output transients are also illustrated.*

*The additional transients are used to determine the possible output waveforms if the* border conditions *(which are explained in the next section) are met relative to the output gate. When such conditions are met then this* 111\ *waveform, pertaining to the* CKLH(N) *timing, belongs to the opposite timing. A change of timing implies that a waveform must be defined within the* A'B'C'D' *regions of the* CKHL(P) *timing context.*

*The regions* A' *and* B' *have the same transients as* C *and* D*, which is the sequence* $A'B' = CD = 1\backslash$. *This is different from the contents of the* C' *and* D' *regions because their contents are the* A *and* B *regions of the next cycle, which is unknown and has been filled with all possibilities.*

*Therefore the possibilities for regions* C' *and* D' *are all the signals obtained with* VL *as initial conditions. The set of possible waveforms under a* CKHL(P) *timing are* $A'B'C'D' = \{1\backslash 00,\ 1\backslash /1, 1\backslash 0/\}$.

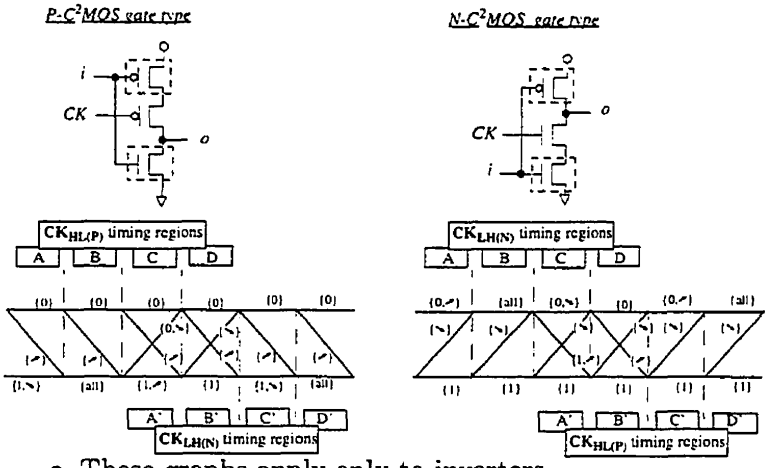



*a) The input to an* N–C²MOS *gate is the* CKLH(N) /‾\0/ *waveform and the output response is the* CKLH(N) 111‾\ *waveform. b) The extension of the* CKLH(N) *waveforms makes three possible waveform shapes whenever the output of this gate is crossed to a* CKHL(P) *timing.*

*Figure 3.11: Example of Opposite Timing Interfacing*

### 3.3.1 WRGs' for $N - C^2MOS$ and $P - C^2MOS$ Gates

Figure 3.12 shows the waveform response graphs for the N-C$^2$MOS and P-C$^2$MOS gates.




- These graphs apply only to inverters.
- The nodes and edges indicate the shape of the **output waveform.**
- The labels 0,1,↗, and ↘ indicate the shape of the **input waveform** on the specified region.
- The label "all" gives the specified output regardless of the input.

Figure 3.12: Waveform Response Graph for P–C$^2$MOS and N–C$^2$MOS gates

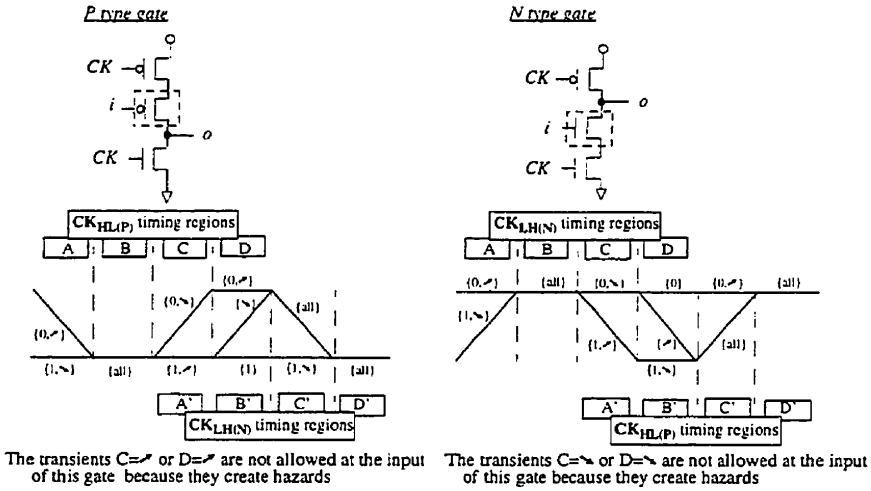
### 3.3.2 WRGs' for N and P Gates

The WRGs for N and P dynamic gates in Figure 3.13 indicate a restriction on the input transitions in regions C and D. Without a restriction in region D these unwanted transitions will corrupt the output data by losing the precharged voltage.

For example, the N type dynamic gate precharges to a VH level. If at the beginning of the evaluate state the input is at a VH level then it will discharge the output node. If this initial VH level is part of a \ transition during the evaluate state, then the output node is unable to provide the required VH voltage. Because this level can only be loaded to the output by the PMOS transistor which is driven by the clock signal.

The restriction for region C also prevents the unwanted loss of charge. For example, when the input data waveform and the clock signal switch simultaneously an uncertain behavior will be induced into the gate, because both signals are racing each other as described in Section 2.4.2. This race condition is avoided by marking as unfeasible the input waveforms with a destructive transition in region C.

The correct mode of operation for the N and P dynamic gates is to either maintain the precharged level or discharge it just once with the appropriate input waveform.

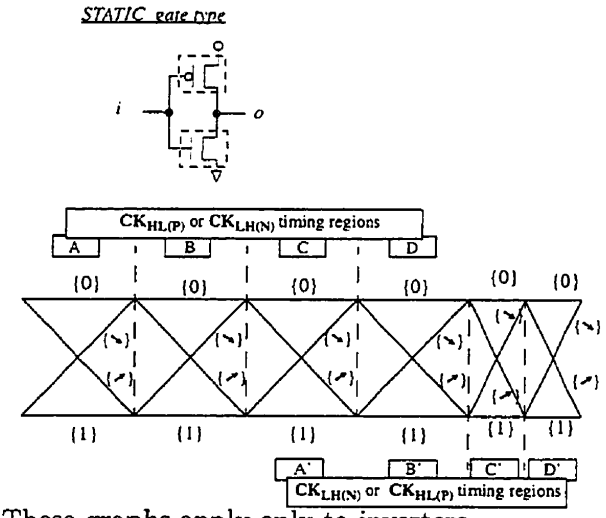



- These graphs apply only to inverters.
- The nodes and edges indicate the shape of the **output waveform**.
- The labels 0,1,/, and \ indicate the shape of the **input waveform** on the specified region.
- The label "all" gives the specified output regardless of the input.

Figure 3.13: Waveform Response Graph for P and N gates

### 3.3.3 WRG for Static Gates

Static gates have no timing dependency other than the intrinsic timing inherited from the input signals. An unspecified timing is given to the functional behavior. Figure 3.14 gives the waveform response graph for a Static gate.

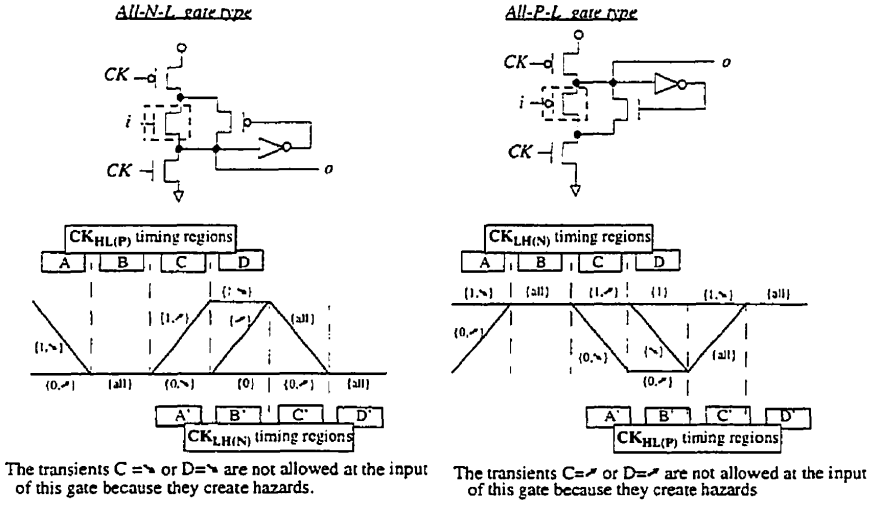



- These graphs apply only to inverters.
- The nodes and edges indicate the shape of the **output waveform**.
- The labels 0,1,↗, and ↘ indicate the shape of the **input waveform** on the specified region.

Figure 3.14: Waveform Response Graph for Static gates

### 3.3.4 WRGs' for All-N-l and All-P-l Gates

The WRGs for the All–N–L and All–P–L dynamic gates in Figure 3.15 have the same descriptions and constraints as those from the N and P dynamic gates. The reason is because these gates are also precharged.




The transients C =↘ or D=↘ are not allowed at the input of this gate because they create hazards.

The transients C=↗ or D=↗ are not allowed at the input of this gate because they create hazards

- These graphs apply only to buffers.
- The nodes and edges indicate the shape of the **output waveform.**
- The labels 0,1,↗, and ↘ indicate the shape of the **input waveform** on the specified region.
- The label "all" gives the specified output regardless of the input.

Figure 3.15: Waveform Response Graph for All–P–L and All–N–L gates

### 3.3.5 Waveform Behaviors

The set of waveforms which occur at a node is named a *waveform behavior*, and the behaviors found at the input and output nodes of a gate are named the *input waveform behavior* and the *output waveform behavior* respectively.

**Example 7** The example in Figure 3.16 shows an N–type dynamic gate which is controlled under a CKLH(N) timing. The number of waveforms in the *input waveform behavior* is assumed to have five waveforms for illustration purposes.

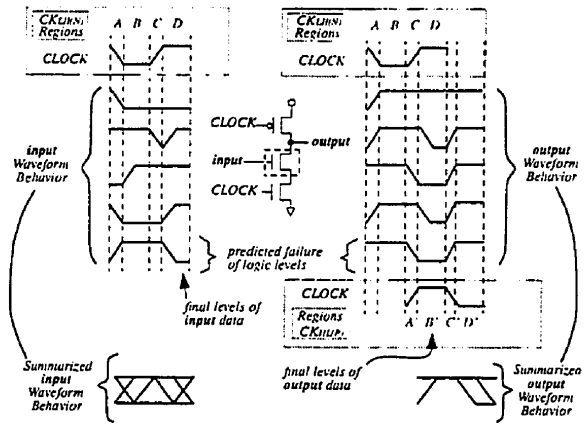



Figure 3.16: Example using Waveform Behaviors

The behavior summaries, introduced in this example, are the result of overlapping all possible waveforms from a behavior. These summaries give a visual aid to identify what kind of transitions are encountered on each of the timing regions.

This example also provides an interesting detail to observe. A failure is generated by the input signal when *input*=/11\. The negative transition of the input signal during the D region shouldn't be used because a dynamic N type gate cannot generate a positive output transition during the evaluate state. This single waveform is hazardous and renders the entire *input*

*waveform behavior*, or the circuit which generated such waveforms, as incompatible with an N type dynamic gate.

## 3.4 Operation of Gates and Latches

The transient symbols (0, 1, / and \) and the regions of the timing framework are used in this section to identify when a gate is working correctly.

The conditions outlined for this analysis are applied over a *single input gate (such as a buffer or an inverter)* and are extended into the analysis for complex gates in Section 3.5.

Section 3.4.1 defines that the correct operation of a gate is determined by the following statements:

- *Logic Transfer* A definition stating that a buffer should be able to propagate signals from the input to the output node.
- *Output Update* A definition stating that the output node of a buffer should not be stuck at any level.
- *Gate Operation* A principle which identifies properly working gate by means of the previous two definitions.

Section 3.4.2 defines that the correct operation of a latch is determined by the following statements:

- *State Hold* A definition stating that a gate is holding a state if the output level is maintained during the precharge period.
- *Latch Operation* A principle which identifies a latch whenever a properly working gate meets the conditions outlined by the State Hold definition.

The waveforms obtained by the WRG's can be crossed into an opposite timing if it is required. Section 3.4.3 defines that the condition which dictates when to perform such crossing is determined by the following statement:

- *Border Conditions* A definition stating that there is a border between two circuits, which are connected, if one circuit is precharging while the other circuit is in the evaluate state. Section 3.3.

### 3.4.1 Gate Operation, Buffers and Inverters

#### Logic Transfer

A logic transfer is the relationship between logic levels at the input and output nodes of a gate. For example, given a logic circuit with two gates, if gate A supplies a logic signal to gate B then gate B performs a logic transfer if the waveforms supplied by gate A cause in gate B a set of waveforms which meet the conditions identifying such transfer.

The following definition states what a logic transfer is:

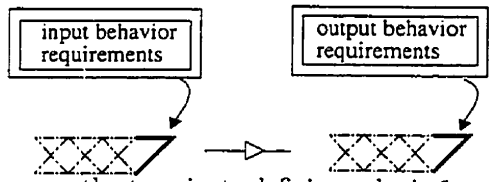
**Definition 3.1 (Logic Transfer)** *Given two nodes, either in a buffer or an inverter, which are named the input node* I *and the output node* O. *There is an effective logic transfer, of binary information, between the input and output nodes of this gate if the relationship between the input and the generated output always maintain* $I \neq O$ *for inverting gates (Static,* N, P, N-C$^2$MOS *and* P-C$^2$MOS*), and* $I = O$ *for non-inverting gates (All-*N-L *and All-*P-L*).*

Expressed on a scheme relative to the ABCD fields of the clock signal: The *final* logic level is determined by region D, therefore the transient occurring at region D of the input node is *I(D)*, and the transient occurring at region D of the output node is *O(D)*. A non-inverting gate has a correct logic transfer if the following statements are true:

- A VH level at the input node generates a VH at the output node if $I(D) \in \{1, /\}$ and $O(D) \in \{1, /\}$ as illustrated in Figure 3.17.
- A VL level at the input node generates a VL at the output node if $I(D) \in \{0, \backslash\}$ and $O(D) \in \{0, \backslash\}$ as illustrated in Figure 3.18

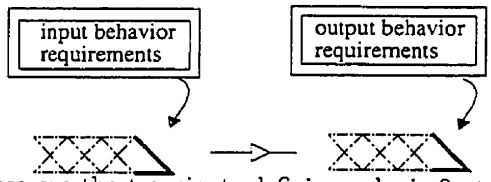
Similarly an inverting gate has a correct logic transfer if the following statements are true:

- A VH level at the input node generates a VL at the output node if $I(D) \in \{1, /\}$ and $O(D) \in \{0, \backslash\}$ as illustrated in Figure 3.19.
- A VL level at the input node generates a VH at the output node if $I(D) \in \{0, \backslash\}$ and $O(D) \in \{1, /\}$ as illustrated in Figure 3.20.



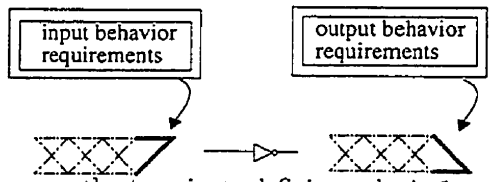

These are the transients defining a logic 1 at the input node of a non-inverting gate and the generated transients which define a logic 1 at the output node.

Figure 3.17: $V_H$ to $V_H$ logic transfer



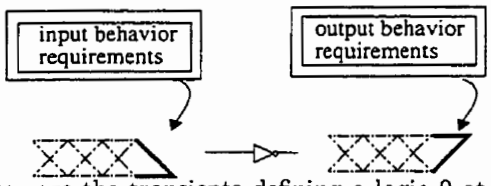

These are the transients defining a logic 0 at the input node of a non-inverting gate and the generated transients which define a logic 0 at the output node.

Figure 3.18: $V_L$ to $V_L$ logic transfer




These are the transients defining a logic 1 at the input node of a non-inverting gate and the generated transients which define a logic 0 at the output node.

Figure 3.19: $V_H$ to $V_L$ logic transfer




These are the transients defining a logic 0 at the input node of a non-inverting gate and the generated transients which define a logic 1 at the output node.

Figure 3.20: VL to VH logic transfer

### Output Update

A gate is also required to change, or update, the state of the output node, showing its capability to change the final data from one logic state to another.

**Definition 3.2 (Output Update)** *The initial and final output levels, during a single clock cycle, define the updated output. A gate is able to perform an update if the generated output behavior has at least one waveform from each of the following sets:*

- *an initial* VL *level to a final* VL *level,*
- *an initial* VL *level to a final* VH *level,*
- *an initial* VH *level to a final* VL *level, and*
- *an initial* VH *level to a final* VH *level.*

Using the clock timing scheme relative to the ABCD fields. Four sets are defined, one for each transient type:

- The waveforms which define a VL to VL update are summarized in Figure 3.21 and are described in the set $L$ = { 0000, 0/\0, 00/\, 0/1\, /\00, /1\0, /\/\, /11\},
- a VH to VH update is summarized in Figure 3.22 and described by the set $H$ = { 1111, 1\/1, 1\0/, 11\/, \/11, \0/1, \00/, \/\/},

- a $V_L$ to $V_H$ update is summarized in Figure 3.23 and described by the set $R = \{$ 0/11, 00/1, 000/, 0/\/, /111, /\/1, /1\/, /\0/$\}$, and
- a $V_H$ to $V_L$ update is summarized in Figure 3.24 and described by the set $F = \{$ 1\00, 11\0, 1\/\, 111\, \000, \/\0, \0/\, \/1\$\}$.

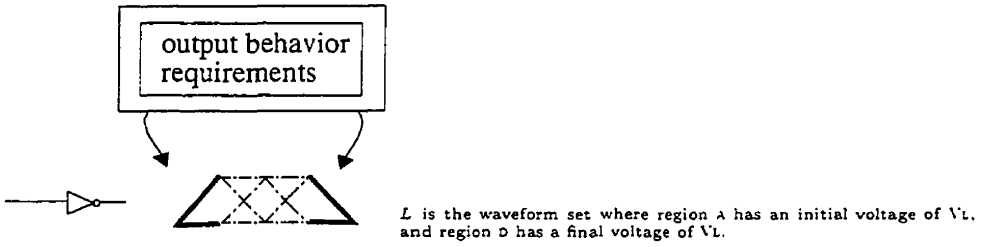



Figure 3.21: Output behavior requirements defining the $L$ set

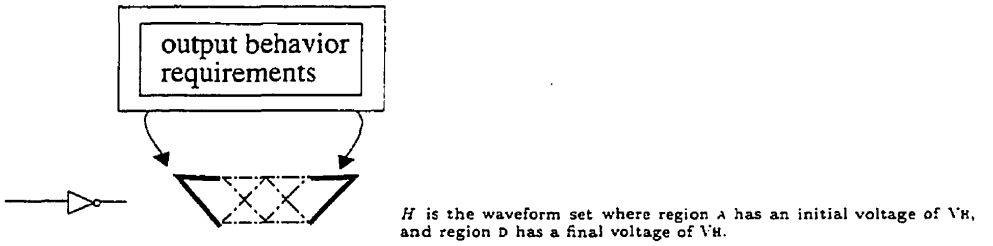



Figure 3.22: Output behavior requirements defining the $H$ set

### Principle of Gate Operation

The error free operation of a gate is guaranteed if the waveforms at the output and input nodes conform with the following principle:

**Principle 3.1 (Gate Operation)** *A gate has a safe mode of operation if it is able to perform a logic transfer, from the input to the output node, and is capable of performing all four types of updates at the output node.*

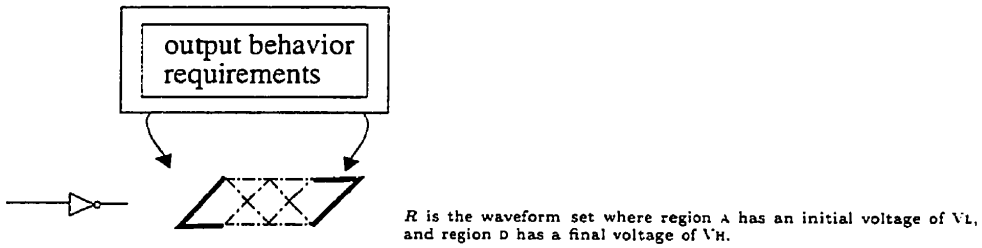



Figure 3.23: Output behavior requirements defining the $R$ set

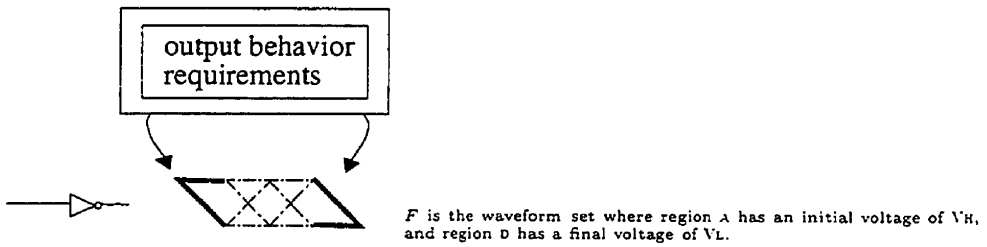



Figure 3.24: Output behavior requirements defining the $F$ set

This principle implies that any gate which is capable of satisfying the conditions set by Definition 3.1 and Definition 3.2 is compatible with the structure that generated the waveform behavior at the input node.

Another characteristic defining a gate behavior is the exclusion of any behavior defined for latch operation, because the conditions which dictate an exchange of data, see the *border conditions* ahead, are a property exclusive to latches.

### 3.4.2 Latch Operation

A latch is the output-gate for a given structure which is able to hold the previous evaluation for the length of the precharge period. This holding of logic levels is a characteristic of latches and it allows one to share information with other structures under an opposite timing. This is true as long as their interconnection meets Principle 3.1 (Gate operation).

The following principle defines the required characteristics for a gate to be considered as a latch:

**Principle 3.2 (Latch Operation)** *Given two gates X and Y, as defined by Principle 3.1; where X feeds Y. Gate Y is said to work as a latch if it meets the characteristics outlined in the* state hold *definition regardless of the output in X during precharge.*

**Definition 3.3 (State hold)** *State holding is only true if a* VL *or a* VH *output level is the result from the previous clock cycle (end of region* D*) and is maintained during the entire precharge period (regions* A *and* B*).*

State holding involves maintaining a certain level during the period of time delimited by regions A and B.

Holding a VL state is defined at the output $O$ along regions A and B by:

- $O(A) \in \{0\}$, and
- $O(B) \in \{0\}$

Expanding this definition, all the possibilities are illustrated in Figure 3.25 and are described by the set: $O = \{$ 0000, 000/, 00/\, 00/1 $\}$.

Holding a VH state is defined at the output $O$ along regions A and B by:

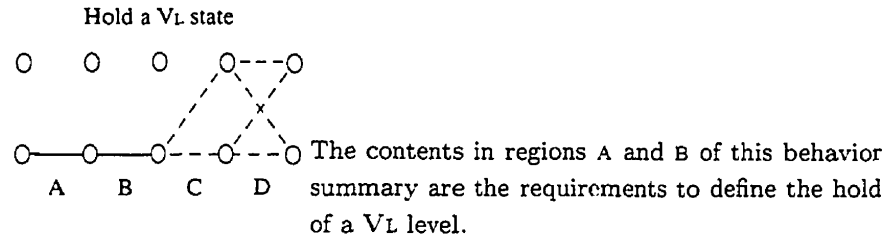



Figure 3.25: Requirements to a hold a VL level.

- $O(A) \in \{1\}$, and
- $O(B) \in \{1\}$

Expanding this definition, all the possibilities are illustrated in Figure 3.26 and are described by the set: $O = \{$ 1111, 111\, 11\/, 11\0 $\}$.

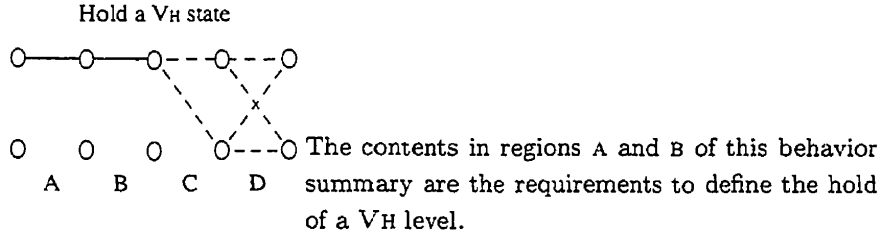



Figure 3.26: Requirements to hold a VH level.

### 3.4.3 Change of Timing

A set of gates operating under the same timing and along a continuos logic path are a type of subcircuit structure that is named a *pipe*. There are two types of pipes considered here: the CKHL(P) and the CKLH(N), which correspond to the timing types defined by the clock signal.

The characteristics defining the boundaries of a pipe are named the *border conditions* and are defined as follows:

**Definition 3.4 (Border Conditions)** *Given two gates $X$ and $Y$, where $X$ feeds $Y$. There is a pipe border at the output node of $X$ if one of the following conditions is true:*

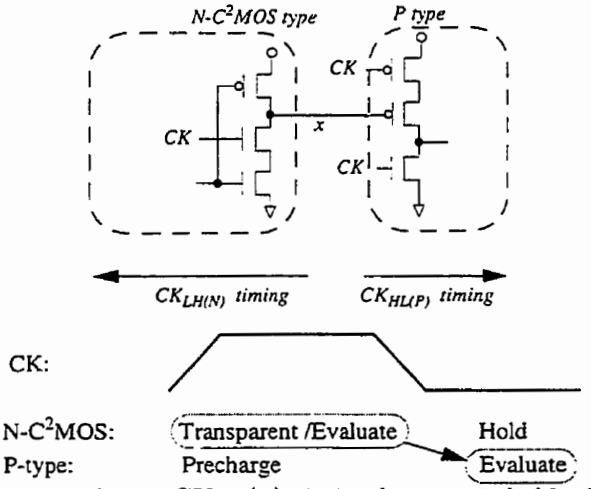
- timing type$(X)$ = timing type$(Y)$ *and the phase of the clock signals is complementary, or*
- timing type$(X) \neq$ timing type$(Y)$ *and the phase of the clock signals is the same.*

Explanation: A *clock* signal is a waveform of binary values and can be supplied in complementary states: *clock* and $\overline{clock}$. If two circuits have the same intrinsic behavior (charge and precharge), relative to their *clock* signals, and have an opposite *clock* phase, then there is in fact an opposite timing of events. The same also applies when both structures use the same *clock* signal but have an opposite timing for their charge and precharge events.

An example of border crossing is found when the N–C$^2$MOS gate of Figure 3.27 is driving a P-type dynamic gate. Both gates are driven by the same clock signal, however the timing of the precharge and evaluate events is delimited by an opposite level in the clock signal. These gates have an opposite operation and node $x$ indicates the boundary where two structures with a different timing meet.

It must be emphasized here that all characteristics identifying gates, latches and borders are used here to describe the output behavior from a structure, rather than describing the characteristics of a single gate.

In the previous example, if the output of the N–C$^2$MOS gate has one or more static inverters in series, then the boundary is not located at the output of the N–C$^2$MOS gate. The boundary is now at the output of the static inverters. The behavior of these static gates is similar to that of the N–C$^2$MOS latch and are part of the CKLH(N) pipe.




An N–C²MOS gate has a CKLH(N) timing because it holds the output with the signal $CK$ = VL. This operation is opposite to that of the P-type gate that has a CKHL(P) timing and precharges with the signal $CK$ = VH. Therefore node $x$ is the boundary between two structures of opposite timing. The data is propagated by the P-type gate with a 180 deg delay.

Figure 3.27: Border crossing example

## 3.5 Complex Gates

Logic gates such as NAND or NOR merge at least two waveform behaviors to make a new one. The procedure for finding the output behavior from a complex gate has three steps:

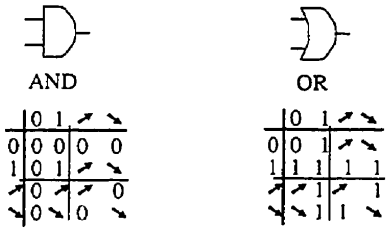
- First, the logic function is separated from the gate structure and *decomposed* into two-input non-inverting gates.
- Second, the input behaviors to this non-inverting complex gates are evaluated into an equivalent waveform behavior.
- Third, the equivalent waveform behavior is again transformed by an inverter (or buffer) of the desired gate type using the corresponding *waveform response graph.*

The *decomposition* step consists of finding an equivalent logic expression of the non-inverting logic function by using the fundamental two-input logic operators AND and OR. Numerous possibilities exist for the more complicated functions [Mic94], however it isn't relevant which version of the same function is chosen because they all generate the same equivalent behavior.

The logic evaluation step evaluates the transients found within the same timing region (A, B, C, or D). This evaluation is performed according to the two input logic functions and the transients found at each input and matched with the tables of Figure 3.28.

The objective of the table in Figure 3.28 is to identify the initial and final state of the output within a single region (A,B,C or D) that is induced by the input transients. The timing model is limited here because it is known that opposite transitions at the input, described by the / and \ symbols, can generate glitches. However this is not a problem because the instance where opposite input transients are present always generates incompatible behaviors to precharged dynamic gates. Therefore the usage of hazardous signals is avoided. This is explained with more detail in Section 3.8.

**Example 8** *The logic gates in Figure 3.29 are given arbitrary input behaviors. These gates are first decomposed into the fundamental non-inverting operators and are followed by an* N*-type inverter. The figure shows that both*




(a) (b)

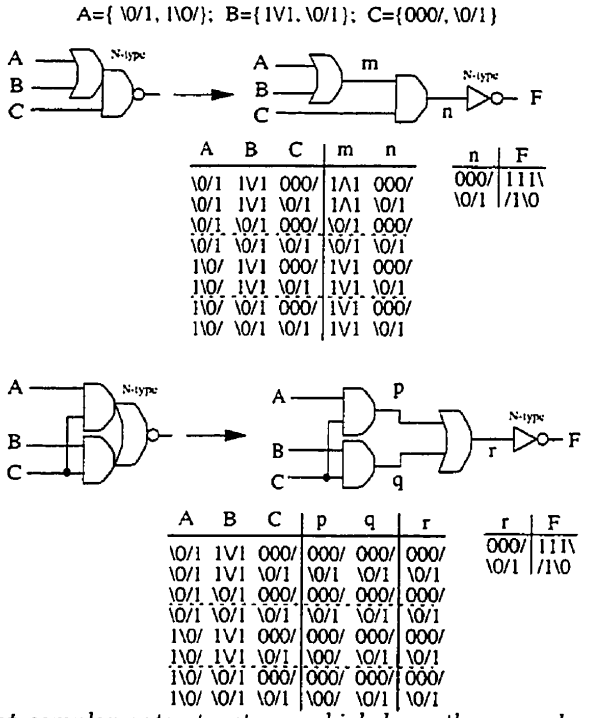
These tables are used to find the output transient in a complex gate induced by the input signals. Complex logic gates are decomposed into two-input AND and OR gates and the inverting function of a gate is analyzed separately. The shaded areas indicate when the output might be affected with glitches.

Figure 3.28: Logic Evaluation of Transients

*gates perform the same logic function and have the same output waveform behavior.*

The resulting waveforms from using the table of Figure 3.28 are valid waveforms since they fully describe the output of a gate within the limits of the timing model (only 0,1, / and \ transients). A better timing framework is presented in Chapter 5 where glitches are considered.

It is emphasized here that the / and \ transients defined by the timing framework represent the changes between discrete levels ($V_H$ and $V_L$) at the beginning and end of a region. The timing framework doesn't considers the glitches that could arise by differentials in the arrival times between inputs within a single region, however it is demonstrated in Section 3.8 that such conditions are avoided whenever susceptible dynamic logic gates might be affected.




| A | B | C | m | n |
|---|---|---|---|---|
| \0/1 | 1V1 | 000/ | 1∧1 | 000/ |
| \0/1 | 1V1 | \0/1 | 1∧1 | \0/1 |
| \0/1 | \0/1 | 000/ | \0/1 | 000/ |
| \0/1 | \0/1 | \0/1 | \0/1 | \0/1 |
| 1\0/ | 1V1 | 000/ | 1V1 | 000/ |
| 1\0/ | 1V1 | \0/1 | 1V1 | \0/1 |
| 1\0/ | \0/1 | 000/ | 1V1 | 000/ |
| 1\0/ | \0/1 | \0/1 | 1V1 | \0/1 |

| n | F |
|---|---|
| 000/ | 111\ |
| \0/1 | /1\0 |

| A | B | C | p | q | r |
|---|---|---|---|---|---|
| \0/1 | 1V1 | 000/ | 000/ | 000/ | 000/ |
| \0/1 | 1V1 | \0/1 | \0/1 | \0/1 | \0/1 |
| \0/1 | \0/1 | 000/ | 000/ | 000/ | 000/ |
| \0/1 | \0/1 | \0/1 | \0/1 | \0/1 | \0/1 |
| 1\0/ | 1V1 | 000/ | 000/ | 000/ | 000/ |
| 1\0/ | 1V1 | \0/1 | \00/ | \0/1 | \0/1 |
| 1\0/ | \0/1 | 000/ | 000/ | 000/ | 000/ |
| 1\0/ | \0/1 | \0/1 | \00/ | \0/1 | \0/1 |

| r | F |
|---|---|
| 000/ | 111\ |
| \0/1 | /1\0 |

*Two different complex gate structures which have the same logic function, $\overline{(A+B)C}$ and $\overline{(AC+BC)}$, will produce the same equivalent behavior.*

*Figure 3.29: Waveform behaviors transformed by complex gates.*

## 3.6 Waveform Behaviors and Compatible Gates

The interconnection rules for the new design technique are found in this section. These rules are described as a group of different *waveform behaviors* interconnected by gates. This method allows to analyze the correctness of a circuit based on its structure and the type of input signals.

### 3.6.1 Searching for Waveform Behaviors

The algorithm which finds the set with all possible *waveform behaviors* is simply a **do ... until loop**. This set of waveform behaviors is identified with the letter $R$

The algorithm starts with one waveform behavior $R = \{W_0\}$. This behavior ($W_0$) is the largest set of waveforms defined by the timing framework, and it has 32 waveform shapes.

Next is the **do ... until** loop:

- Each behavior in $R$ is tested against all gate types of buffers and inverters to determine if their interconnection is feasible. The concepts in the previous sections are applied to determine if the relationship between the input and the generated output behavior indicate a properly working logic gate. If it is, then the output behaviors are stored in $X$.
- The same waveform behaviors in $R$ are used to find new equivalent behaviors when they drive AND and OR gates. The output behaviors are stored in $Y$.
- Determine if there were any new behaviors: $(X \cup Y) \cap R \neq \emptyset$. If there are new behaviors then add them to $R$ and repeat the loop ($R = R \cup X \cup Y$).

#### Rejection of Waveform Behaviors

Numerous waveform behaviors are found by the search loop and some of them are rejected. The rejection criteria is based not only on the principles of Section 3.4.1 and Section 3.4.2 but on the additional constraints for dynamic gate operation.

The cases to consider are the N, P, All–N–L and All–P–L gate types. These gates are protected against races by not allowing their waveforms to be used if there is a potentially destructive transition in region C of the timing scheme, see their WRGs in Figure 3.13 and Figure 3.15. With this restriction the input transition is forced to be well defined within the evaluate or precharge regions.

### 3.6.2 Waveform Behavior Categories

The results from the search loop are large and there is no need to show all behaviors, instead, the summaries of the resulting waveform behaviors are classified within the behavior categories illustrated in Figure 3.30. The general description for each category is given next, the behaviors described in Figure 3.30 have added a postfix according to their timing. "-N" for CKLH(N), and "-P" for CKHL(P):

- The *(Out)* category is a waveform behavior which has been *crossed* (Section 3.4.3) from one timing to the opposite. The *(Out-N)* waveform behavior is the output from a circuit under a CKLH(N) timing and has been crossed to a CKHL(P) timing. Similarly, the *(Out-P)* waveform behavior is the output of a circuit under a CKHL(P) timing that has been crossed to a CKLH(N) timing.

- The *Out* category is a behavior which is characteristic of latches. This is the only behavior that can be crossed because it maintains a *steady* voltage level during regions A and B.

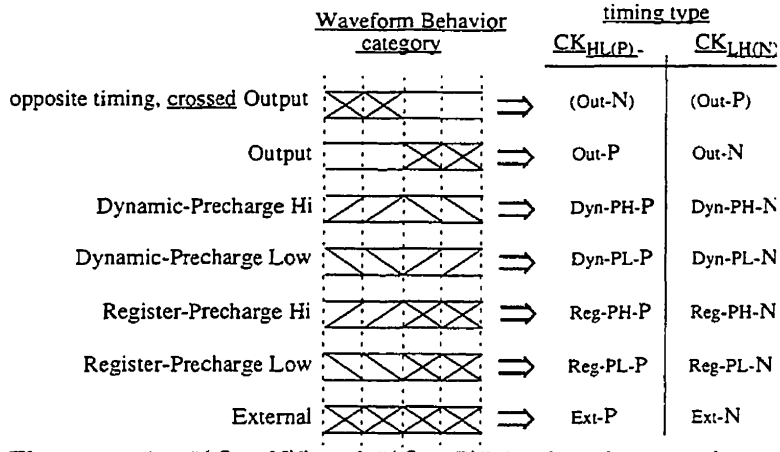
- The *Dyn-PH* category identifies the behavior whose waveforms are able to drive dynamic gates that aren't affected by a falling transient (↘) during the evaluate state. The waveforms in this behavior category are restricted to only have *rising* transitions during precharge and *falling* transitions during the evaluate state.

- The *Dyn-PL* category has the opposite characteristics of the *Dyn-PH* category. The waveforms in this category are restricted to only have *falling* transitions during precharge and *rising* transitions during the evaluate state.

- The *Reg-PH* category has the characteristic waveforms in the first stage of a latch. This category is able to register only *rising* transitions during the precharge state, and is able to perform all types of transitions during the evaluate state (/ and \). This behavior is unable to drive precharged dynamic gates, but it isn't a problem for Static, N–C$^2$MOS and P–C$^2$MOS gates.

- The *Reg-PL* category has the opposite characteristics of the *Reg-PH* category. This behavior category is able to register only *falling* transitions during the precharge state, and is able to perform all types of transitions during the evaluate state (/ and \).

- The *External* category identifies either a waveform behavior which has all 32 waveform shapes, or performs both types of transitions during the precharge and evaluate states, or it is simply unknown. An example of an unknown behavior is that from a circuit fully built in static logic which has no interface for driving dynamic logic circuits.




The categories "(*Out-N*)" and "(*Out-P*)" implies that waveforms in the "*Out-N*" and "*Out-P*" behaviors are *crossed*(Section 3.4.3) from one timing to the opposite.

Figure 3.30: Waveform Behavior Categories.

### 3.6.3 Interconnection Rules for Buffers and Inverters

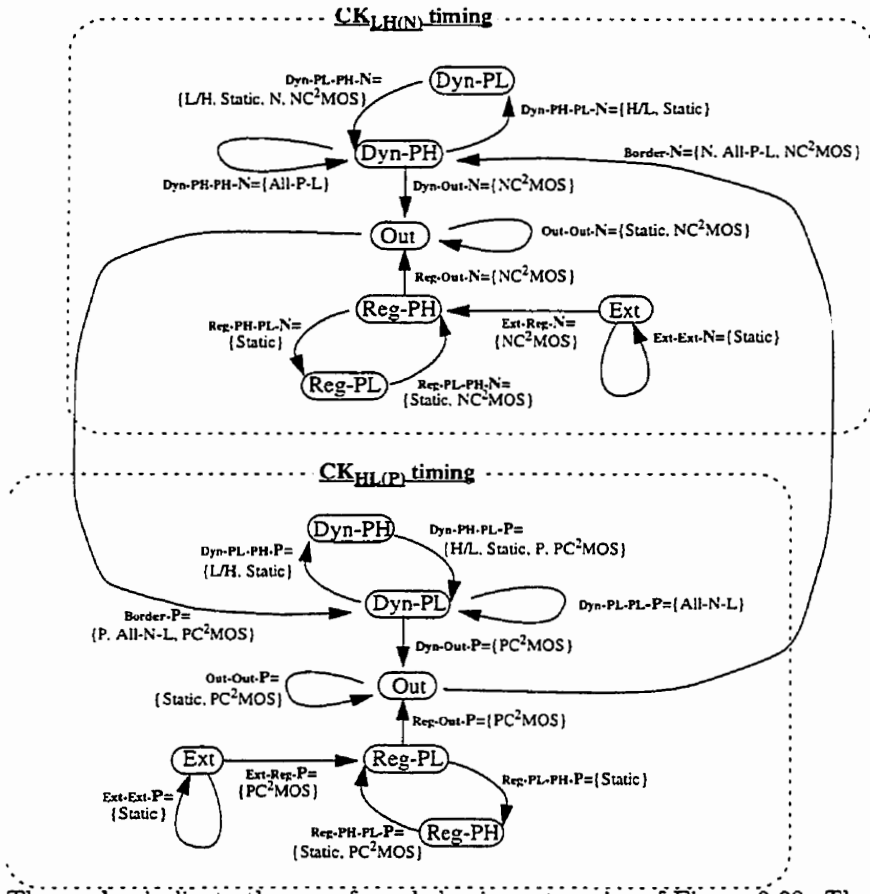
The circuit design rules for buffers and inverters of the new circuit design technique are described in Figure 3.31 with a network of nodes and unidirectional edges. The nodes correspond to particular waveform behavior categories, and the edges indicate which gate-types will have such input to output relationship. The input behavior is the behavior at the tail of an edge and the output behavior is pointed to by the head.

An alternative description of the interconnection rules is the table in Figure 3.32. This table is derived from Figure 3.31 and it describes the same interconnection rules. The horizontal axis corresponds to the input behavior, the vertical axis is the gate type, and the contents of the table are the output behaviors.

### 3.6.4 Equivalent Behaviors for Complex Gates

The search loop tests for new behaviors by evaluating the current set of waveform behaviors into the AND and OR logic operators. The information from these tests is kept to find a table of equivalent behaviors for the AND and OR logic operators.

The analysis for complex gates follows the same procedure as for individual waveforms. The logic function is first decomposed into two-input AND and OR gates, and are followed by an inverter of the desired gate type. The matrix in Figure 3.33 shows which is the equivalent waveform behavior category given that of the two inputs.




The **nodes** indicate the waveform behavior categories of Figure 3.30. The edge **labels** indicate which gates types meet the input to output relationship pointed out by the unidirectional edges.

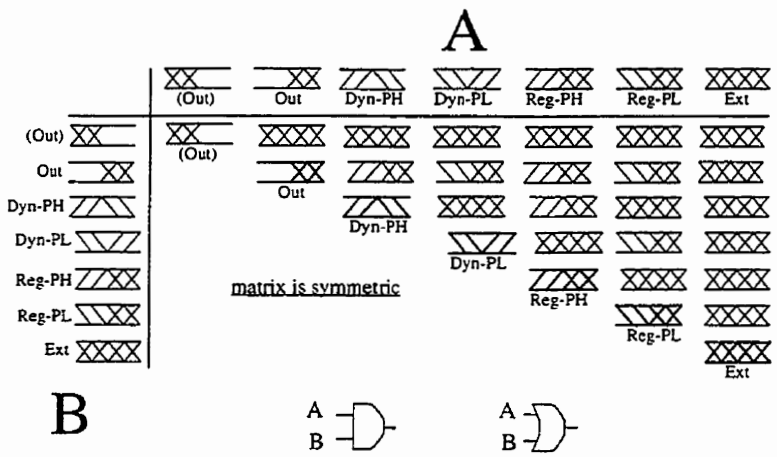
Figure 3.31: Interconnection Rules for Dynamic Logic

| | (Out-P) | Out-N | Dyn-PH- N | Dyn-PL-N | Reg-PH-N | Reg-PL-N | Ext-N |
|---|---|---|---|---|---|---|---|
| N | Dyn-PH-N | | | Dyn-PH-N | | | |
| P | | **Dyn-PL-P** | | | | | |
| N-C$^2$MOS | Dyn-PH-N | Out-N | Out-N | Dyn-PH-N | Out-N | Reg-PH-N | Reg-PH-N |
| P-C$^2$MOS | | **Dyn-PL-P** | | | | | |
| Static | | Out-N | Dyn-PL-N | Dyn-PH-N | Reg-PL-N | Reg-PH-N | Ext-N |
| All-N-L | | **Dyn-PL-P** | | | | | |
| All-P-L | Dyn-PH-N | | Dyn-PH-N | | | | |
| H/L | | | Dyn-PL-N | | | | |
| L/H | | | | Dyn-PH-N | | | |

| | (Out-N) | Out-P | Dyn-PH-P | Dyn-PL-P | Reg-PH-P | Reg-PL-P | Ext-P |
|---|---|---|---|---|---|---|---|
| N | | **Dyn-PH-N** | | | | | |
| P | Dyn-PL-P | | Dyn-PL-P | | | | |
| N-C$^2$MOS | | **Dyn-PH-N** | | | | | |
| P-C$^2$MOS | Dyn-PL-P | Out-P | Dyn-PL-P | Out-P | Reg-PL-P | Out-P | Reg-PL-P |
| Static | | Out-P | Dyn-PL-P | Dyn-PH-P | Reg-PL-P | Reg-PH-P | Ext-P |
| All-N-L | Dyn-PL-P | | | Dyn-PL-P | | | |
| All-P-L | | **Dyn-PH-N** | | | | | |
| H/L | | | Dyn-PL-P | | | | |
| L/H | | | | Dyn-PH-P | | | |

The horizontal axis is the input behavior, the vertical axis is the gate type. The contents of this table are the output behaviors and those left blank indicate infeasible connections.
The behaviors in the "(*Out-N*)" and "(*Out-P*)" columns are the same as those in bold face in the "*Out-N*" and "*Out-P*" columns respectively.
(The gate types H/L and L/H are the optimized gates of Section 2.5.6)

Figure 3.32: Output Behavior Tables for Dynamic Logic Gates




The equivalent behavior is found by matching the behavior categories at the input of an AND or OR gate with this table. The contents are symmetric, and as expected if both inputs show the same behavior the output will be the same too.

Figure 3.33: Equivalent Behavior Categories for AND and OR operators

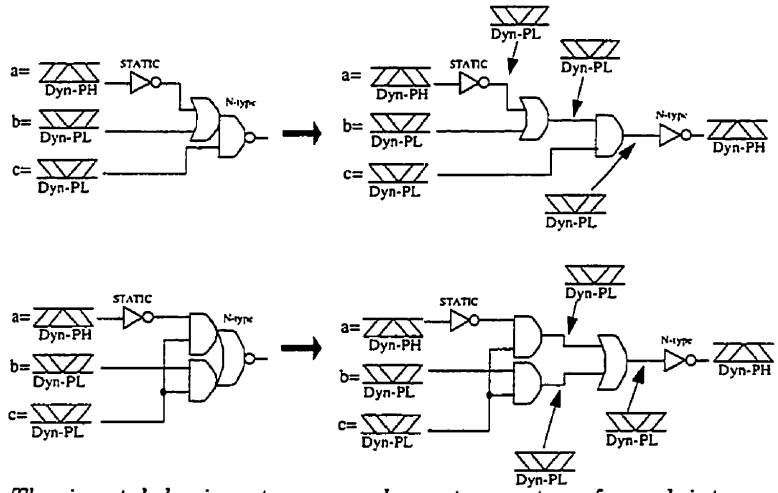
**Example 9** *The example in Figure 3.34 demonstrates how to use the table in Figure 3.33 for finding the output behavior of the following complex gates:*

$$\overline{(\bar{a}+b)c}$$

*and*

$$\overline{(\bar{a}c+bc)}$$

*The behaviors for the input signals are chosen arbitrarily and the paths followed from input to output are all valid paths according to the rules network of Figure 3.31 and the equivalents of Figure 3.33.*




*The input behaviors to a complex gate are transformed into an equivalent behavior. This equivalent behavior is obtained after using the table in Figure 3.33 into every two-input logic operator. Finally, the resulting equivalent behavior is transformed by the gate with the rules network described by Figure 3.31.*

*Figure 3.34: Example using the Behavior tables for Complex Gates*

## 3.7 Assumptions Before Analysis

Larger circuits such as microprocessors utilize different circuit techniques for their many sections, and not all of them can have its logic gates designed in dynamic logic. A few assumptions are listed here to help classify the incoming signals from non-dynamic logic circuits into the rules and behavior categories of Figure 3.31,

- Incoming waveforms from an external non-dynamic logic circuit are assumed to be synchronized with the clock signal and their shape is classified by the same timing model of Section 3.1.2,
- The unknown waveform behaviors, expected from external circuits, are given the behavior category of *Ext-N* whenever these behaviors are driving a CKLH(N) pipe, or a category of *Ext-P* when driving CKHL(P) pipes.
- Static Edge-triggered latches can be replaced with their dynamic counterparts, either for the purpose of analysis or to effect the latch function itself.

The last item can be generalized for pipes. Inside a CKLH(N) pipe there are gate types which are predominantly implemented with NMOS devices and their VL to VH transient in the clock signal shows a similar behavior to a *positive edge-triggered latch.* This similarity is because a CKLH(N) pipe propagates the input data for only half the period of the clock cycle, starting from the rising edge and as long as it stays at the VH level, to continue the propagation of data for the duration of the following VL level it's required to use a pipe under a CKHL(P) timing for the output.

The same is true for *negative edge-triggered latches* which can be replaced by a pipe operating under a CKHL(P) timing followed by a CKLH(N) pipe.

## 3.8 Solved Problems

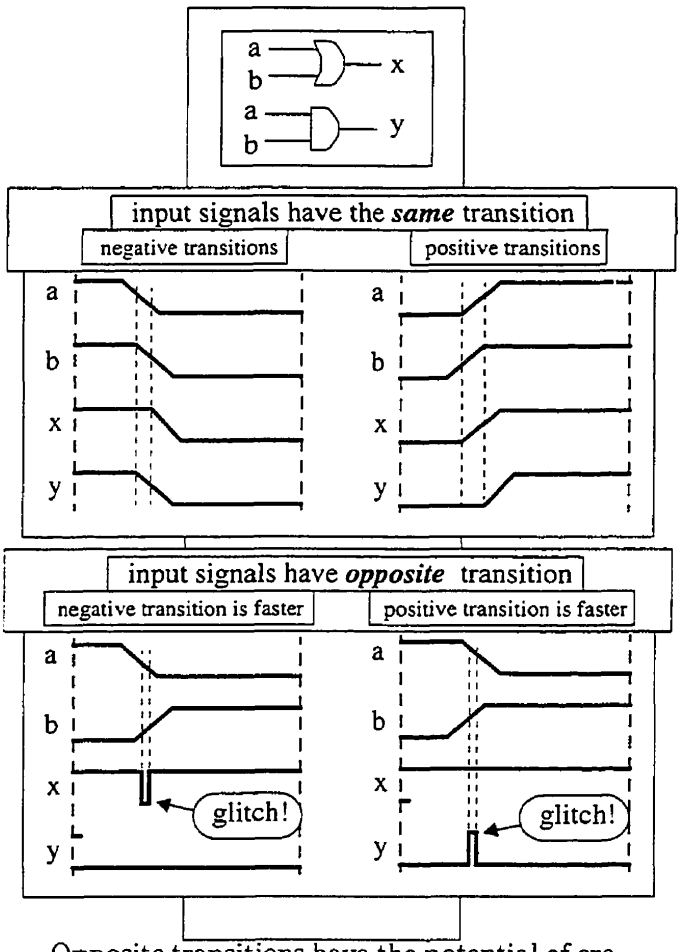
This section explains how the new technique is able to protect a dynamic logic circuit against the following common problems:

- Opposite-Edge input Transients present problems such as glitches. This type of problem is solved by the new technique.
- Loss of Precharged Levels is a problem due to races between clock and data. This problem was first solved with the Domino technique and is also solved with the new technique.
- Charge Feedthrough is a problem found when mixing complementary dynamic gates, N and P, in the same pipe. Therefore, opposite evaluate transitions are indirectly coupled and may induce errors. Because the new technique is operated under a single phase it avoids this type of problems.

Another limitation of dynamic logic circuits are the structural requirements within pipes. Odd/Even constraints are structural limitations found in techniques such as NORA and TSPC. The new technique is still bound by the fundamental constraints from these limitations, but it efficiently delivers such requirements in a single graph that can be implemented by a synthesis tool. Chapter 6 contains two examples where the concepts of this chapter are used to manually transform a circuit from a fully static version to a mixture with dynamic logic.

### 3.8.1 Opposite-Edge Input Transients

The arrival of transitions in two or more input signals, in a complex gate, do not always occur at the same time. These time differentials are prone to create intermediate evaluations, or glitches, which can be disastrous to a precharged dynamic gate. The waveforms in Figure 3.35 show that if these transitions have the same direction, there will be no intermediate evaluations or glitches. However if these waveforms perform opposite transitions then the output is prone to create hazardous glitches to a precharged dynamic gate.




Opposite transitions have the potential of creating glitches, whereas transitions with the same direction have no glitches.

Figure 3.35: Logic bound problems

The gate types which are susceptible to glitches are the N, P, All–N–L and All–P–L gates because they all have non–recoverable precharged voltages. The new technique is able protect these gates against glitches because having two inputs with opposite transitions involves using at least one that is not permitted by the new technique.

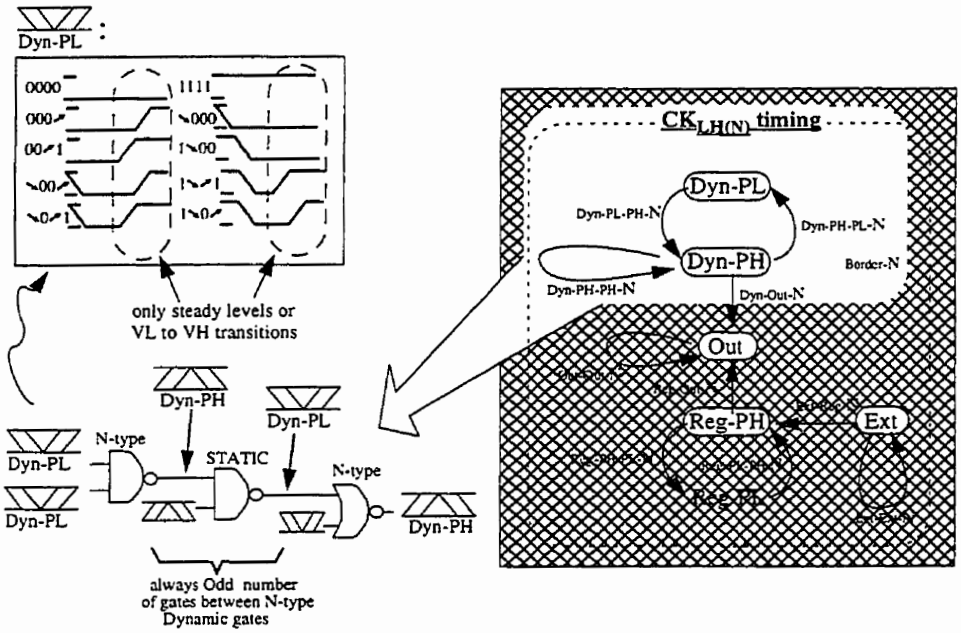
The proof to this statement is that the waveform behavior of a logic gate, whose input signals have opposite transitions, will generate an equivalent behavior that is not compatible with any of the precharged dynamic gates.

### 3.8.2 Loss of Precharged Levels

The Domino logic technique was created to prevent the loss of precharged levels between interconnected dynamic logic gates. The new technique includes all the components of a Domino logic gate and is able to arrange these gates in a similar fashion.

If a circuit has an N-type gate, whose output behavior is always described by the *Dyn-PH-N* behavior, then subsequent N-type gates will only be driven by the *Dyn-PL-N* behavior. The *Dyn-PL-N* behavior is restricted to have the waveforms where only the VL to VH transitions exist in the evaluate state as illustrated in Figure 3.36.

In general, the behaviors *Dyn-PH, Dyn-PL* and *(Out)* are the only behaviors driving precharged dynamic logic gates. The interconnection of precharged dynamic gates is specified by the network graph in Figure 3.31 and it guarantees their proper placement within a circuit. The table of equivalent behaviors for logic operators also helps to maintain this integrity by assigning an incompatible equivalent whenever different behaviors are present at the input.




N-type gates are protected from destructive $V_H$ to $V_L$ transitions

No glitches are generated at the input of a dynamic gate because only the behaviors *Dyn-PL-N* and *(Out-P)* can drive an N-type dynamic gate

A Similar mechanism protects P, All-N-L and All-P-L gate types

Figure 3.36: N-type gate example for protection against destructive transitions

# Chapter 4

# Examples using the Circuit Technique

This chapter is a tutorial for the new technique. The examples in this chapter are simple and meaningful enough to demonstrate how to interconnect dynamic gates as well as to predict their output response.

The first section has three sample circuits from where their output response is predicted. The first two are a chain of buffers and a Domino AND gate, these circuits are analyzed by using the *waveform response graphs* (WRG's). The third example is the same Domino AND gate and it is analyzed by using the behavior categories of the new technique. The second section has two examples showing how to determine the compatible gate types and behaviors at the input and output of a circuit.

Finally, the benchmark circuit *cm150a*, which is implemented in dynamic logic, is simulated and compared against the waveform shapes predicted by the *waveform response graphs* (WRG's).

## 4.1 Forecasting the Output Response of Dynamic Circuits

The examples of this section are simple problems where the output response of circuits are found. The input behaviors are given as part of the problem, either as a waveform behavior or as a behavior category.

### 4.1.1 Buffer Example using WRG's

This example consists of four inverters connected in series, and as specified in Figure 4.1a their gate types are P-C$^2$MOS, P-C$^2$MOS, N-C$^2$MOS and N-C$^2$MOS respectively. The input behavior at node $N_1$ is under a CKHL(P) timing and is specified by the set

$$N_1 = \{0000, 0/11, /111, 1111, 1\backslash 00, \backslash 000, \backslash/11, /\backslash 00\}$$

The exercise's objective is to find the output behavior at node $N_5$ by using *waveform response graphs* (WRG's).

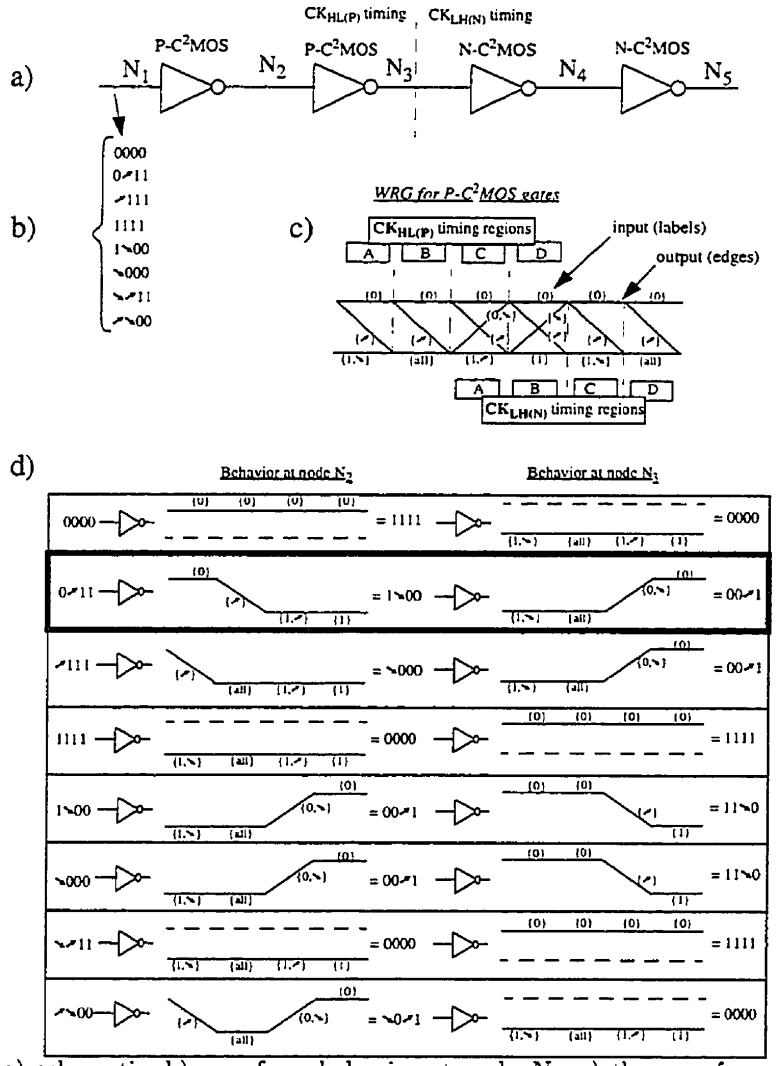
#### Solution

The solution is divided into three steps. The first step finds the behavior at nodes $N_2$ and $N_3$ under a CKHL(P) timing. The second step performs the behavior crossing at node $N_3$ from a CKHL(P) to a CKLH(N) timing, and the third step finds the behavior at nodes $N_4$ and $N_5$

#### Step One

The problem indicates the use of WRG's. The WRG for P-C$^2$MOS gates is illustrated again in Figure 4.1c for convenience. The main components of this graph are the labels and the graph edges. The labels correspond to the transients inside the input waveform (using the symbols 0, 1, / and \) of the specified region, and the edges correspond to the output response of the buffer.

The input behavior at node $N_1$ has, among others, the waveform $N_1 = 0/11$ indicating that




a) schematic, b) waveform behavior at node $N_1$, c) the *waveform response graph* for an P–C²MOS inverter and d) the detailed analysis showing how to obtain the output behavior of two P–C²MOS inverters in series.

Figure 4.1: Analysis of the CKHL(P) section for a chain of buffers

- region A = 0,
- region B = /,
- region C = 1 and
- region D = 1

Which are under a CKHL(P) timing. The output of the P–C$^2$MOS gate is then indicated by the edges next to the labels containing these symbols as illustrated in Figure 4.1d. The edges are translated into symbols, which for the input waveform $N_1 = 0/11$ the output response is then $N_2 = 1\backslash 00$.

Similarly for the next P–C$^2$MOS gate, the input waveform $N_2 = 1\backslash 00$ indicates that region A = 1, B = \, etc. These symbols are matched to the labels within the corresponding region to indicate the output edge. The edges are translated into symbols and the output waveform response by the input $N_2 = 1\backslash 00$ is $N_3 = 00/1$.

### Step Two

The timing of the waveform behavior at nodes $N_1$, $N_2$ and $N_3$ is CKHL(P), however the N–C$^2$MOS buffers operate under a CKLH(N) timing. The communication between two different timings is possible by transforming the waveforms present at node $N_3$ from one timing to another.

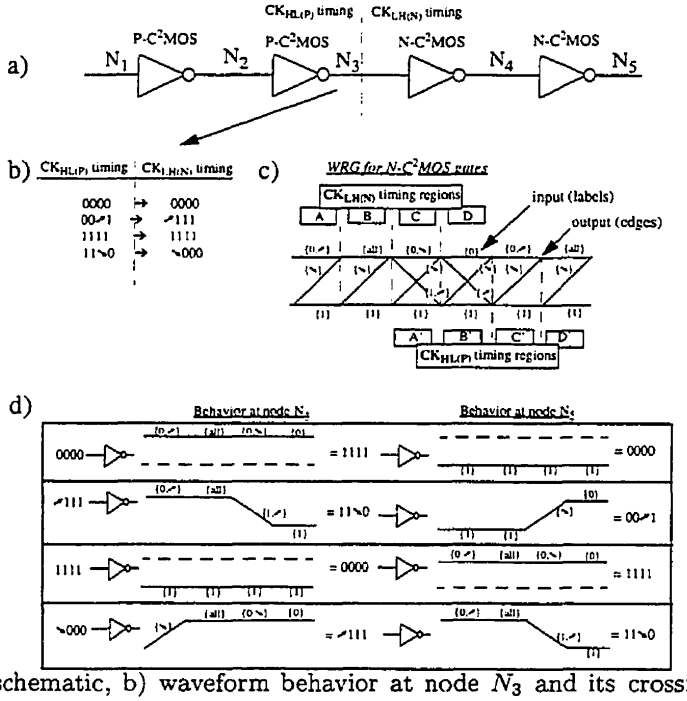
This crossing of timings is performed by copying, for each waveform, the contents of the C and D regions of the CKHL(P) timing into the A and B regions of a waveform under a CKLH(N) timing. There are two possibilities for the C and D regions of the CKLH(N) timing either CD = 00 or CD = 11. These are all the feasible possibilities given by the waveforms $N_3$. Therefore if $N_3 = 00/1$ under a CKHL(P) timing then this waveform is viewed as $N_3 = /111$ under a CKLH(N) timing.

The crossing from one timing to the other is illustrated in Figure 4.2b. In general, if the rules of the new technique are followed then the crossing of timings is as simple as the example in this section. The reason is because the OUT behavior is the only one which will be crossed, and it provides the symbols for regions C and D of CD = 00 and CD = 11.

### Step Three

The last step propagates the behavior at node $N_3$ under a CKLH(N) timing thru the N–C$^2$MOS buffers. The appropriate WRG is illustrated again in Figure 4.2c for convenience.

Similarly to the first step each waveform has its symbols matched with the labels in the WRG to determine the edges of the output waveform. These edges are represented by symbols which define a waveform belonging to the behavior at node $N_4$. The behavior at node $N_5$ is found by propagating first all the waveforms from node $N_3$ to $N_4$, and from $N_4$ to $N_5$.



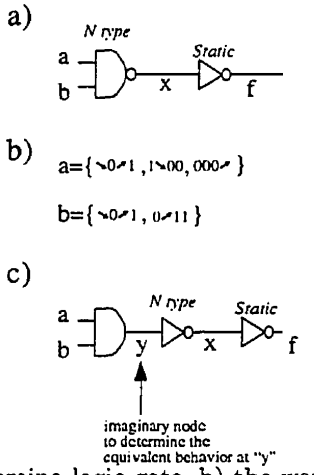

a) schematic, b) waveform behavior at node $N_3$ and its crossing from a CKHL(P) to a CKLH(N) timing, c) the *waveform response graph* for an N–C$^2$MOS inverter and d) the detailed analysis showing how to obtain the output behavior of two N–C$^2$MOS inverters in series.

Figure 4.2: Analysis of the CKLH(N) section for a chain of buffers

Observe that the behavior at node $N_5$ shows fewer transients such as $\nearrow$ or $\searrow$. The reason for this effect is because dynamic latches ignore one kind of transient during its *hold* state and recovers to the correct output level during the *evaluate* state. In general, it is observed that latches act as filters for unwanted transitions at the expense of introducing a delay latency.

### 4.1.2 Domino NAND gate using WRG's

This example consists of a dynamic N type NAND gate that is driving a Static buffer. The input behavior is given in Figure 4.3b and it is defined under a CKLH(N) timing. The objective is of finding the output behavior at node $f$ by using *waveform response graphs* (WRG's).




a) schematic of a Domino logic gate, b) the waveform behavior at nodes $a$ and $b$, and c) the equivalent schematic for the purpose of analysis.

Figure 4.3: Domino AND gate, its input behavior and the gate's equivalent

#### Solution

The solution is divided in three steps. The first step determines an equivalent for the purpose of analysis with the new technique. The second step finds the

test vectors for the AND gate and propagates the behaviors to node $f$. The last step determines the shape of the *waveform behavior summaries* at each node for the purpose of comparing them to the results of the next example.

### Step One

A circuit with logic gates more complex than a buffer has to be decomposed into an equivalent. The equivalent circuit is required to provide a circuit that is compatible with the requirements from the logic tables and with the format of the rules of the new technique. The required format is the following

- Decompose every logic gate into non-inverting, two input logic AND and OR gates.
- The inverting function is separated and evaluated by a buffer of the corresponding gate type.

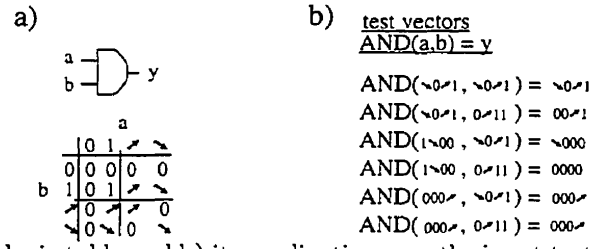
The decomposition of the logic introduces the non-existent imaginary node "$y$", and the AND is given no gate type.

### Step Two

The input behavior at nodes $a$ and $b$ are tested for all possible combinations. Each behavior is a set of waveforms, where only one can occur during a clock cycle, and there is no additional information specifying what waveform in $a$ is restricted to which one in $b$. This independence between inputs is analyzed by testing all $(a, b)$ waveform pairs and creates a total of 6 test vectors. The table which evaluates the AND function using the symbols for transient events is illustrated again in Figure 4.4a.

The evaluation of a test vector $(a, b) = (\backslash 0/1, 0/11)$ starts with its decomposition into the ABCD regions. The table in Figure 4.5a is then used to evaluate each region:

$$\begin{aligned} \text{AND}(\backslash, 0) &= 0 \\ \text{AND}(0, /) &= 0 \\ \text{AND}(/, 1) &= / \\ \text{AND}(1, 1) &= 1 \end{aligned}$$




a) AND logic table and b) its application over the input test vectors for $AND(a, b)$.

Figure 4.4: Equivalent behavior at the *imaginary* node $y$

giving the waveform at node $y = 00/1$ as illustrated in Figure 4.4b.

Similarly to the buffer example, the WRG's for the N and Static type buffers, illustrated in Figure 4.5, are applied to propagate the behavior found from node $y$ to node $x$, and from node $x$ to node $f$.

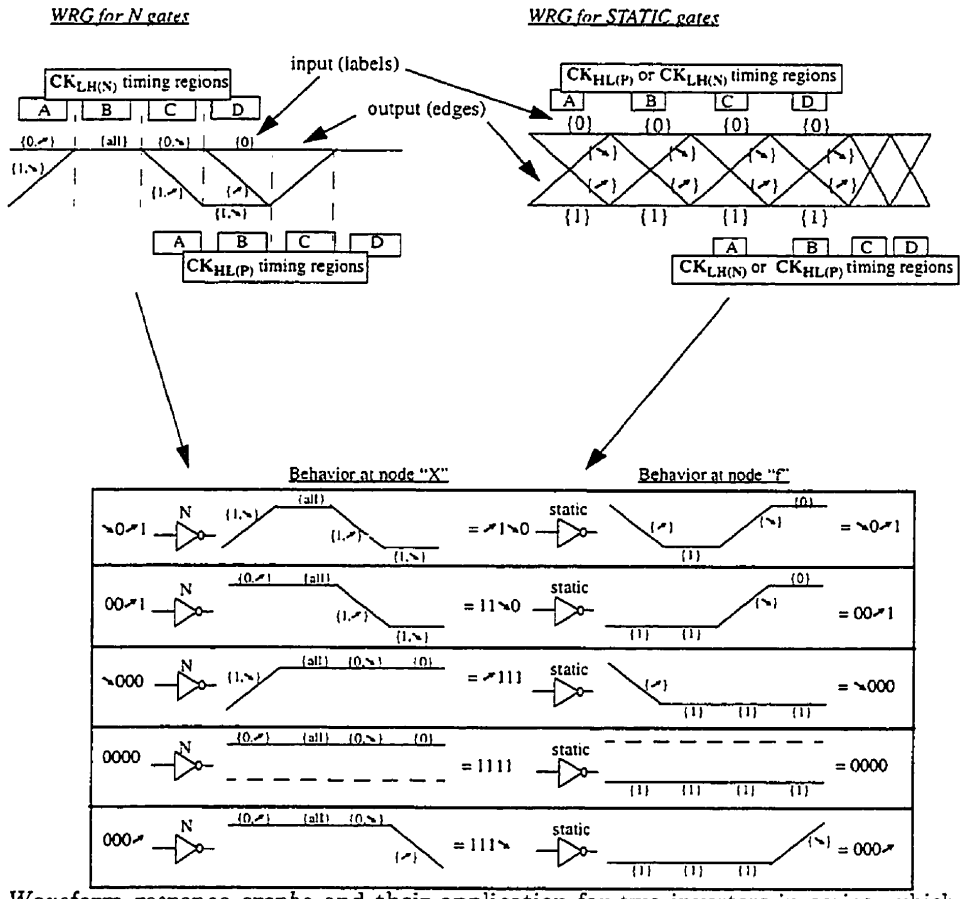
### Step Three

The summary of a waveform behavior is a graphic aid to classify a waveform behavior into the rules of the new technique. The summary is found by the superimposition of all waveforms within a behavior.

Finding the summary of the behavior at node $a = \{\backslash 0/1, 1\backslash 00, 000/\}$ requires listing all transients that occur on each region

- region A has the transients 0, 1 and $\backslash$
- region B has the transients 0, $\backslash$
- region C has the transients 0, $/$, and
- region D has the transients 0, 1, $/$

the shape of the resulting behavior summary for node $a$ is listed in Figure 4.6.




*Waveform response graphs* and their application for two inverters in series, which are part of the equivalent Domino logic example.

Figure 4.5: Output response at each node of a Domino AND Gate

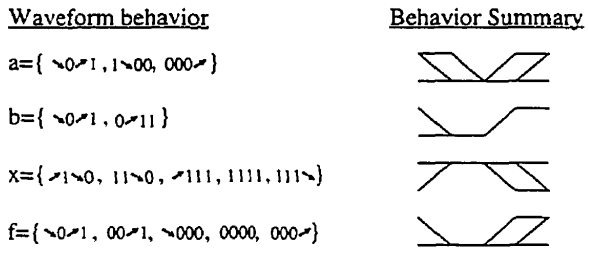



Figure 4.6: Waveform Behavior and summary at each node of a Domino AND Gate

### 4.1.3 Domino NAND gate using Behavior Categories

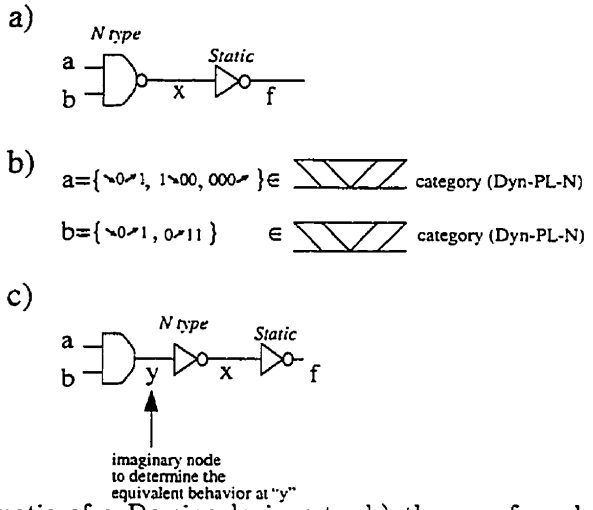
The Domino gate in this example consists of an N type dynamic NAND gate driving a Static type buffer, the same as in Section 4.1.2. The *behavior category* for each input is given in Figure 4.7b and it is defined under a CKLH(N) timing. The objective is of finding the *category* of the output behavior at node $f$.

#### Solution

The solution to this example has two steps. The first one is the decomposition of the logic gates into the equivalent circuit of Figure 4.7c, and the second step is the propagation of behaviors up to node $f$.

Similarly to the previous example, the input behaviors to the logic AND gate are being used to find an equivalent. The output response in node $y$ has this equivalent behavior and to find it the table in Figure 3.33 is applied. The contents of this table indicate that if two behavior categories are the same then the output of a logic gate is this same behavior category.

The propagation of the equivalent behavior from node $y$ to node $x$ gives the output response of the dynamic NAND gate. The response by the N type gate is found by localizing its input behavior in the network graph of Figure 3.31, the portion of this graph meaningful to this example is illustrated again in Figure 4.8b. The label *Dyn-PL-PH-N* indicates a set of gate types, which includes the N type, whose input behavior is *Dyn-PL*, or *Dyn-PL-N*



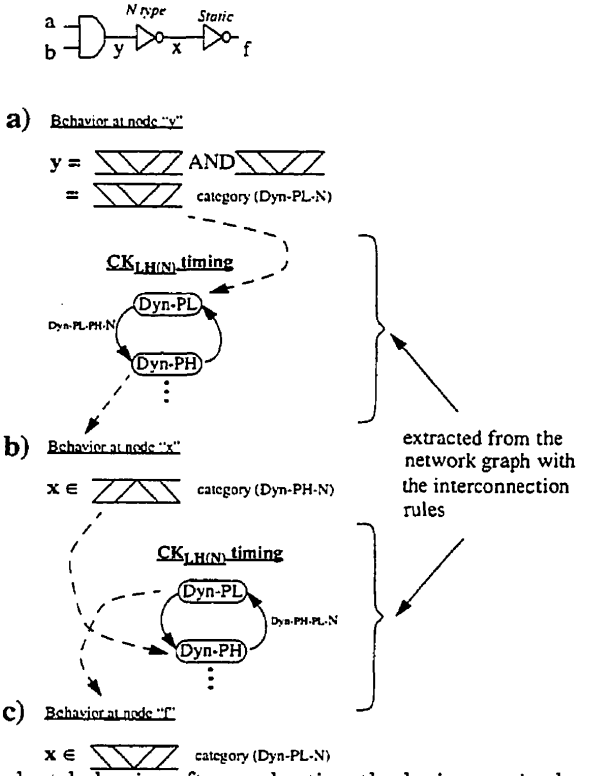

a) schematic of a Domino logic gate, b) the waveform behavior *category* at nodes *a* and *b*, and c) the equivalent schematic for the purpose of analysis.

Figure 4.7: Domino AND gate and its input behavior categories

under a CKLH(N) timing, and its output behavior belongs to the category *Dyn-PH-N*.

The propagation of waveforms from node $x$ to node $f$ follows the same procedure. First, the input behavior is allocated in the interconnection rules illustrated by the network graph of Figure 3.31. Next, the set containing the Static gate type is localized, which in this case is *Dyn-PH-PL-N*. Therefore, the output behavior at node $f$ is the behavior category of *Dyn-PL-N* as illustrated in Figure 4.8b and Figure 4.8c.

Observe the similarity between the explicit behavior for node $f$ found in Figure 4.6 and the behavior category just found in Figure 4.8c. The behavior in Figure 4.6 is a subset of Figure 4.8c.




a) The equivalent behavior after evaluating the logic AND is the *Dyn-PL-N* category. This behavior is localized in the network graph of interconnection rules to determine which is the output behavior of an N type inverter. b) The output behavior to this N type inverter belongs to the *Dyn-PH-N* category, and is the input to a Static type inverter. The network graph indicates that the output waveform at node *f* belongs to the *Dyn-PL-N* category, illustrated in c).

Figure 4.8: Output behavior category at each node of a Domino AND Gate

## 4.2 Compatible Gates and Behaviors to Unclassified Circuits

The first example in this section illustrates the use of the rules to correctly drive an example of an All-N Logic dynamic circuit, and the second example determines which gate types can be driven by the output node of a Domino gate.

### 4.2.1 Alternatives to the Input of a Gate

The circuit of Figure 4.9 is extracted from the figures in [GE96]. The objective of this exercise is to determine what kind of waveforms are able to properly drive this circuit.

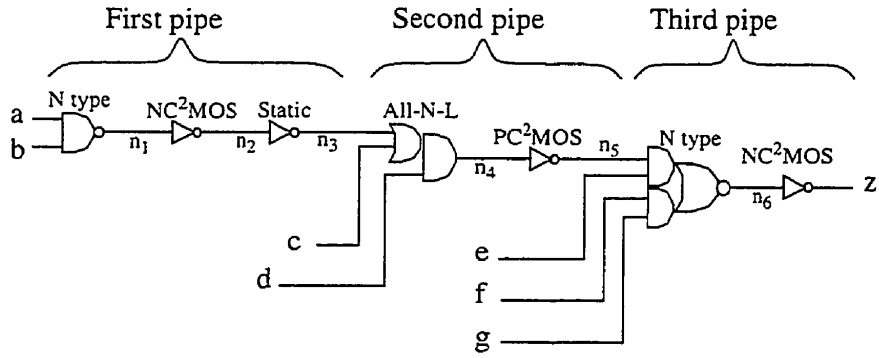
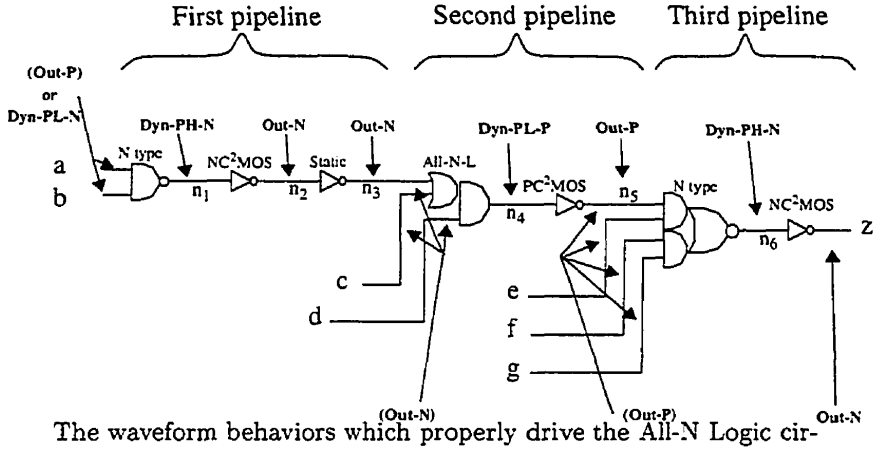



Figure 4.9: Sample All-N Logic circuit

**Solution**

The compatible waveforms to this circuit are specified in Figure 4.10 by using the behavior categories of Figure 3.30. The gate response is found by using the gate type sets and interconnection rules of Figure 3.31. Complex gates need to be studied with logic equivalent table of Figure 3.33.




The waveform behaviors which properly drive the All-N Logic circuit of Figure 4.9 are found after investigating the possibilities allowed by the interconnection rules.

Figure 4.10: Waveform Behavior for each node in the Sample circuit

### First pipe

The first gate is the N type dynamic NAND gate with the input nodes $a$ and $b$. N type gates are found in the sets *Border-N* and *Dyn-PL-PH-N* and their possible input behaviors are "*(Out-P)*" and "*Dyn-PL-N*" respectively. These two behaviors can only be generated by a logic AND gate whenever $a$ and $b$ belong to the same behavior category, either "*(Out-P)*" or "*Dyn-PL-N*", refer to the table in Figure 3.33. The output behavior at node $n_1$ is in either case the same behavior category of "*Dyn-PH-N*".

The output response by the N–C$^2$MOS gate between nodes $n_1$ and $n_2$ is next. The gate type for this inverter is contained by the set *Dyn-Out-N* and in the network graph this gate has the output behavior category of "*Out-N*". The same behavior "*Out-N*" is found at the output of the following Static inverter, located between nodes $n_2$ and $n_3$.

### Second pipe

All–N–L gates are found in the sets *Border-P* and *Dyn-PL-PL-P*, resulting in an input behavior of "*(Out-N)*" or "*Dyn-PL-P*", these two behaviors are

compatible with a buffer of an All–N–L type. The equivalent circuit separates the logic from the All–N–L buffer, therefore the logic gate driven by the input nodes $n_3$, $c$ and $d$ should be either "*(Out-N)*" or "*Dyn-PL-P*".

The behavior at node $n_3$ is set to "*Out-N*", which is crossed into a compatible timing to give the behavior "*(Out-N)*". The table in Figure 3.33 indicates that there is no other choice for $c$ and $d$ but to assign them to an "*(Out-N)*" behavior, otherwise they won't be able to properly drive an All–N–L gate.

#### Third pipe

The behavior category at node $n_4$ is *Dyn-PL-P* and at node $n_5$ it is the behavior category "*Out-P*". Similarly to the All–N–L gate, the N type gate, driving node $n_6$, can only have the input behavior of "*(Out-P)*" for all inputs. The only possibility for the output behavior at node $n_6$ is the "*Dyn-PH-N*" category.

The output response for the entire circuit is the behavior at node $z$. According to the network graph of Figure 3.31, if the input behavior to an N–C$^2$MOS inverter is the behavior category "*Dyn-PH-N*" then its output response is the "*Out-N*" behavior.

### 4.2.2 Alternatives to the Output of a Gate

A Domino gate with unspecified input behaviors is given in Figure 4.11, and is required to drive many types of buffers. The objective of this example is of finding which buffer types are compatible with this gate.

#### Solution

The Domino gate is made of a dynamic N type gate driving a Static type inverter. There are two possible input behaviors at node $a$ and $b$; regardless of what they are the output behavior at node $x$ is the behavior "*Dyn-PL-N*". The proof for this conclusion is easily found in Figure 3.32.

The network graph for the rules in the new technique indicates which gate types can be driven by this behavior. The label in Figure 3.31 next to

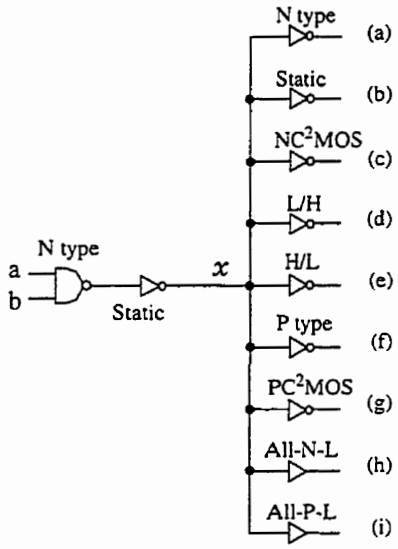



Figure 4.11: Output alternatives to a Domino Logic gate

the edge departing from the node marked as "Dyn-PL" is the set

$$\textit{Dyn-PL-PH-N} = \{\text{L/H}, \text{Static}, \text{N}, \text{N–C}^2\text{MOS}\}$$

therefore, the only gate types which can be driven by a Domino gate are the buffers labelled as (a), (b), (c) and (d) of Figure 4.11.

## 4.3 Comparison of Forecasted Waveform Shapes against Simulations

The dynamic logic version of the benchmark circuit *cm150a* is simulated in this section and compared against the waveform shapes predicted by the *waveform response graphs*. This benchmark circuit is the final version for the *cm150a* circuit found in Chapter 6. The schematic for the *cm150a* benchmark is illustrated in Figure 4.12 and the type for each gate will be indicated as the comparison continues thru the five stages comprising the circuit.

### WRG's and Logic Tables

The circuit utilizes only four types of gates. The Static, N–C$^2$MOS, P–C$^2$MOS and the All–N–L gate type. The *waveform response graphs* for these types are illustrated again in Figure 4.13a for convenience, and the tables for logic functions in Figure 4.13b

The predicted waveforms at each node of the circuit are given as a list of contents per clock cycle with the following notation:

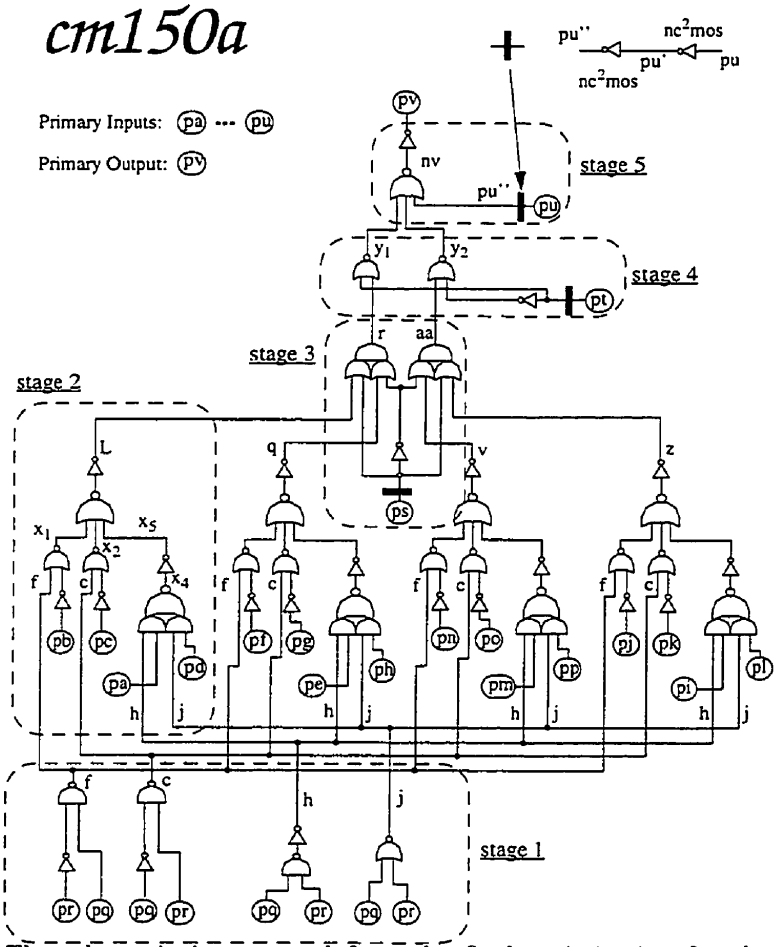
**Binary Value Contents:** $n = b(v_1, v_2, ..)$. $n$ is a node whose contents at the end of region D (final value at the end on the evaluate state) for the first clock cycle is the binary value $v_1$. The next clock cycle, node $n$ has a final binary value of $v_2$, and so on. For example $pq = b(1, 0, 1, X)$, where $X$ is a *don't care.*

**Waveform Shape Contents:** $n = tm(wv_1, wv_2, ..)$. $n$ is a node which has a waveform of shape $wv_1$ during the first clock cycle. The next clock cycle node $n$ has a waveform of shape $wv_2$, and so on. $tm$ is the timing under which the waveforms are defined. The final steady level at region D, of each waveform, indicates the binary value of the waveform. For example, the input node $pq$ will be assumed to have:

$$pq = \text{CKLH(N)}(XXX1, \diagup 1\diagdown, \diagdown 0\diagup, 11XX) = b(1, 0, 1, X)$$

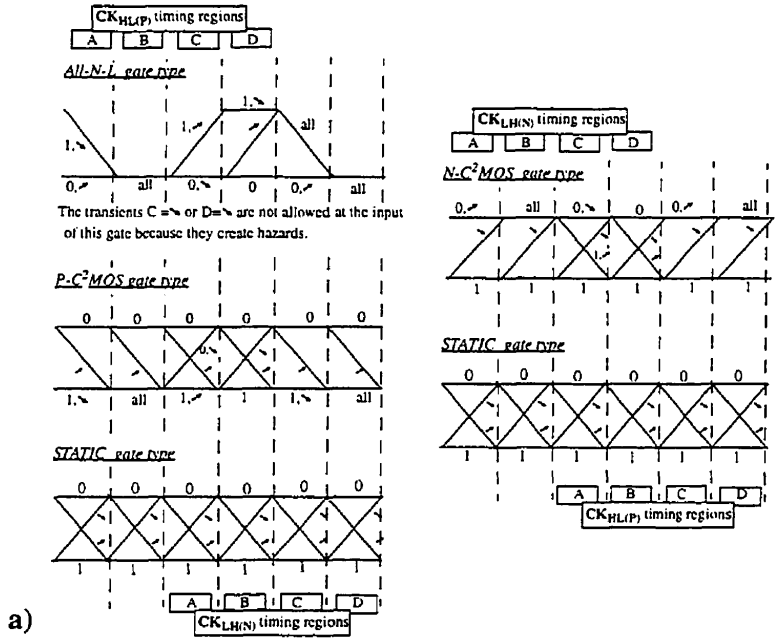
where $X$ is an unknown value or a *don't care.*

The following is the notation to evaluate a *waveform response graph* or a logic function:




The schematic is extracted from the final optimization for the *cm150a* benchmark that is performed in Chapter 6. The primary inputs belong to the *Ext-N* category. The type of timing for the primary inputs is CKLH(N).

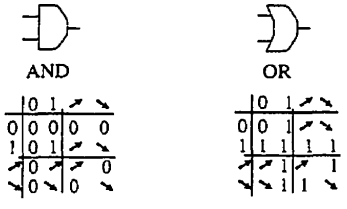
Figure 4.12: *cm150a* benchmark circuit




a)

- These waveform respose graphs apply only for inverters and buffers.
- The edges indicate the shape of the **output waveform**.
- The labels 0, 1, ↗, and ↘ indicate the shape of the **input waveform** on the specified region (A, B, C or D).
- The label "all" gives the specified output regardless of the input.

b)




a) *waveform response graphs* and b) logic evaluation tables

Figure 4.13: WRG's useful for the *cm150a* benchmark

***Waveform Response Graph* Evaluation:** $n = \text{WRG}\{type(i)\}$. $n$ is a node whose contents are the result of applying the *waveform response graph* for a buffer of type *type* and input $i$. For example if

$$i = \text{CKHL(P)}(X111, \backslash 000, 0/11) = b(1, 0, 1)$$

then

$$w = \text{WRG}\{\text{All-N-L}(i)\} = \text{CKHL(P)}(X0/1, \backslash 000, 00/1) = b(1, 0, 1)$$

(All-N-L gates are non-inverting).

**Logic Function Evaluation:** $n = f(a, b)$. $n$ is a node whose contents are the result of evaluating the logic function $f$ with the input waveforms $a$ and $b$. For example if

$$a = \text{CKHL(P)}(X111, 1\backslash 00, /111) = b(1, 0, 1)$$

and

$$b = \text{CKHL(P)}(X111, \backslash/11, \backslash/11) = b(1, 1, 1)$$

then

$$w = a \bullet b = \text{CKHL(P)}(X111, \backslash 000, 0/11) = b(1, 0, 1)$$

where "$\bullet$" is logic function AND.

### 4.3.1 Stage One

#### Predicted Results

The schematic for the gates within Stage One have their gate type assigned in Figure 4.14. The waveforms in this figure are the predicted waveforms where given the two primary inputs $pq$ and $pr$ to be defined by

$$\begin{array}{rcll} pq & = & \text{CK}_{\text{LH(N)}}(XXX1, \backslash/1\backslash, /\backslash 0/, 11XX) & = & b(1,0,1,X) \\ pr & = & \text{CK}_{\text{LH(N)}}(XXX0, /1\backslash 0, 0/11, 11XX) & = & b(0,0,1,X) \end{array}$$

The first waveforms ($XXX1$ and $XXX0$) define a dummy clock cycle which loads the initial conditions in a CK$_{\text{LH(N)}}$ pipe. The last waveforms ($11XX$ for both) allow for the propagation of the final data in a CK$_{\text{HL(P)}}$ pipe. The inverted signals for $pq$ and $pr$ are

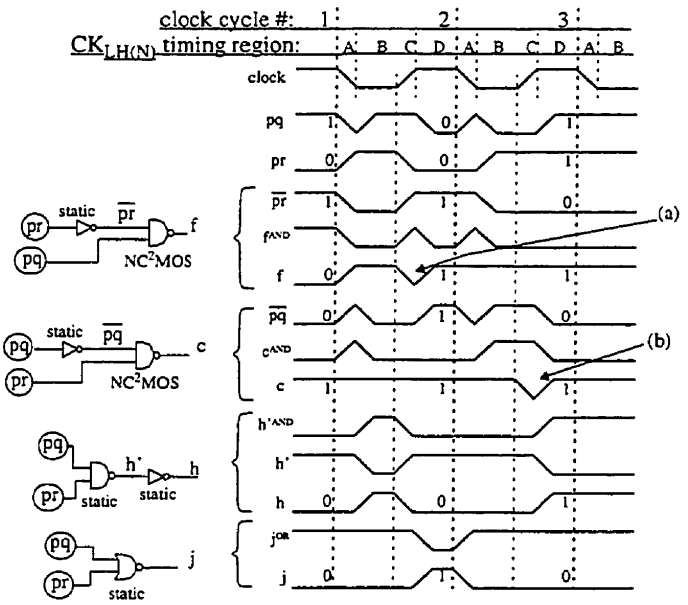
$$\begin{array}{rcll} \overline{pq} & = & \text{WRG}\{\text{Static}(pq)\} & = & \text{CK}_{\text{LH(N)}}(XXX0, /\backslash 0/, \backslash/1\backslash, 00XX) \\ & & & = & b(0,1,0,X) \\ \overline{pr} & = & \text{WRG}\{\text{Static}(pr)\} & = & \text{CK}_{\text{LH(N)}}(XXX1, \backslash 0/1, 1\backslash 00, 00XX) \\ & & & = & b(1,1,0,X) \end{array}$$

The equivalent behavior for the AND gate in $f$ is $f^{AND}$:

$$\begin{array}{rcll} f^{AND} & = & (\overline{pr} \bullet pq) & = & \text{CK}_{\text{LH(N)}}(XXX1, \backslash 0/\backslash, /\backslash 00, 00XX) \\ & & & = & b(1,0,0,X) \\ f & = & \text{WRG}\{\text{N-C}^2\text{MOS}(f^{AND})\} & = & \text{CK}_{\text{LH(N)}}(XXX0, /1\backslash/, 1111, 11XX) \\ & & & = & b(0,1,1,X) \end{array}$$

Similarly for $c, h', h$ and $j$:

$$
\begin{array}{lcll}
c^{AND} & = & (\overline{pq} \bullet pr) & = \text{CK}_{\text{LH(N)}}(XXX0, /\backslash 00, 0/1\backslash, 00XX) \\
 & & & = b(0,0,0,X) \\
c & = & \text{WRG}\{\text{N-C}^2\text{MOS}(c^{AND})\} & = \text{CK}_{\text{LH(N)}}(XXX1, 1111, 11\backslash/, 11XX) \\
 & & & = b(1,1,1,X) \\
h'^{AND} & = & (pq \bullet pr) & = \text{CK}_{\text{LH(N)}}(XXX0, 0/\backslash 0, 000/, 11XX) \\
 & & & = b(0,0,1,X) \\
h' & = & \text{WRG}\{\text{Static}(h'^{AND})\} & = \text{CK}_{\text{LH(N)}}(XXX1, 1\backslash/1, 111\backslash, 00XX) \\
 & & & = b(1,1,0,X) \\
h & = & \text{WRG}\{\text{Static}(h')\} & = \text{CK}_{\text{LH(N)}}(XXX0, 0/\backslash 0, 000/, 11XX) \\
 & & & = b(0,0,1,X) \\
j^{OR} & = & (pq + pr) & = \text{CK}_{\text{LH(N)}}(XXX1, 111\backslash, /111, 11XX) \\
 & & & = b(1,0,1,X) \\
j & = & \text{WRG}\{\text{Static}(j^{OR})\} & = \text{CK}_{\text{LH(N)}}(XXX0, 000/, \backslash 000, 00XX) \\
 & & & = b(0,1,0,X)
\end{array}
$$



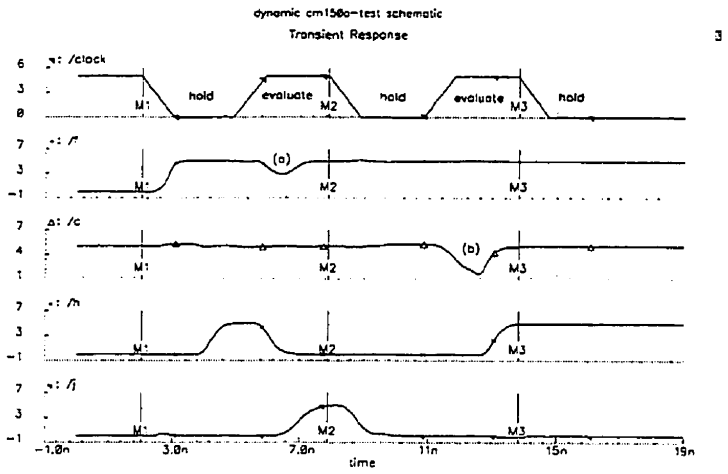

Points (a) and (b) are also specified in Figure 4.15.

Figure 4.14: Schematic and Predicted waveforms for stage 1 of *cm150a*

### Simulated Results

Node $f$ is expected to do a $\backslash\!/$ sequence in regions C and D of cycle 2. However the simulation results of Figure 4.15a do not show a full swing neither a time alignment for this sequence of transients. The reason is because the heavy load to gate $f$ is significant and together with an imbalance between input signal arrivals the output has little time to react to two full swing transitions.

The output at node $c$ is also expected to perform a $\backslash\!/$ sequence. This sequence is able to do a full swing because the transient $\backslash$ is caused by well defined VH levels at the inputs. The delay present in the $\backslash$ transient of $c$ is due to the large output load.




Full swing was predicted at point a), however due to the large load at node $f$ and the differential of arrival between the $\overline{pr}$ and the $pq$ signals the $\backslash\!/$ sequence is barely visible.

Figure 4.15: Simulated waveforms for stage 1

### 4.3.2 Stage Two

#### Predicted Results

The gates in Stage Two are defined within the CKLH(N) timing and are illustrated in Figure 4.16. The predicted waveforms for node $x_1$ are obtained by propagating the primary input $pb$ and the waveforms found at node $f$ to nodes $\overline{pb}$ and $x_1$. Node $x_1^{OR}$ is the equivalent behavior for the OR gate in $x_1$.

$$\begin{array}{rcll} f & = & \text{CKLH(N)}(XXX0, /1\backslash/, 1111, 11XX) & = & b(0,1,1,X) \\ pb & = & \text{CKLH(N)}(XXX1, \backslash 0/1, \backslash/\backslash/, 11XX) & = & b(1,1,1,X) \end{array}$$

$$\begin{array}{rcll} \overline{pb} & = & \text{WRG}\{\text{N-C}^2\text{MOS}(pb)\} & = & \text{CKLH(N)}(XXX0, /1\backslash 0, /11\backslash, 00XX) \\ & & & = & b(0,0,0,X) \\ x_1^{OR} & = & (f + \overline{pb}) & = & \text{CKLH(N)}(XXX0, /1\backslash/, 1111, 11XX) \\ & & & = & b(0,1,1,X) \\ x_1 & = & \text{WRG}\{\text{Static}(x_1^{OR})\} & = & \text{CKLH(N)}(XXX1, \backslash 0/\backslash, 0000, 00XX) \\ & & & = & b(1,0,0,X) \end{array}$$

The output behavior at node $x_2$ is found from the primary input $pc$ and $c$. Node $x_2^{OR}$ has the equivalent behavior for the OR gate in $x_2$.

$$\begin{array}{rcll} c & = & \text{CKLH(N)}(XXX1, 1111, 11\backslash/, 11XX) & = & b(1,1,1,X) \\ pc & = & \text{CKLH(N)}(XXX1, \backslash/\backslash 0, /\backslash/1, 11XX) & = & b(1,0,1,X) \end{array}$$

$$\begin{array}{rcll} \overline{pc} & = & \text{WRG}\{\text{N-C}^2\text{MOS}(pc)\} & = & \text{CKLH(N)}(XXX0, /111, 11\backslash 0, 00XX) \\ & & & = & b(0,1,0,X) \\ x_2^{OR} & = & (c + \overline{pc}) & = & \text{CKLH(N)}(XXX1, 1111, 11\backslash/, 11XX) \\ & & & = & b(1,1,1,X) \\ x_2 & = & \text{WRG}\{\text{Static}(x_2^{OR})\} & = & \text{CKLH(N)}(XXX0, 0000, 00/\backslash, 00XX) \\ & & & = & b(0,0,0,X) \end{array}$$

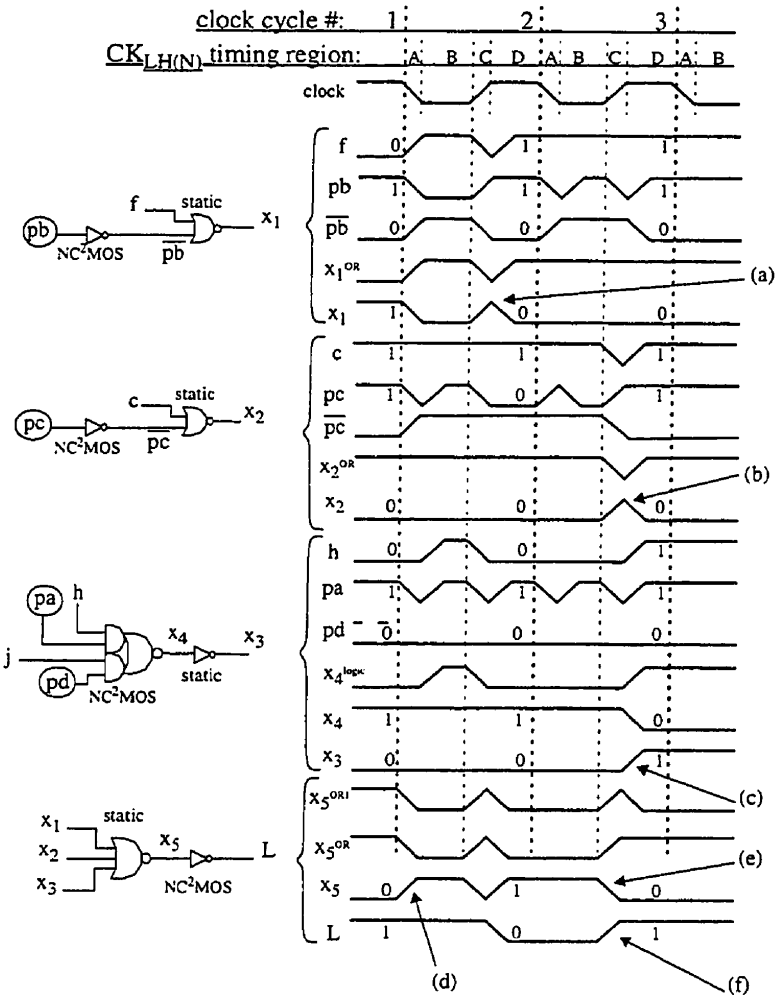
The output behavior at nodes $x_4$ and $x_3$ are dependent on the primary inputs $pa$ and $pd$, and on the waveforms at nodes $j$ and $h$. The waveform in $pd$ is a constant VL level and it helps simplify the contents of the complex gate of $x_4^{logic} = (h\,pa + j\,pd)$ into $x_4^{logic} = (h\,pa)$.

$$\begin{array}{rcll} h & = & \text{CKLH(N)}(XXX0, 0/\backslash 0, 000/, 11XX) & = & b(0,0,1,X) \\ pa & = & \text{CKLH(N)}(XXX1, \backslash/\backslash/, \backslash/\backslash/, 11XX) & = & b(1,1,1,X) \\ pd & = & \text{CKLH(N)}(XXX0, 0000, 0000, 00XX) & = & b(0,0,0,X) \end{array}$$

$$
\begin{array}{lcll}
x_4^{logic} & = & (h \bullet pa) & = \text{CK}_{\text{LH}(\text{N})}(XXX0, 0/\backslash 0, 000/, 11XX) \\
 & & & = b(0,0,1,X) \\
x_4 & = & \text{WRG}\{\text{N-C}^2\text{MOS}(x_4^{logic})\} & = \text{CK}_{\text{LH}(\text{N})}(XXX1, 1111, 111\backslash, 00XX) \\
 & & & = b(1,1,0,X) \\
x_3 & = & \text{WRG}\{\text{Static}(x_4)\} & = \text{CK}_{\text{LH}(\text{N})}(XXX0, 0000, 000/, 11XX) \\
 & & & = b(0,0,1,X)
\end{array}
$$

The main output for Stage Two is $L$. The logic OR in $x_5$ has the equivalent $x_5^{OR}$, and is defined by means of two nodes $x_5^{OR1} = (x_1 + x_2)$ and $x_5^{OR} = (x_5^{OR1} + x_3)$

$$
\begin{array}{lcll}
x_5^{OR1} & = & (x_1 + x_2) & = \text{CK}_{\text{LH}(\text{N})}(XXX1, \backslash 0/\backslash, 00/\backslash, 00XX) \\
 & & & = b(1,0,0,X) \\
x_5^{OR} & = & (x_5^{OR1} + x_3) & = \text{CK}_{\text{LH}(\text{N})}(XXX1, \backslash 0/\backslash, 00/1, 11XX) \\
 & & & = b(1,0,1,X) \\
x_5 & = & \text{WRG}\{\text{Static}(x_5^{OR})\} & = \text{CK}_{\text{LH}(\text{N})}(XXX0, /1\backslash/, 11\backslash 0, 00XX) \\
 & & & = b(0,1,0,X) \\
L & = & \text{WRG}\{\text{N-C}^2\text{MOS}(x_5)\} & = \text{CK}_{\text{LH}(\text{N})}(XXX1, 111\backslash, 00/1, 11XX) \\
 & & & = b(1,0,1,X)
\end{array}
$$




Points (a) thru (f) are also specified in Figure 4.17.

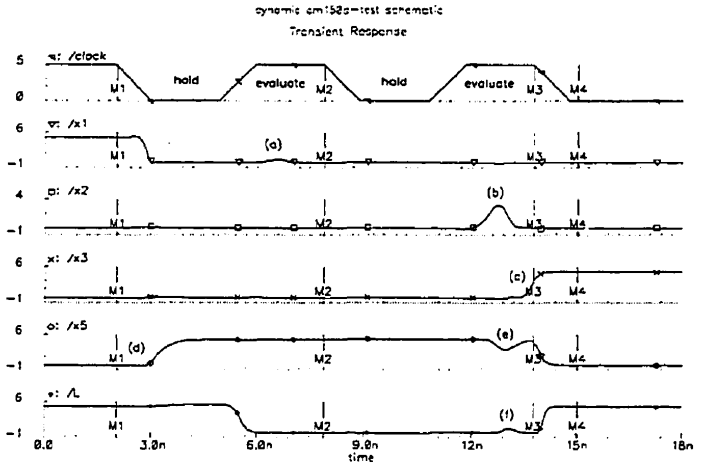
Figure 4.16: Schematic and Predicted waveforms for stage 2 of *cm150a*

### Simulated Results

The predicted behavior at node $x_1$ assumes that the input signal $f$ is able to perform a CD $= \backslash /$ sequence during cycle # 2. However the waveforms from the simulation show that node $f$ is barely able to show this sequence. Subsequently the results in Figure 4.17a have a waveform in $x_1$ that is unable to detect this input sequence.

The simulation shows that the waveform in node $x_2$ has the right shape. A sequence of two transients during the clock cycle # 3 appears with a full voltage swing in Figure 4.17b, however these transient aren't located within the C and D regions as predicted. Propagation delays have affected the location of this sequence and now they occur within region D.

Node $x_3$ is also delayed and the final value isn't reached until the time marker $M_4$; the $/$ transient still occurs in Figure 4.17c because the driving gate of node $x_4$ reaches VL and the propagation of this signal from node $x_4$ to node $x_3$ is thru a Static inverter. More delays are observed in $x_5$ and $L$ as they propagate the signals in $x_1, x_2$ and $x_3$.




The gate driving node $x_1$ is also unable to detect in (a) node $f$'s $\downarrow\uparrow$ sequence. The other waveforms have the predicted shape but are affected by delays. The extent of such delays is clear in c), e) and f) where transients are unable to stay within the $V_H$ level of the clock. However no loss of data occurs here due to the quick response of the N–C$^2$MOS gate driving node $L$.

Figure 4.17: Simulated waveforms for stage 2

### 4.3.3 Stage Three

#### Predicted Results

The timing type for Stage Three is CKHL(P), and as a consequence all input signals to this stage which are defined within a CKLH(N) timing must be *crossed* to a CKHL(P) timing. This crossing is an **overlap** of regions between both timings as illustrated in Figure 4.18.

For example, the waveform in node $L$ has the following definition

$$L = \text{CKLH(N)}(XXX1, 111\backslash, 00/1, 11XX) = b(1,0,1,X)$$

This waveform is crossed to a CKHL(P) timing by shifting the location of its timing regions and by filling with $X$ the unknown transients. Node $L$ is then

$$\begin{array}{rcll} L & = & \text{CKHL(P)}(XXXX, X111, 1\backslash 00, /111, XXXX) & = b(X,1,0,1,X) \\ & = & \text{CKHL(P)}(X111, 1\backslash 00, /111) & = b(1,0,1) \end{array}$$

The same *crossing* is performed for the waveforms in $ps''$ and $\overline{ps''}$.

$$ps = \text{CKLH(N)}(XXX0, /\backslash/\backslash, /\backslash/\backslash, 00XX) = b(0,0,0,X)$$

$$\begin{array}{rcll} ps' & = & \text{WRG}\{\text{N-C}^2\text{MOS}(ps)\} & = \text{CKLH(N)}(XXX1, 11\backslash/, 11\backslash/, 11XX) \\ & & & = b(1,1,1,X) \\ ps'' & = & \text{WRG}\{\text{N-C}^2\text{MOS}(ps')\} & = \text{CKLH(N)}(XXX0, 00/\backslash, 00/\backslash, 00XX) \\ & & & = b(0,0,0,X) \\ & & & = \text{CKHL(P)}(X000, /\backslash 00, /\backslash 00) \\ & & & = b(0,0,0) \\ \overline{ps''} & = & \text{WRG}\{\text{Static}(ps'')\} & = \text{CKLH(N)}(XXX1, 11\backslash/, 11\backslash/, 11XX) \\ & & & = b(1,1,1,X) \\ & & & = \text{CKHL(P)}(X111, \backslash/11, \backslash/11) \\ & & & = b(1,1,1) \end{array}$$

The primary inputs affecting the waveform in node $q$ are such that this node is a constant VL level.

$$q = \text{CKHL(P)}(X000, 0000, 000) = b(0,0,0)$$

As a consequence the logic function in $r = (L + ps'') \bullet (q + \overline{ps''})$ is simplified

to $r = (L + ps'')\overline{ps''}$, or to $r = (L \bullet \overline{ps''})$

$$
\begin{array}{lcll}
r^{(L+ps'')} & = & (L + ps'') & = \text{CK}_{\text{HL}}(\text{P})(X111, 1\backslash 00, /111) \\
 & & & = b(1,0,1) \\
r^{AND} & = & (r^{(L+ps'')} \bullet \overline{ps''}) & = \text{CK}_{\text{HL}}(\text{P})(X111, \backslash 000, 0/11) \\
 & & & = b(1,0,1) \\
r & = & \text{WRG}\{\text{All-N-L}(r^{AND})\} & = \text{CK}_{\text{HL}}(\text{P})(X0/1, \backslash 000, 00/1) \\
 & & & = b(1,0,1)
\end{array}
$$

Similarly at node $aa$, the input waveforms at nodes $z$ and $v$ are a steady $V_L$ level

$$z = v = \text{CK}_{\text{HL}}(\text{P})(X000, 0000, 0000) = b(0,0,0)$$

which simplifies the logic of $aa = (\overline{ps''} + v) \bullet (ps'' + z)$ into $aa = 0$ or the waveform

$$aa = \text{CK}_{\text{HL}}(\text{P})(X000, 0000, 0000) = b(0,0,0)$$

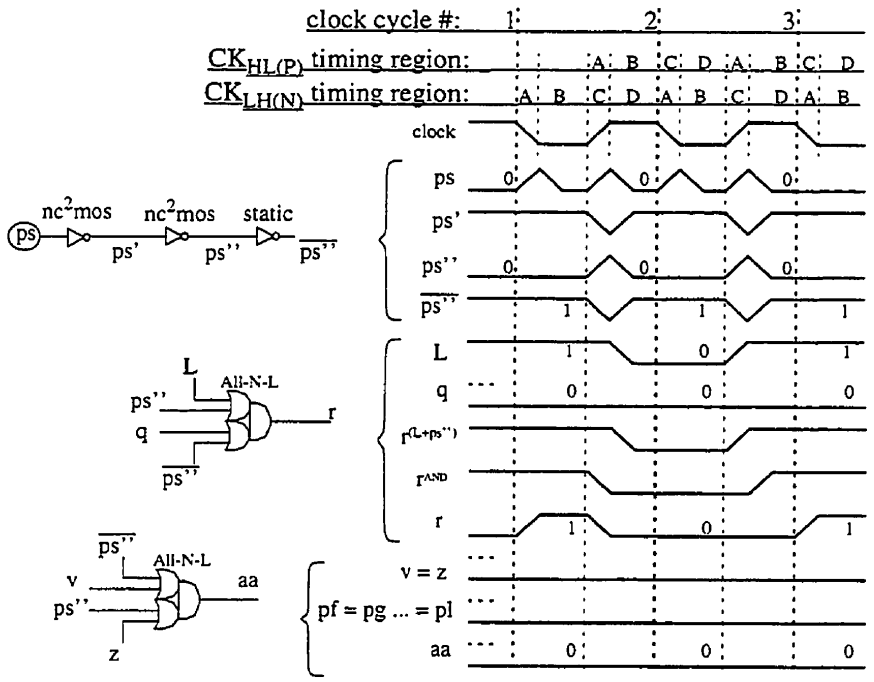
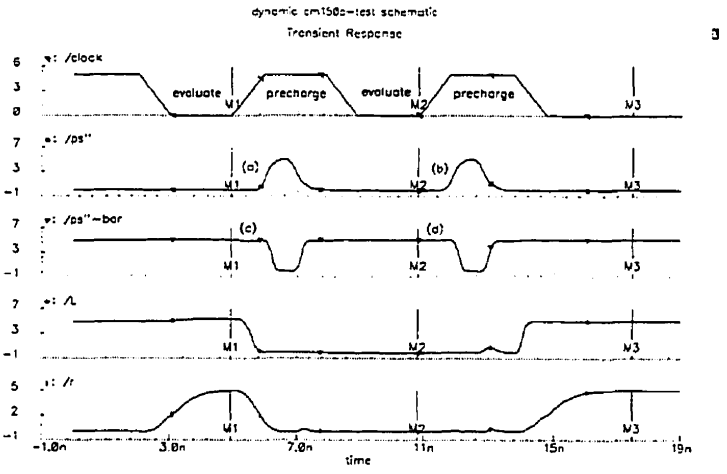



Figure 4.18: Schematic and Predicted waveforms for stage 3 of *cm150a*

### Simulated Results

The waveforms in the simulation of Figure 4.19a to Figure 4.19d show a delay at nodes $ps''$ and $\overline{ps''}$. These propagation delays are non-destructive during the VH level of the clock because that's when an N–C$^2$MOS gate is able to faithfully propagate a /\ sequence.

The waveform at node $L$ is now defined under the opposite timing and, as indicated in Figure 4.19, a final stable level is expected at the time markers $M_1, M_2$ and $M_3$. These markers are further in time when compared to those for a CKLH(N) timing definition and therefore allow for more time to a final value to stabilize in $L$ and to propagate into $r$




All waveforms have the predicted shape, however delays have shifted the /\ and \/ sequences at nodes $ps''$ and $\overline{ps''}$.

Figure 4.19: Simulated waveforms for stage 3

### 4.3.4 Stage Four

#### Predicted Results

Stage Four is a simple circuit with one primary input $pt$ that is equal to $ps$. Therefore $pt'' = ps''$ and $\overline{pt''} = \overline{ps''}$. The predicted waveforms at nodes $y_1$ and $y_2$ are illustrated in Figure 4.20 and were obtained from the following description:

$$
\begin{array}{lcll}
pt'' & = & \text{CK}_{\text{HL(P)}}(X000, /\backslash 00, /\backslash 00) & = b(0,0,0) \\
r & = & \text{CK}_{\text{HL(P)}}(X0/1, \backslash 000, 00/1) & = b(1,0,1)
\end{array}
$$

$$
\begin{array}{lcll}
y_1^{OR} & = & (pt'' + r) & = \text{CK}_{\text{HL(P)}}(X0/1, 1\backslash 00, /\backslash/1) \\
 & & & = b(1,0,1) \\
y_1 & = & \text{WRG}\{\text{Static}(y_1^{OR})\} & = \text{CK}_{\text{HL(P)}}(X1\backslash 0, 0/11, \backslash/\backslash 0) \\
 & & & = b(0,1,0)
\end{array}
$$

$$
\begin{array}{lcll}
\overline{pt''} & = & \text{CK}_{\text{HL(P)}}(X111, \backslash/11, \backslash/11) & = b(1,1,1) \\
aa & = & \text{CK}_{\text{HL(P)}}(X000, 0000, 0000) & = b(0,0,0)
\end{array}
$$

$$
\begin{array}{lcll}
y_2^{OR} & = & (\overline{pt''} + aa) & = \text{CK}_{\text{HL(P)}}(X111, \backslash/11, \backslash/11) \\
 & & & = b(1,1,1) \\
y_2 & = & \text{WRG}\{\text{Static}(y_2^{OR})\} & = \text{CK}_{\text{HL(P)}}(X000, /\backslash 00, /\backslash 00) \\
 & & & = b(0,0,0)
\end{array}
$$

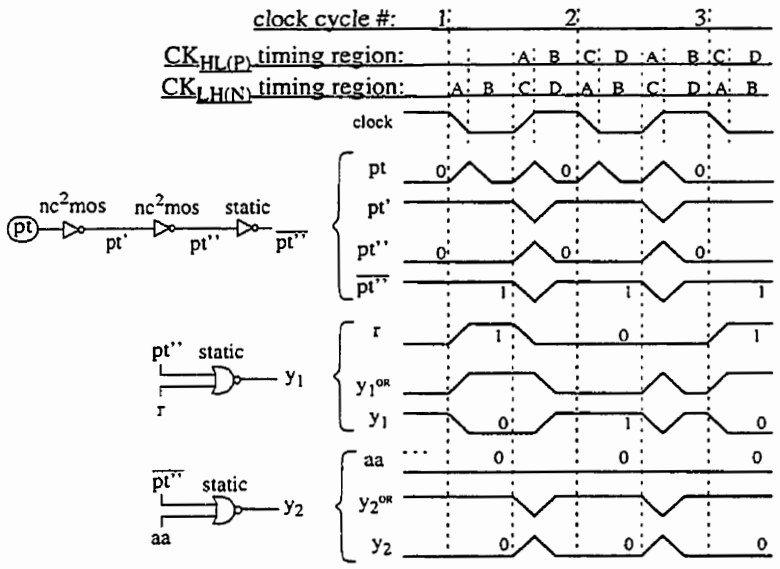
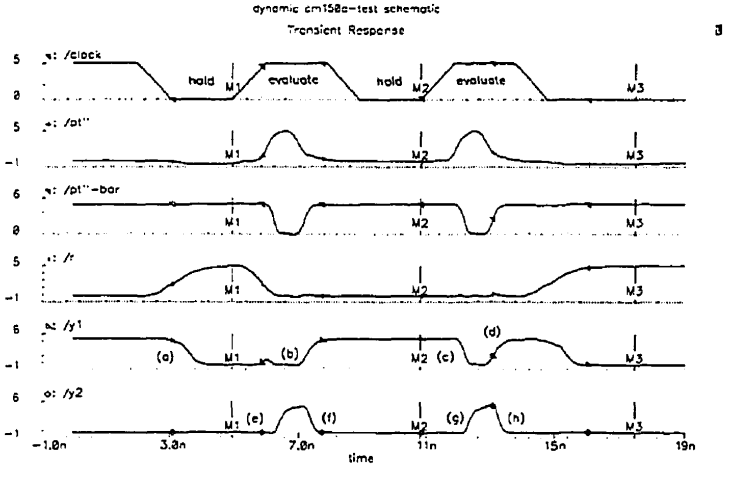



Figure 4.20: Schematic and Predicted waveforms for stage 4 of *cm150a*

### Simulated Results

The simulated circuit shows that the waveforms in Figure 4.21 at nodes $y_1$ and $y_2$ have the same shape as the predicted waveforms. There is a delay effect which has shifted the /\ and \/ sequences, however no levels have been corrupted since this event is the output of static gates and they do occur within the evaluation period.




All waveforms have the predicted shape, however delays have shifted the / and \ transients marked from (a) to (h).

Figure 4.21: Simulated waveforms for stage 4

### 4.3.5 Stage Five

#### Predicted Results

The gates in Stage Five have the following characteristics: The node $pu$ is a primary input and its waveform is equal to that in $ps$. The output of node $nv$ is a three input logic NOR that is evaluated by $nv^{OR} = (nv^{OR1} + pu'')$ where $nv^{OR1} = (y_1 + y_2)$.

The nodes in Stage Five are illustrated by the schematic in Figure 4.22 and have the following waveforms:

$$\begin{aligned} y_1 &= \text{CK}_{\text{HL(P)}}(X1\backslash 0, 0/11, \backslash/\backslash 0) = b(0,1,0) \\ y_2 &= \text{CK}_{\text{HL(P)}}(X000, /\backslash 00, /\backslash 00) = b(0,0,0) \end{aligned}$$

$$\begin{aligned} nv^{OR1} &= (y_1 + y_2) = \text{CK}_{\text{HL(P)}}(X1\backslash 0, /111, 11\backslash 0) \\ &= b(0,1,0) \\ nv^{OR} &= (nv^{OR1} + pu'') = \text{CK}_{\text{HL(P)}}(X1\backslash 0, /111, 11\backslash 0) \\ &= b(0,1,0) \\ nv &= \text{WRG}\{\text{P-C}^2\text{MOS}(nv^{OR})\} = \text{CK}_{\text{HL(P)}}(X0/1, \backslash 000, 00/1) \\ &= b(1,0,1) \\ pv &= \text{WRG}\{\text{P-C}^2\text{MOS}(nv)\} = \begin{cases} \text{CK}_{\text{HL(P)}}(X1\backslash 0, 00/1, 11\backslash 0) \\ \text{CK}_{\text{HL(P)}}(X000, 00/1, 11\backslash 0) \end{cases} \\ &= b(0,1,0) \end{aligned}$$

Node $pv$ is given two possible waveforms because there is an $X$ transient on region A of the first clock cycle. This undefined transient cannot be ignored because the P-C$^2$MOS gate is a latch which at region A is defining the output level to hold during region B. Subsequently two possibilities are given, either the $X1\backslash 0$ or the $X000$ waveform will be present at node $pv$ for the duration of the clock cycle # 1.

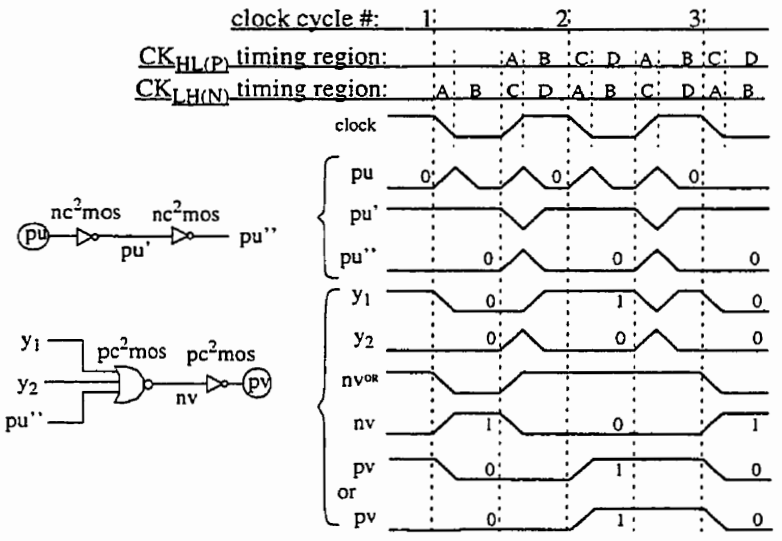
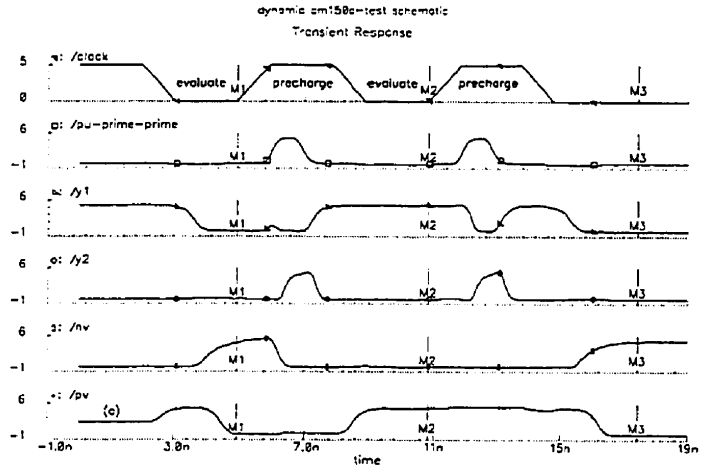



Figure 4.22: Schematic and Predicted waveforms for stage 5 of *cm150a*

### Simulated Results

The waveforms from the simulated circuit in Figure 4.23 show that nodes *nv* and *pv* have all the predicted transients, which are also affected by delays.

An important feature of these waveforms is found in node *pv* at the beginning of the clock cycle # 1. The predicted waveforms indicated that the initial conditions for this node could not be established. This uncertainty is verified in Figure 4.23a where the initial voltage is maintained at an intermediate level until the dynamic P–C$^2$MOS gate enters the evaluate period. Once this node is given a path to one of the supplies the waveforms follow the same shape as predicted by the *waveform response graphs*.




An undetermined initial level was forecasted at node *pv*, this is visible in (a) where a metastable level is held for the region predicted by the *waveform response graphs.*

Figure 4.23: Predicted and Simulated waveforms for stage 5

# Chapter 5

# Performance Measurement Model

This chapter presents the components of a performance measurement model based on the behavior information derived from the analysis in Chapter 3. This model is presented as an application example to illustrate the potential of the waveform representation model and the Waveform Response Graphs.

The property of a circuit selected to be optimized is the dynamic power consumption and is based on the charge/discharge of the gate node in a MOS transistor. For this reason the description of waveforms is extended to represent some forms of glitches. There are other possible objectives such as propagation delay and consumed silicon area. However the dynamic power parameter demonstrates the practical use of the waveform behaviors of Chapter 3.

The waveform description model, and the Waveform Response Graphs, provide enough information to appropriately investigate the basic switching behavior of the fundamental gates. The switching behavior is found by the charge/discharge of the gate nodes as described by each type of waveform and its probability. Therefore, the probability of these switching transients is the average switching activity to occur within a single clock period. The waveform probabilities are given no correlation from one cycle to the next one because this type of probabilities provide the simplest form of analysis for switching circuits [Yea98].

The propagation of waveform probabilities is accomplished by adapting traditional methods to a system of 32 waveform shapes and the Waveform Response Graphs. A computer program has been developed to facilitate the computation of these power estimates. The probability distribution is arbitrarily selected to a uniform distribution and the W/L ratio is also arbitrary. Therefore $P(i) = p(j)$ and $\sum P(x) = 1$ for all input waveforms.

There are alternative methods of getting this information, such as circuit simulations. However, the objective of this model is to provide a good idea of what to expect in a circuit and to demonstrate the application of the new concepts introduced in Chapter 3.

The organization of this chapter is as follows:

- The extended timing model is presented first and it defines the same 32 waveform shapes plus two new symbols defining anomalous glitch transients for $V_H$ and $V_L$ levels.
- Next, the extended *waveform response graphs* that account for glitches are presented for each of the fundamental gates.
- The procedure for propagating a waveform-probability is introduced for buffers and complex gates.
- Next, a delay mechanism is accounted into the waveform-probabilities.
- The power formulas are derived for each of the fundamental gates.
- The *cutting algorithm*[SDB84] is reviewed. This algorithm is the basis for an alternative solution to the conditional probabilities that are generated within the circuit.

## 5.1 Extended timing model

An extended timing model for the description of waveforms is introduced to account for anomalous transients on the $V_L$ and $V_H$ levels.

To describe the shape of a waveform and its transients let us first recall the notation for a single event as described in Section 3.1.2, where the symbol

0 denotes a $V_L$ to $V_L$ transition,
1 denotes a $V_H$ to $V_H$ transition,
/ denotes a $V_L$ to $V_H$ transition, and
\ denotes a $V_H$ to $V_L$ transition.

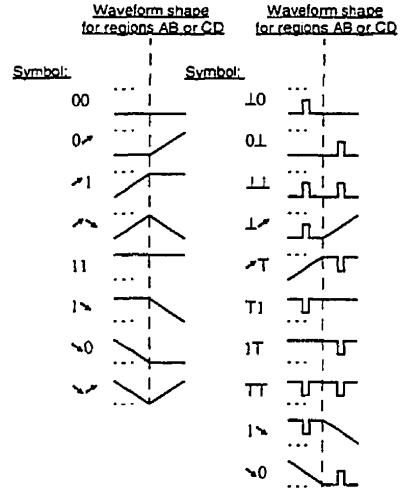
The new signal events are described by the symbol

- $\perp$ to represent a $V_L$ level with a positive glitch, and
- $\top$ to represent a $V_H$ level with a negative glitch.

In the *original timing model*, in "Chapter 3: Analysis of Dynamic Logic", four transitions are allowed per clock cycle, one for each of the ABCD regions, that help describe 32 waveform shapes. With the new transient symbols, $\perp$ and $\top$, the number of waveform shapes is larger making the analysis to determine the output waveforms a slow and therefore unfeasible method. For example, a complex gate with three inputs would have to test at least $32^3 = 32768$ input vectors, and considerably more under a full waveform set for the extended timing model.

The *simplified timing model* is introduced here to describe a waveform, within a full clock cycle, by using two smaller waveforms of two transients each. Giving in Figure 5.1a a total of 8 waveform shapes.

The *precharge* period is described by waveforms within the regions A and B, while the *evaluate* period is described by those within regions C and D. The output response of the same three-input complex gate is found by testing $8^3 + 8^3 = 1024$ vectors. This is 32 times faster at the expense of loosing the relationship between the final-state of the precharge state and the initial-state of the evaluate state.



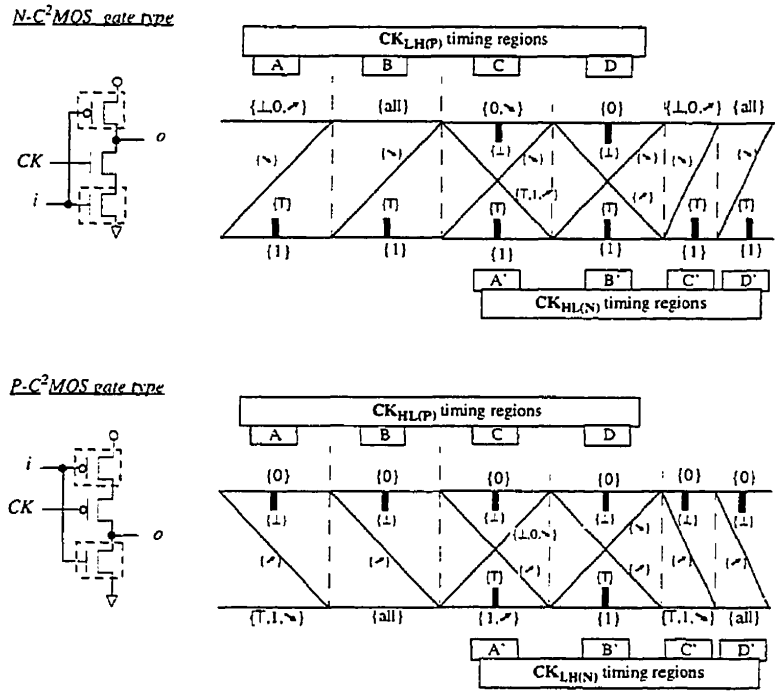

**(a)** **(b)**

(a) Two regions of activity, A and B or C and D, with four transients, the 0, 1, /, and \, define 8 possible waveform shapes for either the precharge or the evaluate period. (b) Under the extended timing model 10 more possible waveforms are defined.

Figure 5.1: Waveform shapes described by two regions

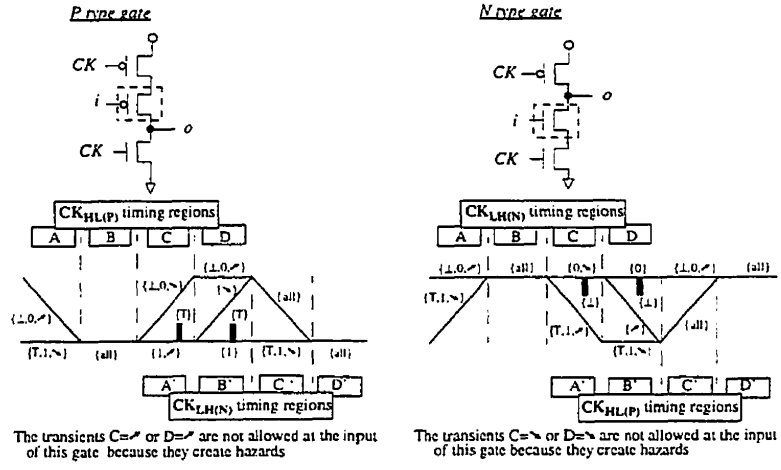
## 5.2 Extended Waveform Response Graphs

The glitches represented by the $\perp$ and $\top$ symbols are transients that have the potential of affecting the output response of a gate. Depending on the kind of protective mechanism these glitches may not appear at all or as indicated in Figure 5.4 an undefined level might occur. The modified *waveform response graphs* in Figure 5.2 to Figure 5.5 account for such glitches.



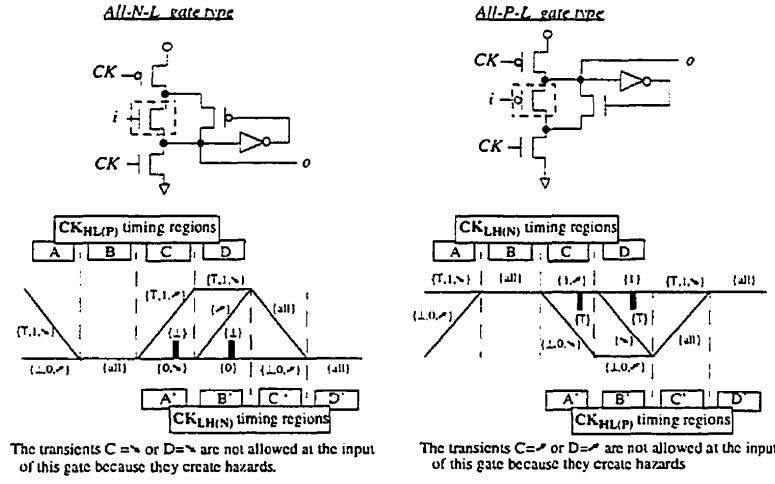

- These graphs apply only to inverters.
- The edges indicate the shape of the **output waveform.**
- The labels 0,1,↗,↘,⊥, and ⊤indicate the shape of the **input waveform** on the specified region.
- The label "all" gives the specified output regardless of the input.

Figure 5.2: Modified Waveform Response Graphs for N–C$^2$MOS and P–C$^2$MOS gates.



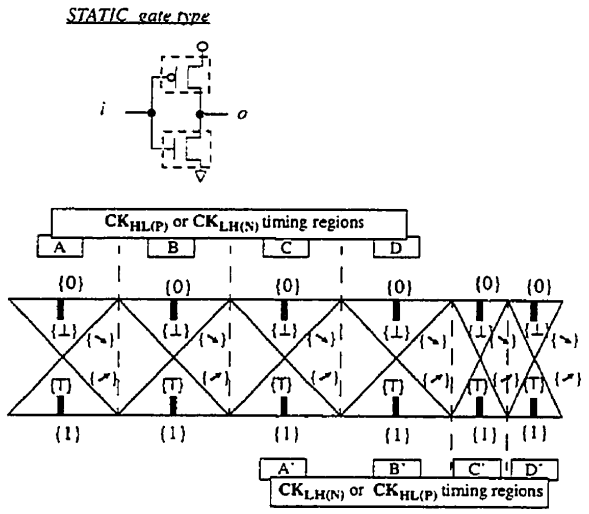

- These graphs apply only to inverters.
- The edges indicate the shape of the **output waveform.**
- The labels 0,1,↗,↘,⊥, and ⊤indicate the shape of the **input waveform** on the specified region.
- The label "all" gives the specified output regardless of the input.

Figure 5.3: Modified Waveform Response Graphs for N and P gates.




- These graphs apply only to inverters.
- The edges indicate the shape of the **output waveform.**
- The labels 0,1,↗,↘,⊥, and ⊤indicate the shape of the **input waveform** on the specified region.
- The label "all" gives the specified output regardless of the input.

Figure 5.4: Modified Waveform Response Graphs for All–N–L and All–P–L gates




- These graphs apply only to inverters.
- The edges indicate the shape of the **output waveform**.
- The labels 0,1,↗,↘,⊥, and ⊤indicate the shape of the **input waveform** on the specified region.
- The label "all" gives the specified output regardless of the input.

Figure 5.5: Modified Waveform Response Graph for Static gates.

## 5.3 Waveform Probabilities

The probability of finding a waveform shape at a node is found from the gate driving this node. The product of the input waveform probabilities of a gate indicate the probability of having a certain output by a vector of input waveforms, subsequently, finding the probability of a waveform requires adding the probabilities found by all vectors which effect the same output waveform.

**Example 10** *In the example of Figure 5.6 the two input* NAND *gate generates a* $\backslash$ *transient if one input is* $/$ *while the other is either 1 or* $/$*, this condition gives a total of three different vectors that generate the same output. The probability of this output is given by*

$$\begin{aligned} Pr(Out = \backslash) \;=\; & Pr(A = /)Pr(B = 1) + \\ & Pr(A = /)Pr(B = /) + \\ & Pr(A = 1)Pr(B = /) \end{aligned}$$

*If an even distribution of probabilities is assigned for the input waveforms,* $Pr(A = 0) = 1/4$*, etc., then according to Figure 5.6 the output waveform probabilities are* $Pr(Out = 0) = 1/16, Pr(Out = 1) = 9/16, Pr(Out = /) = 3/16$ *and* $Pr(Out = \backslash) = 3/16$

### 5.3.1 Buffer Example Using the Original Timing Model

The buffer example in Figure 5.7 is the same from Chapter 3. The difference between them is that waveform-probabilities have been assigned. Glitches are not considered for this example to keep the main concepts clear. In addition to the waveform probabilities the example also indicates the accumulated probability for each transient at nodes $I$ and $O$. These probabilities are found to compare them against the probabilities from Section 5.3.2 which are under the *simplified timing model.*

The probabilities for each region are found in Figure 5.7b by adding the probability of the waveforms which generate the same transient on the specified region. For example, the probability at the input node $I$ of having a $/$ transient in region A is the the same as the probability of having $Pr(I =$

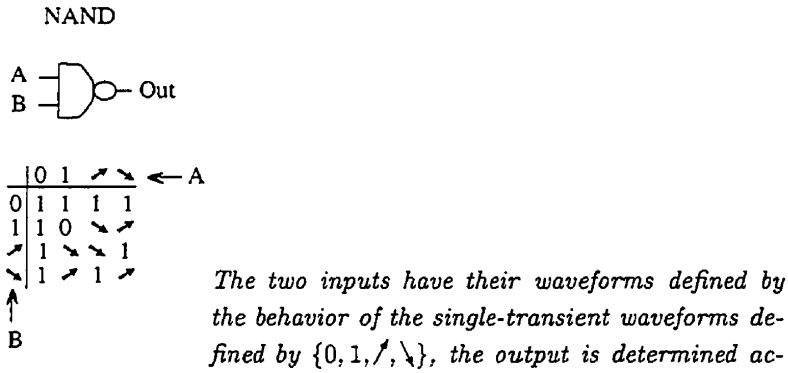
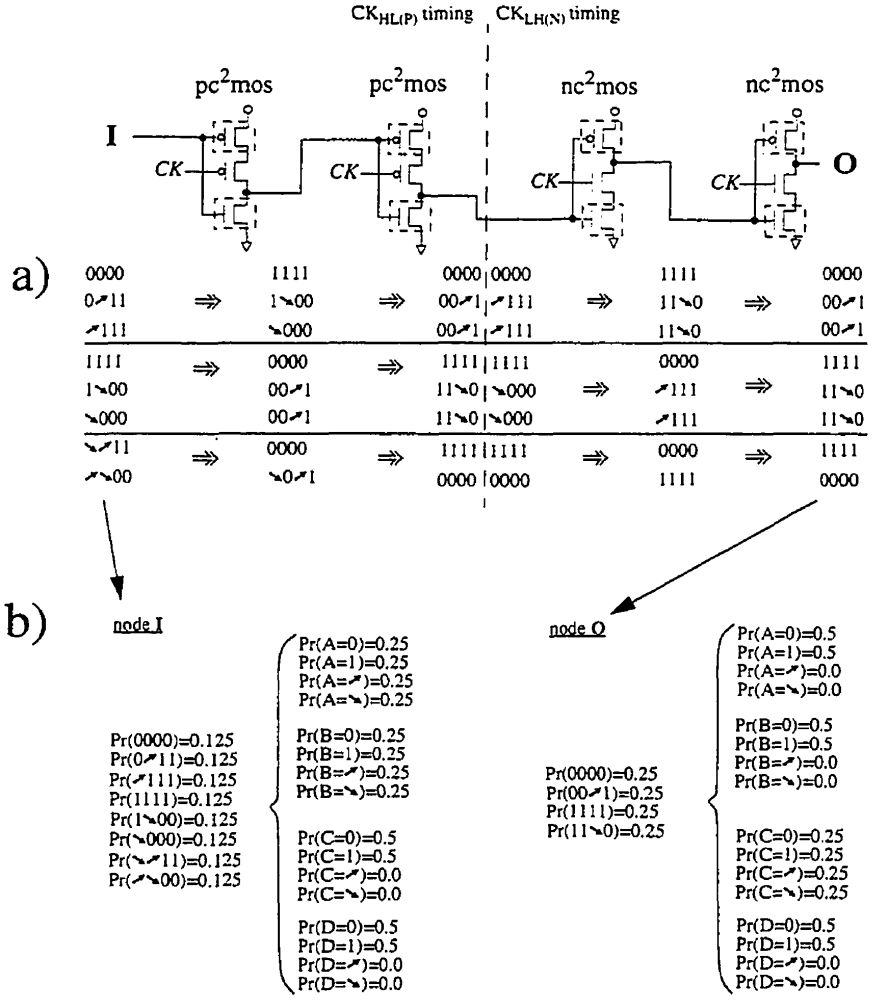



*Figure 5.6: Output behavior of a* NAND *gate*

/111) plus $Pr(I = /\backslash 00)$. The following equations were used to determine the probabilities for all transients in regions ABCD at node $I$,

$$
\begin{aligned}
Pr(A = 1) &= Pr(I = 1111) + Pr(I = 1\backslash 00) \\
Pr(A = 0) &= Pr(I = 0/11) + Pr(I = 0000) \\
Pr(A = /) &= Pr(I = /111) + Pr(I = /\backslash 00) \\
Pr(A = \backslash) &= Pr(I = \backslash 000) + Pr(I = \backslash/11) \\
Pr(B = 1) &= Pr(I = /111) + Pr(I = 1111) \\
Pr(B = 0) &= Pr(I = 0000) + Pr(I = \backslash 000) \\
Pr(B = /) &= Pr(I = 0/11) + Pr(I = \backslash/11) \\
Pr(B = \backslash) &= Pr(I = 1\backslash 00) + Pr(I = /\backslash 00) \\
Pr(C = 1) &= Pr(I = 0/11) + Pr(I = /111) + Pr(I = 1111) + Pr(I = \backslash/11) \\
Pr(C = 0) &= Pr(I = 1\backslash 00) + Pr(I = 0000) + Pr(I = \backslash 000) + Pr(I = /\backslash 00) \\
Pr(C = /) &= 0.0 \\
Pr(C = \backslash) &= 0.0 \\
Pr(D = 1) &= Pr(I = 0/11) + Pr(I = /111) + Pr(I = 1111) + Pr(I = \backslash/11) \\
Pr(D = 0) &= Pr(I = 1\backslash 00) + Pr(I = 0000) + Pr(I = \backslash 000) + Pr(I = /\backslash 00) \\
Pr(D = /) &= 0.0 \\
Pr(D = \backslash) &= 0.0
\end{aligned}
$$




(a) The waveform behavior present at node $I$ has eight waveforms, (b) and are assigned an equal distribution of probabilities. This behavior propagates one waveform at a time and thru the buffers, first P-C$^2$MOS and then N-C$^2$MOS, to node $O$. Therefore all vectors create the same output probability. A summary for the input and output nodes, with labels $I$ and $O$, adds the probabilities accordingly and shows the probability for each transient per region.

Figure 5.7: Probability propagation example under the "original timing model"

### 5.3.2 Buffer Example Using the Simplified Timing Model

If the simplified timing model is used instead, then the waveform behavior in $I$ is represented by two independent sets of waveforms. The buffer example has in the first set the waveforms within the AB regions and is equal to the following set

$$\text{AB} \in \{00, 0/, /1, 11, 1\backslash, \backslash 0, \backslash/, /\backslash\}$$

The second set which has the waveforms within the C and D regions are contained by the set

$$\text{CD} \in \{00, 11\}$$

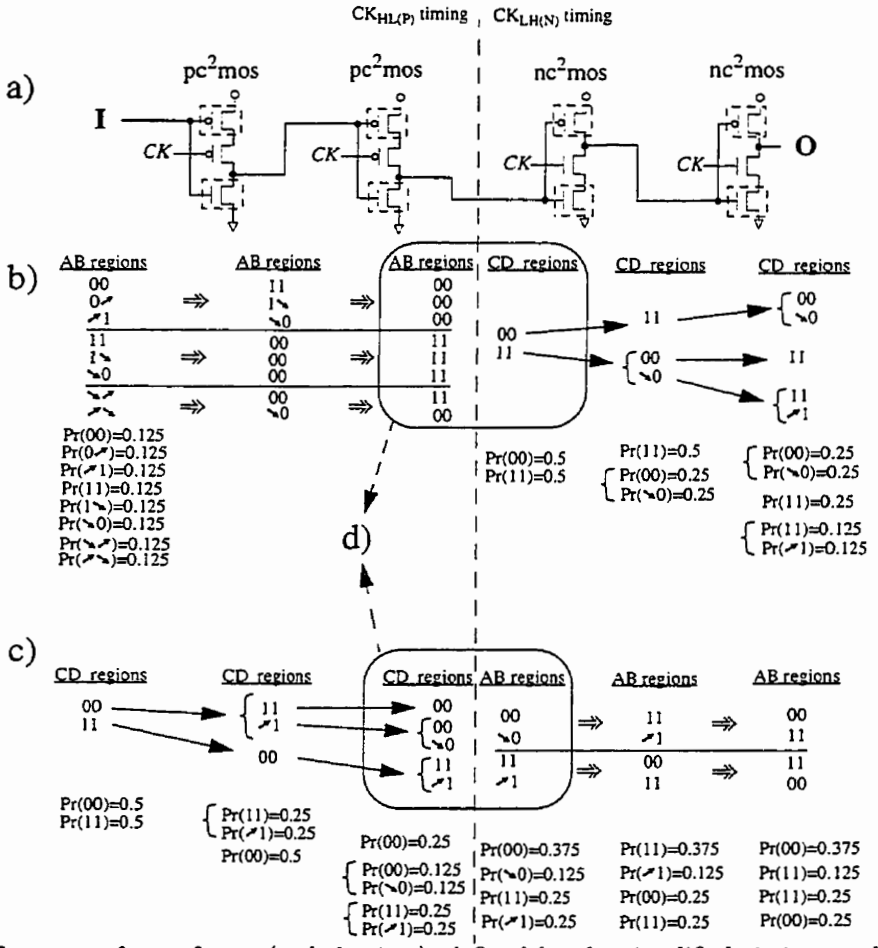
The propagation of probabilities for this simplified timing model uses the *waveform response graphs* exclusively for the specified region pairs, either AB or CD, and there is no more dependence between both. A number of unknown initial states will be found for waveforms within the CD regions, and when the example of Figure 5.7 is tested again in Figure 5.8 it uses the simplified timing model. Figure 5.8 shows that if a 0 or a / transient is present in the C region of a waveform feeding a P-C$^2$MOS buffer then the output has two possible waveform shapes.

The resulting probability is split evenly in this example. If a waveform at node $I$ defined within the CD regions is 00 and has a probability of 0.5 then it will be taken as generating the output 11 and /1 waveforms with a probability of 0.25 each.

The probability for each transient type is found in Figure 5.9. The difference with the probabilities found in Figure 5.7 is the cost of having a simplified model, where the final level during precharge is not linked to the initial level of the evaluate state.

#### Improved Probabilities

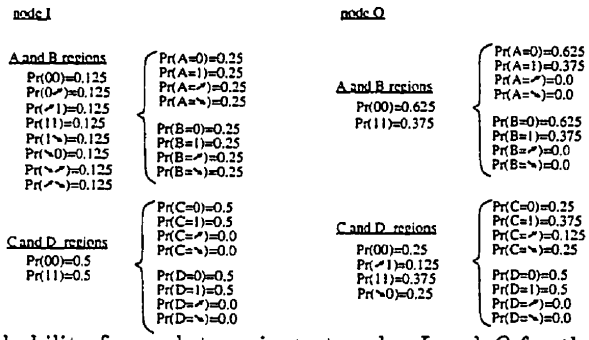
The example of Figure 5.8 introduces a maximum error of up to 50% because the lack of information regarding the initial output levels. This problem is diminished for waveforms within the CD regions by distributing the probabilities proportionally to the probability of the final level at the end of the B region.




a) Two sets of waveforms (or behaviors), defined by the simplified timing model, are propagated thru a series of buffers. b) The waveforms defined within the A and B regions are propagated, and each waveform creates just one possible output waveform.

This is different in c) where the input waveforms are first defined within the C and D regions. Because there is no information about the initial output levels the probability for each possible output is the even distribution from that of the input waveform. d) The crossing of timings from CKHL(P) to CKLH(N) is simply the waveform exchange between the set defined for the AB regions with those for the CD regions.

Figure 5.8: Probability propagation example under the "simplified timing model"




The probability for each transient at nodes $I$ and $O$ for the example in Figure 5.8 is obtained using the probabilities of individual transient regions similar to that for the example in Figure 5.7.

Figure 5.9: Transient probabilities for the input and output nodes

For example the probability of a VH level at the end of region B is equal to the added probability of the waveforms within the AB regions: $Pr(0\nearrow) + Pr(\nearrow 1) + Pr(11) + Pr(\searrow\nearrow)$

The buffer example is tested again in Figure 5.10. Consider the input waveform at node $I$ when it is 00, within the CD regions, which has a probability of 0.5 (see dashed arrow). This waveform can either generate a 11 or a $\nearrow 1$ waveform when driving the P–C$^2$MOS buffer. The probability for these waveforms is found from the probability of having a VH final level at the output:

$$Pr(\mathrm{B_{final}} = \mathrm{V_H}) = 0.125$$

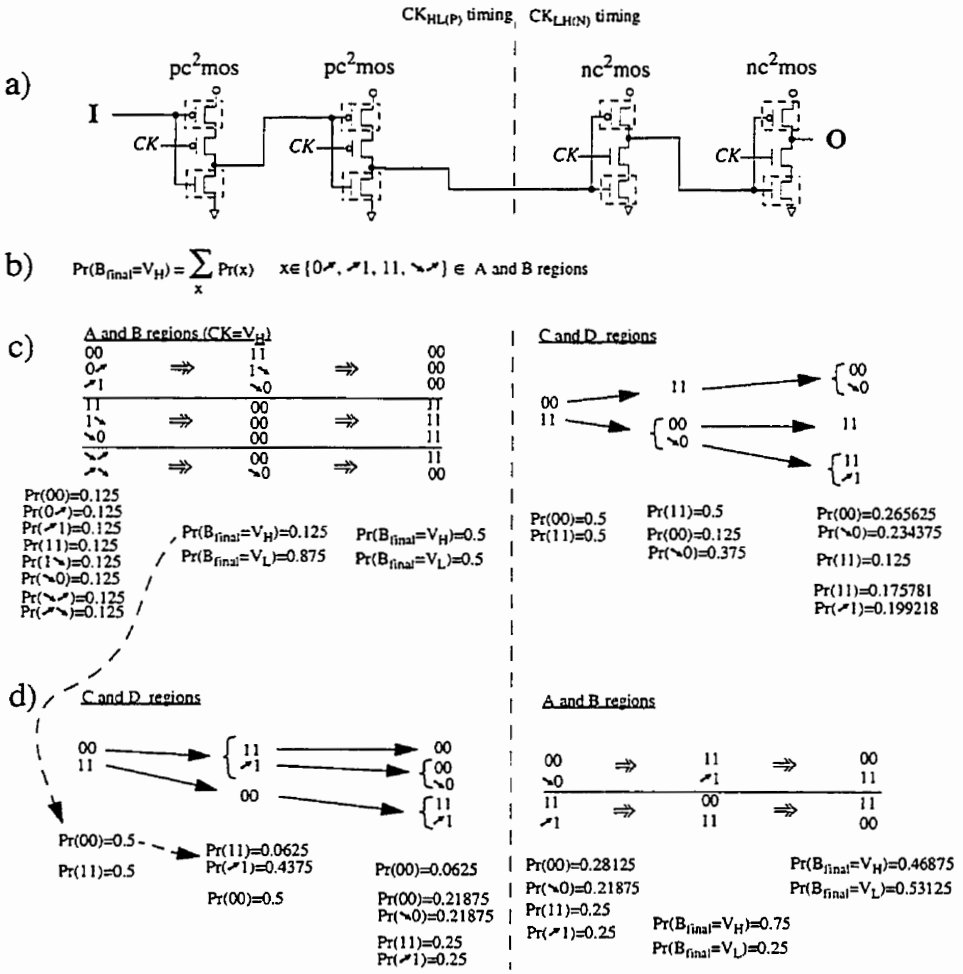
The probability of having the 11 output waveform is then

$$Pr(11) = Pr(I = 00) * Pr(B_{final} = \mathrm{V_H}) = 0.5 * 0.125 = 0.0625$$

and the $\nearrow 1$ waveform has a probability of

$$Pr(\nearrow 1) = Pr(I = 00) * Pr(B_{final} = \mathrm{V_L}) = 0.5 * 0.875 = 0.4375$$

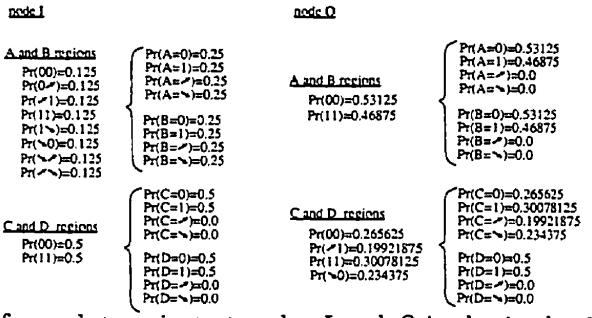
The transient probabilities are found once again in Figure 5.11, and it demonstrates how the simplified timing model can provide results within 6%




a) The simplified timing model is tested again for the buffer example. The difference here is the better distribution of probabilities whenever more than one waveform can be generated within the CD regions. b) The probability of the initial level before the beginning of the C region is found in c) and multiplied d) to the input probability.

Figure 5.10: Improved buffer example using the simplified timing model

to 20% of the more accurate *original timing model.*




The probability for each transient at nodes $I$ and $O$ is obtained using an equation set similar to those in Figure 5.7. These results have an error between 6% and 20% of those from the *original timing model.*

Figure 5.11: Transient probabilities for the improved buffer example

### 5.3.3 Complex Gates

The output waveform response for complex gates is found similarly to the method in Chapter 3. The additional transients described by the $\top$ and $\perp$ symbols are considered by evaluating the two input non-inverting logic gates of Figure 5.12. The procedure is the same as before, where a logic gate is first partitioned into non-inverting two input AND and OR gates. The resulting equivalent waveform behavior is then evaluated by a buffer of the corresponding gate type.

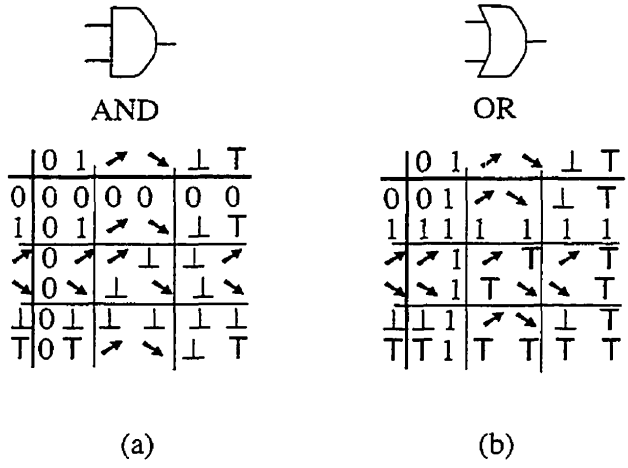
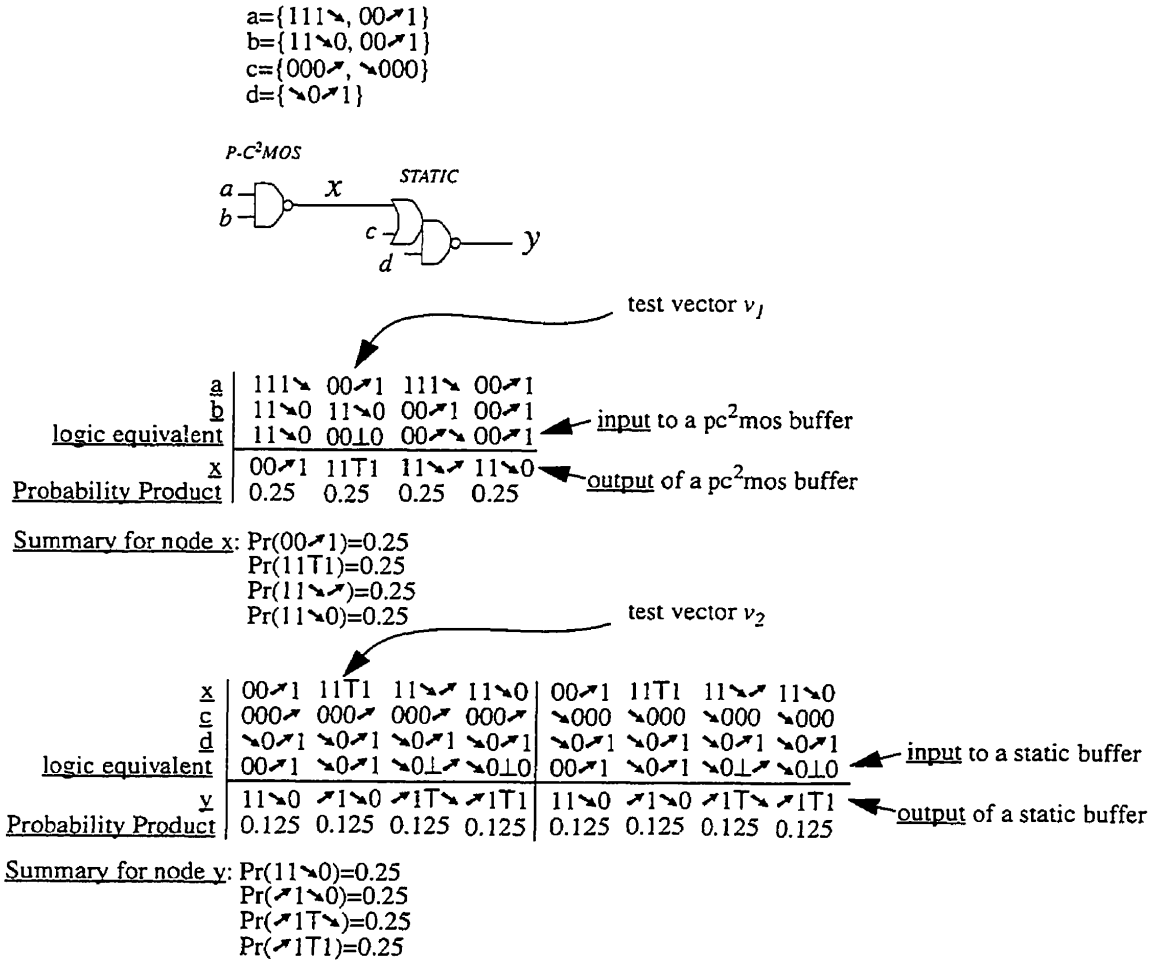



Figure 5.12: Logic Evaluation Using Transient Events

For example, two complex gates in Figure 5.13 have a limited input waveform behavior for illustration purposes, and have no glitches at their inputs. The evaluation of opposing transients by both gates creates, according to the test vector *v1* in Figure 5.13, a glitch whose shape is unknown and yet it is identified by the $\perp$ symbol.

According to the table in Figure 5.12 it is possible to eliminate glitches whenever a dominant logic value is evaluated, for instance an input of 0 for an AND function always outputs a 0. Similarly in Figure 5.13 the vector *v2* is a case where a raising transient is able to ignore a glitch.

The simplified timing model is used for complex gates without additional



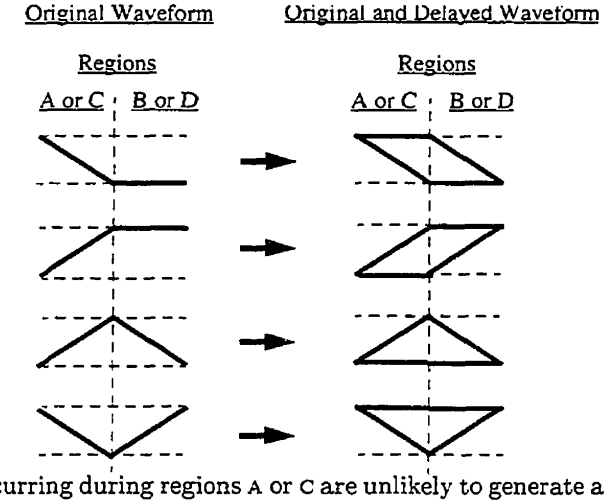

Complex gates are affected by glitches in this example. The complex gates evaluate the input waveforms to obtain a logic equivalent according to the tables in Figure 5.12. The probability of the input waveforms is distributed evenly, where $Pr(a = 111\backslash) = 0.5$, etc. The response due to the gate-type is found by evaluating the equivalent logic waveforms into a buffer of the desired gate type and its output is the result from using the waveform-response graphs presented for this chapter.

Figure 5.13: Complex gate example for glitches

changes, because glitches and differences due to the timing model are only important when evaluating the output buffer.

## 5.4 Transient Shifts due to Delays

The effect of a time delay from the input to the output node of a gate is considered by adding to the output behavior the extra waveforms created by a shift on the activity transients. Figure 5.14 shows which are the extra waveforms generated to account for gate delays.




A transient occurring during regions A or C are unlikely to generate a response within the time limits defining these regions. Slower transients are considered simply by adding into regions B and D the new shifted transients.

Figure 5.14: Delay Effects in a summarized Waveform Behavior

To incorporate this effect it is assumed that each new waveform has the same probability. However if a library of dynamic gates exists then the distribution of the waveform probability is dependent on the gate type and logic function.

The buffer example is tested for a single input waveform in Figure 5.15, and every time more than one waveform results due to delays then the probabilities are evenly distributed. The probability for each transient is given in Figure 5.16 and the results of this example reveal that the introduction of delay effects have redistributed the probability among those transients within the precharge period, still within regions AB, or among transients within the

evaluate period defined by the CD regions.

## 5.5 Power Formula

A power consumption estimate is presented in this section and it is based on the switching activity of nodes. This activity is found by the product of the clock frequency $f$ times the probability of a transition. For example, if the probability of having a $\backslash$ transient in region A is given by $Pr(\mathrm{A} = \backslash) = x$ then there are $fx$ transients per second of type $\backslash$, or in terms of power, the energy stored by the capacitive load at a node it is discharged $fx$ times per second.

The measured power consumption is restricted here to the dynamic case. The equation measuring power is derived from the energy consumption for a capacitor when charged by a CMOS transistor from 0 to $V$, assuming that the capacitor is always fully charged the energy is given by

$$E = \frac{1}{2}CV^2$$

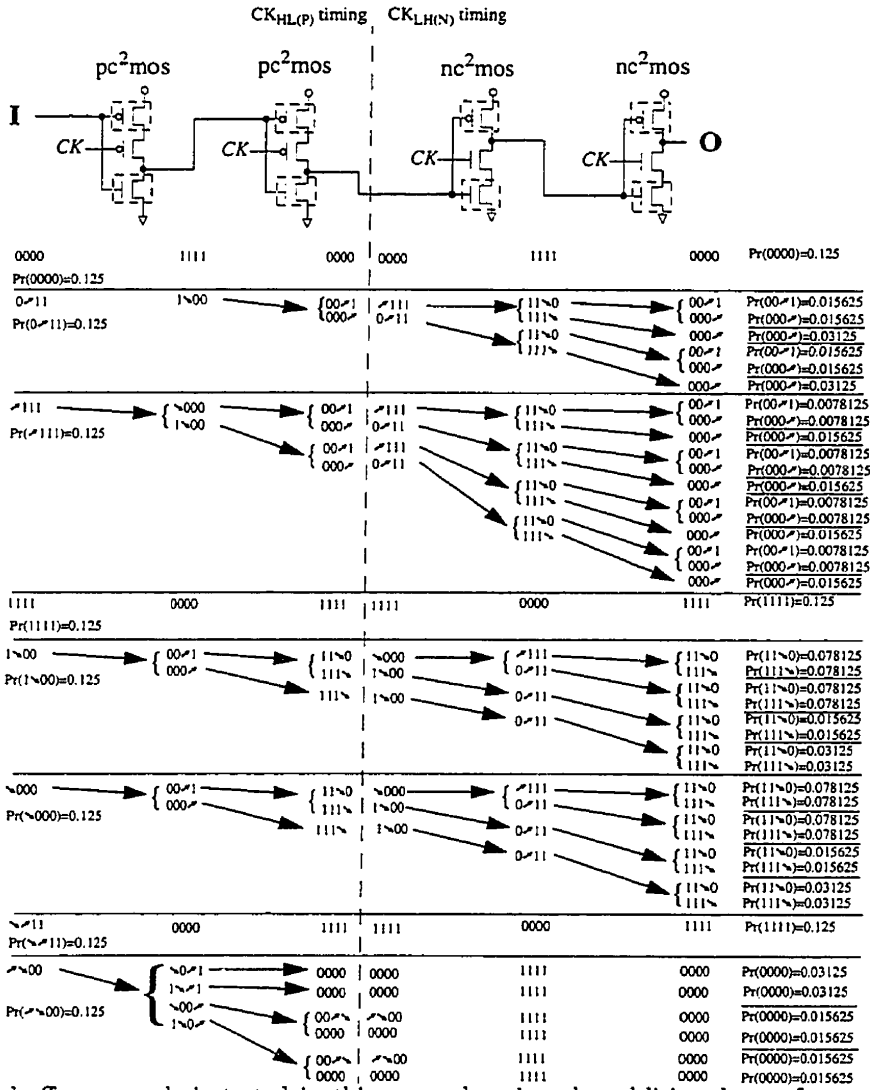
this formula is also the energy of a discharging capacitor. The estimated power consumption for a capacitor $i$ is then the product between the activity and the energy consumed by all transients, this power is given by

$$P_i = \frac{1}{2}fC_iV^2 \sum_x \sum_y Pr(x = y), \quad \begin{array}{l} x \in \{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}\}, \\ y \in \{/, \backslash\} \end{array}$$

The glitches presented by the extended timing model perform both transitions within the same region, therefore, every glitch accounts for the charge and discharge of the output load and their power consumption is given by

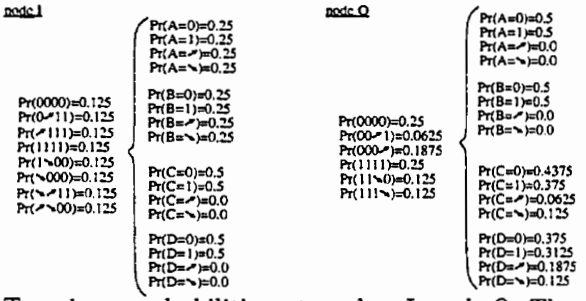
$$P_i = fC_iV^2 \sum_x \sum_y Pr(x = y), \quad \begin{array}{l} x \in \{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}\}, \\ y \in \{\top, \bot\} \end{array}$$

The clock signal is the only waveform for which no probabilities need to be estimated. A clock signal always charges and discharges its transistor




The buffer example is tested in this example using the additional waveforms due to the delay effect explained in Figure 5.14. The assumptions are that all delayed waveforms do occur and have an even probability distribution for all possibilities. If a 1\00 input waveform has a probability of 0.125 then the P–C$^2$MOS buffer creates a 00/1 and a 000/ waveform, each with a probability of 0.0625.

Figure 5.15: Buffer example with a delayed waveform behavior




Transient probabilities at nodes $I$ and $O$. The probability of ↗ and ↘ transients are redistributed within each state, either the precharge (AB regions) or the evaluate state (CD regions).

Figure 5.16: Transient distribution for buffers using delay effects

loads once per cycle. The power consumption of a single CLOCK device is

$$P_{clk} = f C_{clk} V^2$$

The overall power consumption is found by the addition of the dynamic power consumed by all gate nodes, including those driving the clock transistors. The power formulas according to each gate type is:

$$\begin{aligned}
Power(\text{Static}) &= \textstyle\sum_i \left[ (C_{i,N} + C_{i,P}) \left( \frac{1}{2} P_{i,tran} + P_{i,glitch} \right) \right] V^2 f \\
Power(\text{N}) &= \textstyle\sum_i \left[ C_{i,N} \left( \frac{1}{2} P_{i,tran} + P_{i,glitch} \right) \right] V^2 f + \\
&\quad (C_{clk,N} + C_{clk,P}) V^2 f \\
Power(\text{P}) &= \textstyle\sum_i \left[ C_{i,P} \left( \frac{1}{2} P_{i,tran} + P_{i,glitch} \right) \right] V^2 f + \\
&\quad (C_{clk,N} + C_{clk,P}) V^2 f \\
Power(\text{All-N-L}) &= \textstyle\sum_i \left[ C_{i,N} \left( \frac{1}{2} P_{i,tran} + P_{i,glitch} \right) \right] V^2 f + \\
&\quad (C_{clk,N} + C_{clk,P}) V^2 f + \\
&\quad (C_{fbk,N} + C_{fbk,P} + C_{dch,P}) \left( \frac{1}{2} P_{o,tran} + P_{o,glitch} \right) V^2 f \\
Power(\text{All-P-L}) &= \textstyle\sum_i \left[ C_{i,P} \left( \frac{1}{2} P_{i,tran} + P_{i,glitch} \right) \right] V^2 f + \\
&\quad (C_{clk,N} + C_{clk,P}) V^2 f + \\
&\quad (C_{fbk,N} + C_{fbk,P} + C_{dch,N}) \left( \frac{1}{2} P_{o,tran} + P_{o,glitch} \right) V^2 f \\
Power(\text{N-C}^2\text{MOS}) &= \textstyle\sum_i \left[ (C_{i,N} + C_{i,P}) \left( \frac{1}{2} P_{i,tran} + P_{i,glitch} \right) \right] V^2 f + \\
&\quad C_{clk,N} V^2 f \\
Power(\text{P-C}^2\text{MOS}) &= \textstyle\sum_i \left[ (C_{i,N} + C_{i,P}) \left( \frac{1}{2} P_{i,tran} + P_{i,glitch} \right) \right] V^2 f + \\
&\quad C_{clk,P} V^2 f
\end{aligned}$$

with the following definitions:

- $\sum_i$ is the indexed sum for the $n$ inputs of a gate where $i = 1..n$
- $P_{i,tran}$ and $P_{o,tran}$ are the addition of probabilities for single-transition transients, $i$ denotes an input node and $o$ is the output node of a gate: $P_{i,tran} = \sum_j \sum_k Pr(i, j = k)$ where $i = 1..n$, $j = \{\text{A}, \text{B}, \text{C}, \text{D}\}$ and $k = \{/, \backslash\}$
- $P_{i,glitch}$ and $P_{o,glitch}$ are the addition of probabilities for double-transition transients, $i$ denotes an input node and $o$ is the output node of a gate: $P_{i,glitch} = \sum_j \sum_k Pr(i, j = k)$ where $i = 1..n$, $j = \{\text{A}, \text{B}, \text{C}, \text{D}\}$ and $k = \{\bot, \top\}$
- $C_{i,N}$ and $C_{i,P}$ are the capacitance due to the $N$ and $P$ devices found at the input node $i$
- $C_{clk,N}$ and $C_{clk,P}$ are the capacitance loads due to the $N$ and $P$ clock devices.
- $C_{fbk,N}$, $C_{fbk,P}$, $C_{dch,N}$ and $C_{dch,P}$ are the capacitance loads due to the $N$ or $P$ devices found in the feedback inverter and discharge device of the All–N–L and All–P–L gates.

The All–N–L and All–P–L gates are improved by the optimizations suggested in Chapter 3. These optimizations modify the number of gate nodes and this is reflected with the following formulas. First, the optimized *N2-block* of the All–N–L and All–P–L gates:

$$\begin{aligned}
Power(\text{All–N–L}) &= \sum_i \left[C_{i,N} \left(\tfrac{1}{2} P_{i,tran} + P_{i,glitch}\right)\right] V^2 f + \\
&\quad (C_{clk,N} + C_{clk,P}) V^2 f + \\
&\quad (C_{fbk,N} + C_{dch,P}) \left(\tfrac{1}{2} P_{o,tran} + P_{o,glitch}\right) V^2 f \\
Power(\text{All–P–L}) &= \sum_i \left[C_{i,P} \left(\tfrac{1}{2} P_{i,tran} + P_{i,glitch}\right)\right] V^2 f + \\
&\quad (C_{clk,N} + C_{clk,P}) V^2 f + \\
&\quad (C_{fbk,P} + C_{dch,N}) \left(\tfrac{1}{2} P_{o,tran} + P_{o,glitch}\right) V^2 f
\end{aligned}$$

and the optimized All–N–L and All–P–L gates by the *revised N2-block*:

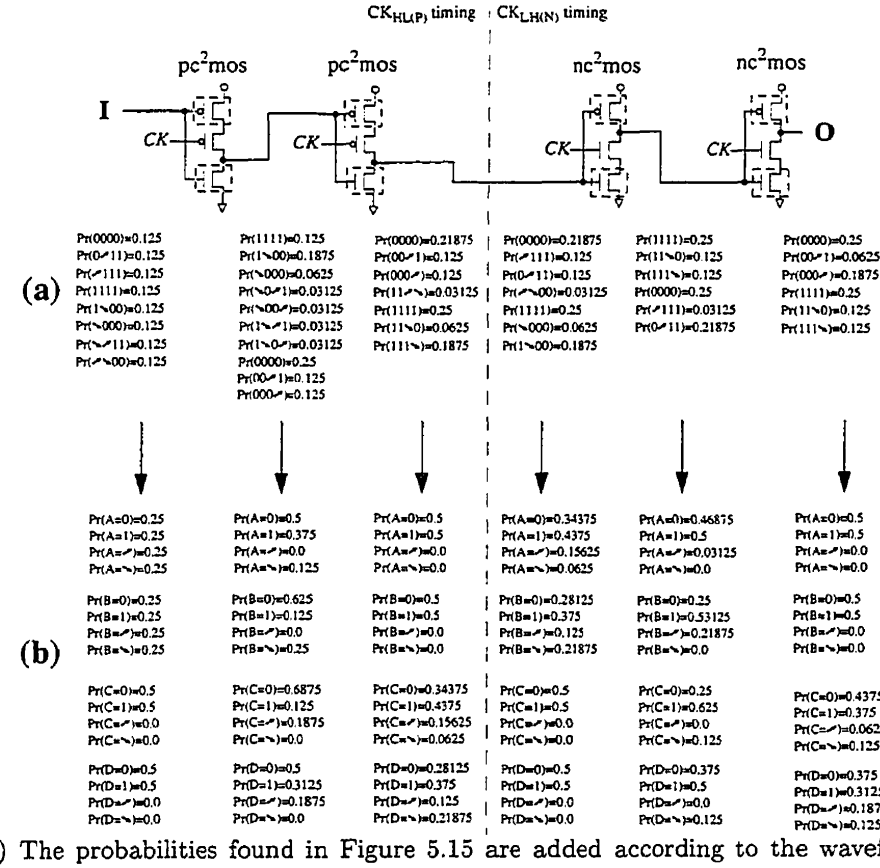
$$\begin{aligned} Power(\text{All–N–L}) &= \sum_i \left[C_{i,N}\left(\tfrac{1}{2}P_{i,tran}+P_{i,glitch}\right)\right]V^2f+ \\ &\quad (C_{clk,N}+C_{clk,P})V^2f+ \\ &\quad C_{dch,P}\left(\tfrac{1}{2}P_{o,tran}+P_{o,glitch}\right)V^2f \\ Power(\text{All–P–L}) &= \sum_i \left[C_{i,P}\left(\tfrac{1}{2}P_{i,tran}+P_{i,glitch}\right)\right]V^2f+ \\ &\quad (C_{clk,N}+C_{clk,P})V^2f+ \\ &\quad C_{dch,N}\left(\tfrac{1}{2}P_{o,tran}+P_{o,glitch}\right)V^2f \end{aligned}$$

In the following example the transient probabilities for the buffer in Figure 5.15 and Figure 5.16 are extracted for all nodes and summarized in Figure 5.18a. If the load for the output $O$ is a static inverter, and the circuit is tested in a technology where $L = 1.2\mu m, W_N = 2\mu m, W_P = 6\mu m, V = 3v, C = LW1.4mF$ and $f = 200MHz$ then according to the equation in Figure 5.17 the total power consumption is of $88.45\mu W$.

$$\begin{aligned} Power &= \frac{1}{2}(C_N+C_P)fV^2(0.25+0.25+0.25+0.25+0+0+0+0)+ \\ &\quad \frac{1}{2}(C_N+C_P)fV^2(0+0.125+0+0.25+0.1875+0+0.1875+0)+ \\ &\quad 2C_PfV^2+ \\ &\quad \frac{1}{2}(C_N+C_P)fV^2(0.15625+0.0625+0.125+0.21875+0+0+0+0)+ \\ &\quad \frac{1}{2}(C_N+C_P)fV^2(0.03125+0+0.21875+0+0+0.125+0+0.125)+ \\ &\quad 2C_NfV^2+ \\ &\quad \frac{1}{2}(C_N+C_P)fV^2(0+0+0+0+0.0625+0.125+0.1875+0.125) \\ &= 88.45\mu W \end{aligned}$$

The transient probabilities found in Figure 5.18 are the charge and discharge activity of the gate nodes. The total power is the addition of the dynamic power consumption at all gate nodes of the MOS transistors. Node $O$ is assumed to be loaded by a Static inverter.

Figure 5.17: Power estimate for buffer example



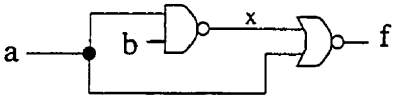

(a) The probabilities found in Figure 5.15 are added according to the waveform they indicate. (b) These probabilities are decomposed into the probability for each transition within the ABCD regions and for all nodes.

Figure 5.18: Transient activity for all nodes in the buffer example

## 5.6 Conditional Probabilities

The propagation of probabilities by means of their product at the input of a gate is a good estimate for the output response of a circuit. This is true as long as no dependencies exist between inputs. These dependencies are originated from either input signal constraints or from reconverging logic paths.

For example the input signals to the circuit in Figure 5.19 have no dependencies between them. However, input $a$ has a fanout to more than one gate, and so when the logic paths from $a$ meet again in gate $f$ then the dependence existing between $x$ and $a$ is such that the output of $f$ is always 0.




| a | b | x | f |
|---|---|---|---|
| 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |

A condition between the $x$ and $a$ inputs is generated such as the output $f$ is always 0. This result is obvious when the logic expression for the function for $f$ includes all logic gates, up to the source of the multiple signals, and becomes the logic function $\overline{\overline{(a\bar{b})} + a} = \overline{(\bar{a} + \bar{b} + a)} = 0$

Figure 5.19: Conditional signals created from independent inputs

The propagation of conditional probabilities is a complex procedure that has been recognized for a long time[AA75], and various algorithms have been developed to approximate their results without incurring large computational expenses. A similar method to that in the *cutting algorithm*[SDB84] has been selected to propagate the conditional probability of waveforms because it approximates its results whenever a conditional probability is created within the circuit, as is the case of the example in Figure 5.19.

### 5.6.1 Cutting Algorithm

The cutting algorithm handles the conditional probabilities of a combinatorial network by cutting reconvergent fanout branches. The cut points are assigned equivalent bounds and it is guaranteed that the probability of all computed bounds will enclose the true values.

The algorithm is executed in three steps and is explained in terms of binary values for the input signals:

- The signal probabilities are propagated up to those gates which have reconvergent paths. In Figure 5.20a the gates $W1$ and $W2$ are the propagation limits because these two gates are the meeting point for the fanout branches of node $a$.

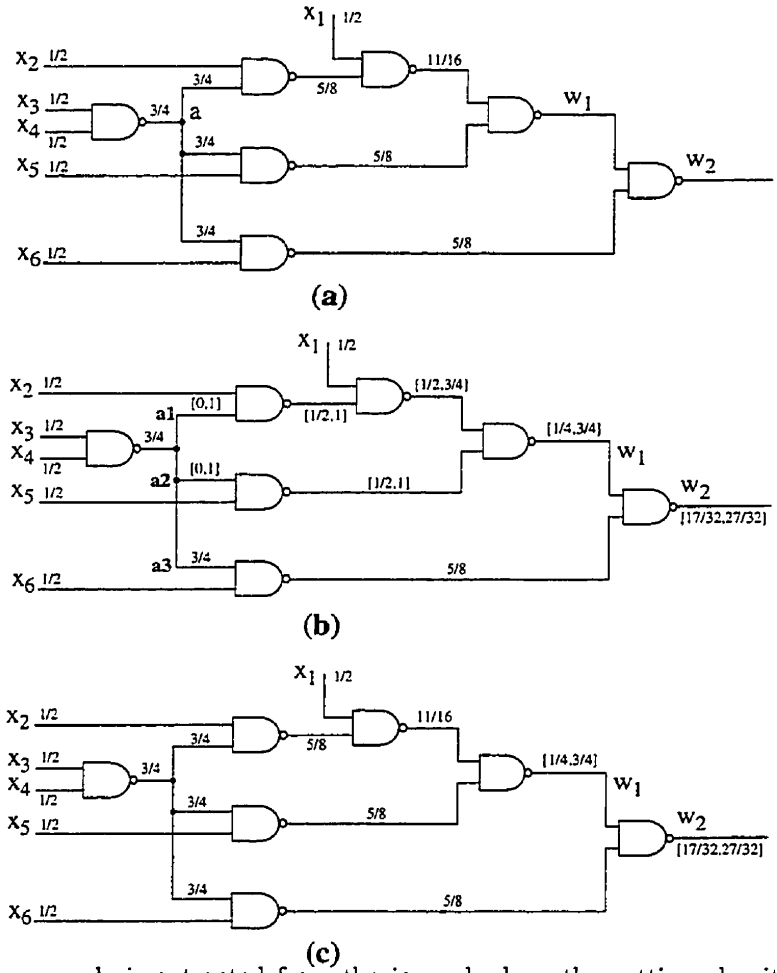
- The branches of a node with multiple fanout are separated and assigned a bound probability of $[0, 1]$, for both logic values, except for one branch. This is equivalent to restructuring the circuit into a tree-like structure where all signals are used just once. For example in Figure 5.20b the branches $a1$ and $a2$ are assigned a bound probability whereas branch $a3$ remains with the value of 3/4.

- The probabilities found on the first step are preserved and the unknown values are replaced with the bounds found in the second step. In Figure 5.20c the probabilities for nodes $W1$ and $W2$ are replaced with bound probabilities.

The formulas in Figure 5.21 by which the bound probabilities propagate are based on the decomposition of the circuit into logic AND, OR and NOT functions. These formulas represent the probability of a logic 1.

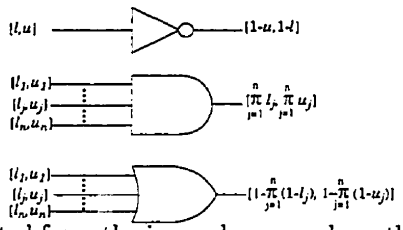
The origin of the formulas stems from two properties in binary logic gates. The logic operators AND and OR have what is called a dominant input level, and the addition of probabilities for binary signals obeys the constraint $Pr(1) = 1 - Pr(0)$, or $Pr(0) = [1 - u, 1 - l]$ for bound probabilities.

A dominant level is a logic 1 for OR gates and a 0 for AND gates. If a dominant level is found at one or more inputs then the output level is that of the dominant. For example the output of an AND gate is 1 if all input




This example is extracted from the journal where the cutting algorithm was presented [SDB84]. (a) Signal probabilities for tree-lines. (b) Signal probability for non-tree lines using formulas from Figure 5.21. (c) The final set of probabilities found by replacing in (a) the values for $W_1$ and $W_2$ with bound probabilities.

Figure 5.20: Cutting algorithm example



This figure is extracted from the journal paper where the cutting algorithm was presented [SDB84], and indicates the formulas used to propagate bound probabilities.

Figure 5.21: Propagation of probability bounds

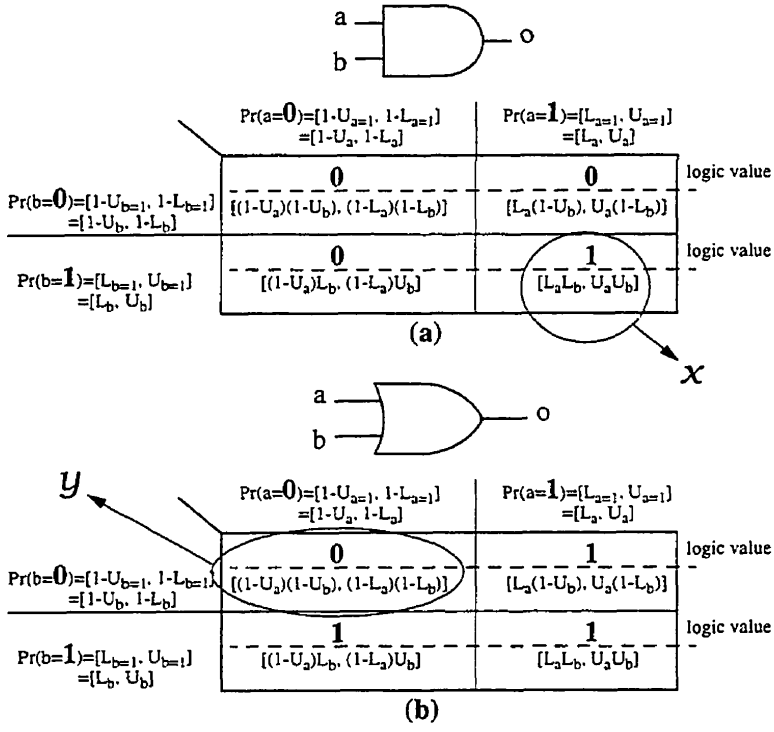
signals are 1, if one or more input is 0 it will dominate over the others and the output is 0.

The formulas to propagate bound probabilities are derived from the test vectors in Figure 5.22. Clearly, the formulas in Figure 5.21 were derived from the operations made in $x$ and $y$. The selection of these operations is the right choice because a single test vector is evaluated to get the right answer.

### Adapted Version of the Cutting Algorithm

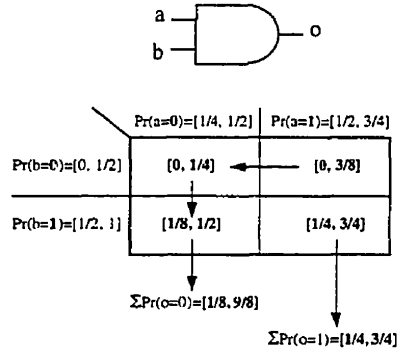
Finding the output probability using the other operations is, however, not possible by means of a simple addition. Because of the constraint requiring that all signal probabilities, when added, to be a total of 1.0. For example, in Figure 5.23 the input bound probabilities are $Pr(a = 1) = [1/2, 3/4]$ and $Pr(b = 1) = [1/2, 1]$. If the addition of probabilities is found for the test vectors using non-dominant levels then it is clear that it gives an incorrect, and out of limit, probability of $[1/8, 9/8]$. This implies that the addition of probabilities under the timing model for the new technique is likely to give bound probabilities beyond $[1, 1]$. Which is incorrect because it doesn't cover the true value.

For the purpose of comparing between versions of the same circuit. A full range of bound probabilities $[0, 1]$ is assigned to each node. These values are used in the next chapter to find the average and extreme worst cases for the waveform activity and power consumption.




(a) Bound probabilities for each test vector of an AND gate, and (b) the bound probabilities of an OR gate. The operations marked with $x$ and $y$ indicate which operations were used for the formulas in the cutting algorithm of Figure 5.21.

Figure 5.22: Detailed estimates for bound probablities




The formulas used to propagate bound probabilities in the cutting algorithm are based on the operations to determine a single test vector, this is when $a = 1$, and $b = 1$. Finding the same probabilities with the remaining test vectors is more complicated than a simple addition. Because as illustrated in this example, adding such probabilities yields the $[1/8, 9/8]$ bounds which make no sense.

Figure 5.23: Logic AND example with bound probabilities

# Chapter 6

# Examples using the Probability Model

This chapter presents two circuits which are transformed from a full Static Logic implementation to several versions where a mixture with Dynamic logic gates are tried to reduce power consumption. The procedure for this transformation is non-automated and it uses the Performance Model of Chapter 5. A computer program is used to find the activity estimates because of the large number of computations. This input for this computer program are the gate netlist and the input waveform probabilities. The output is the addition of all the dynamic power estimates found for each gate node of the MOS transistors.

The two benchmark examples are transformed to illustrate the flexibility gained by the analysis in Chapter 3. The logic behavior of the circuits is preserved as well as the logic structure. This is guaranteed as long as all gates maintain the waveform behavior predicted by the Waveform Response Graphs.

The activity information is propagated within the circuits and their power estimates are found. This amount of information is very limited since it ignores important phenomenon that might be critical, such as propagation delays and other forms of charge leakage. However this information is obtained very quickly and is a good estimate of what could be expected from the measured phenomena.

The performance model of Chapter 5 uses the "cutting algorithm" to estimate dependent probabilities. These estimates are given in the form of bound probabilities, therefore a minimum and a maximum are obtained. Bound probabilities are a mathematical range whose extreme values may never be reached in reality. For this reason, the average of the bound values is primarily used to compare circuits. The maximum is recorded with the intention of providing the numbers for a pessimistic estimate.

Glitches can be considered by the performance model but they were not considered for the optimization of the benchmark examples. This type of model should be related to a library of gates and their layout. Therefore the performance model used in this chapter is better suited to compare Dynamic Logic circuits. Furthermore, a performance estimate can show a Dynamic circuit that is better than a Static version of the same. In this case, the Dynamic version is a better choice because it has a potential for fewer glitches. Therefore, finding a Dynamic circuit which under the right conditions, such as these benchmarks, can perform better than a fully Static version indicates the existence of situations where Dynamic Logic is an acceptable alternative.

### Assumptions

The benchmark circuits are transformed under the following assumptions:

- TSPC registers are used in both the static and the dynamic versions. The reason is because it is a common practice to use dynamic TSPC registers to provide such function [Rab96].
- The benchmark circuits presented here are assumed to have their inputs supplied by non-dynamic logic circuits. This assumption forces to classify the input behaviors as external.
- Both benchmark circuits are allowed to relocate their registers from the input to the output.
- All input to output paths are assumed to have a latency of one clock cycle and it cannot be changed.

- All registers are assumed to be *positive edge-triggered.* Consequently the input behavior is *Ext-N* and the output behavior must be *Out-P*.
- The input behavior is assigned an even distribution of probabilities for each transient type—namely the 0, 1 / and \ transients.
- No delays or glitches are considered. Glitch propagation, and delay estimates, is a mechanism which provides better results when data from a library of cells is available.

### Circuit parameters

The benchmark circuits have the following parameters:

- The supply voltage is 3v.
- Transistor devices have the following widths,
  - Logic blocks: $W_n = 2\mu$, and $W_p/W_n = 3$
  - Clock devices: $W_{N,clk} = W_n$, and $W_{P,clk} = W_p$
  - Feedback and discharge devices for All–N–L/All–P–L gates are all equal to $W_n$
- The capacitance found at the gate node of a $W_n \times L$ device is $C_N = 3.36e - 15$.
- The fanout for all primary outputs is of one static inverter.
- The clock frequency is $f = 200MHz$.

Once a final Dynamic Logic version is found for a benchmark a timing comparison with the Static Logic version is provided. The delay values are arbitrary and are provided to illustrate what kind of timing can be expected.

The last section of this chapter is the summary. The power estimates are presented in a table to compare the best static and dynamic logic version. In addition, the summary provides the power estimates when the transient shift mechanism of Section 5.4 is applied and is presented under the "delayed" label.

## 6.1 Benchmark "cm150a"

LGSynth93 is the database from where the cm150a benchmark circuit was extracted. The circuit in Figure 6.1 was mapped in static logic using the SIS synthesizer [S+92], and some of its complex gates were broken into simple gates to illustrate a larger range of possibilities using dynamic logic.

The static logic version of the circuit in Figure 6.2 places the TSPC latches at the primary inputs. The total power consumption indicated in Figure 6.2 is the accumulated power for all gates and latches as obtained by the power measurement model described in Chapter 5.

Power estimates are obtained from the Cutting-Algorithm as bound estimates, there is a minimum and maximum, and the comparisons made for the test circuits are based on the average of the two as well as the maximum power consumption.

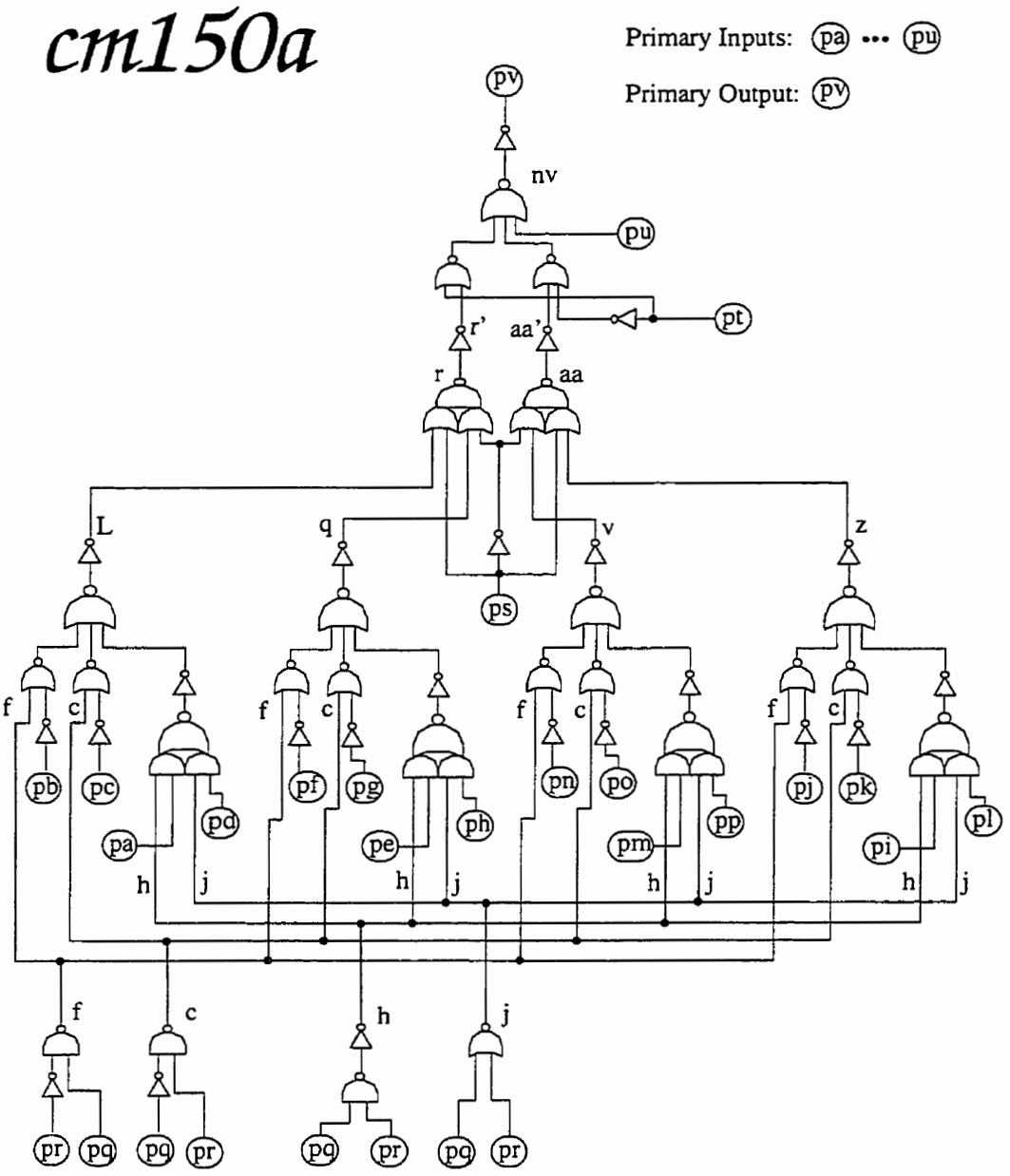



Figure 6.1: Test circuit: the cm150a benchmark without registers

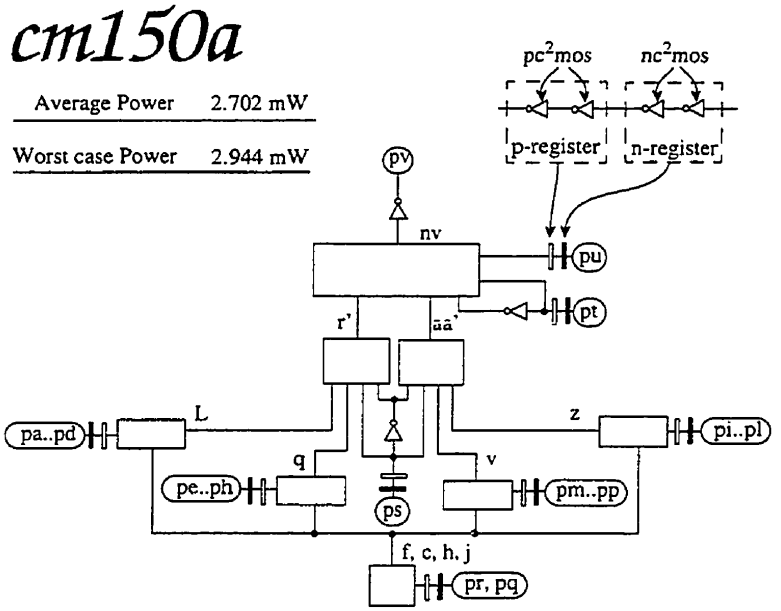


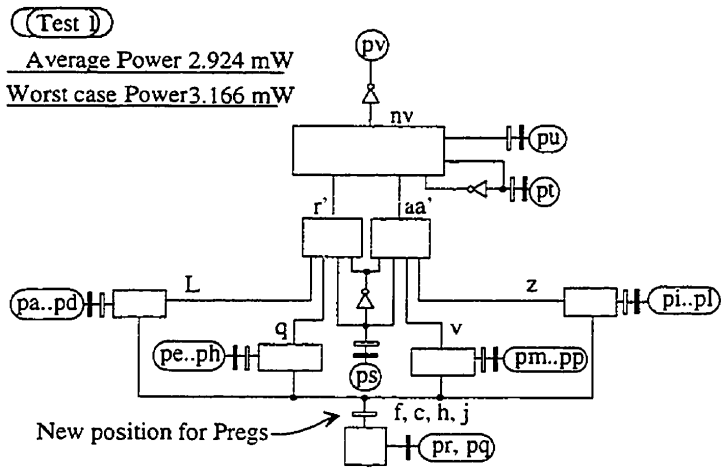

Figure 6.2: Static logic version of cm150a with all latches located at the primary inputs

### 6.1.1 Static version: Relocate P type latches

The first set of test circuits is the result of a search to relocate the P part of the TSPC latches. The first four test circuits in Figure 6.3 to Figure 6.6 relocate the P type latches from the input to the output.

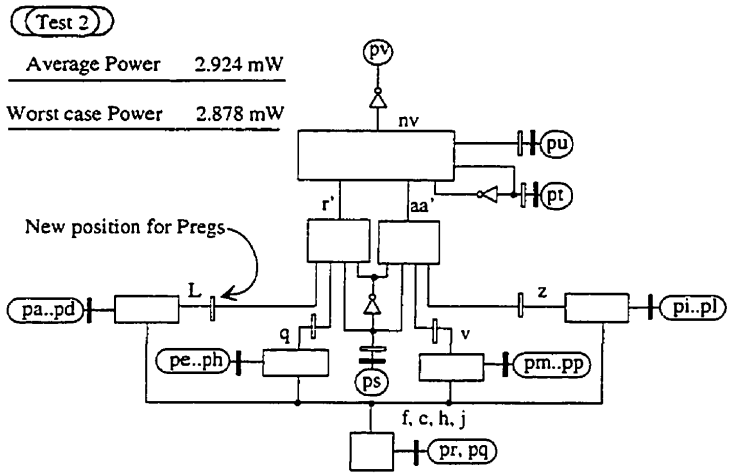
The difference of power consumption by these circuits is primarily influenced by the total capacitance because the probabilities for transitions among circuits is very similar. The reason for such probabilities is because all primary inputs are fed to N–C$^2$MOS gates and their ability to hold a state reduces the number of transitions. As a consequence the rest of the circuit propagates a small set of waveforms–this was observed before in Chapter 5.

Power consumption increases in test 1 because the number of latches has increased too, and the further the P latches are pushed towards the output the fewer latches are required. The best performing circuit is in test 4 (Figure 6.6).



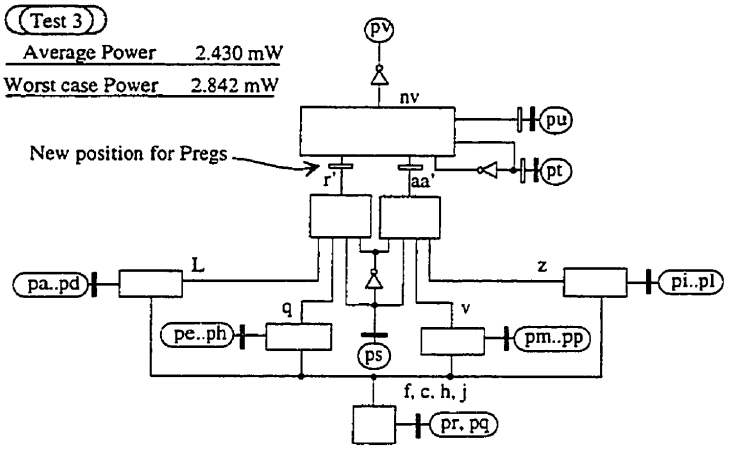

Static logic version of cm150a, P type latches are relocated to the output of *f, c, h and j*

Figure 6.3: Static Version: Test 1



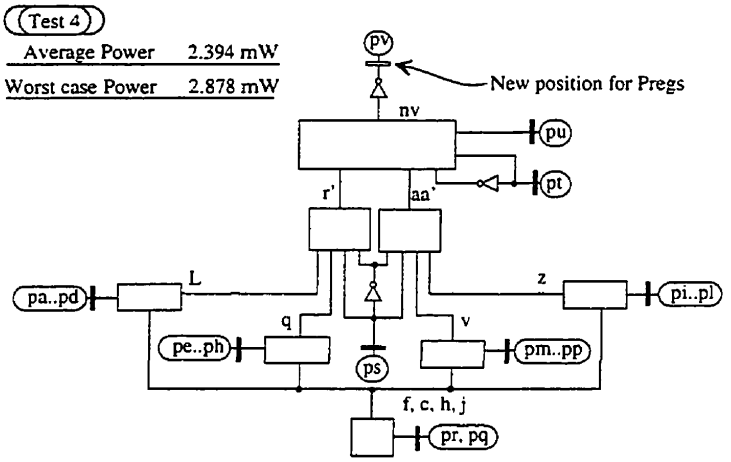

Static logic version of cm150a, P type latches are relocated to the output of *L, q, v and z*

Figure 6.4: Static Version: Test 2




Static logic version of cm150a, P type latches are relocated to the output of *r' and aa'*

Figure 6.5: Static Version: Test 3




Static logic version of cm150a, P type latches are relocated to the output of *pv*

Figure 6.6: Static Version: Test 4

### 6.1.2 Static version: Relocate N type latches

The second set of test circuits relocates the N type latches using the circuit for test 4 in Figure 6.6. In the test circuit set from Figure 6.8 to Figure 6.11, the further the N latches are pushed the more gates are left with external input behaviors, and they contribute to a higher power consumption. However, the capacitance is also reduced by the smaller number of latches hence reducing the total power from one circuit to another.

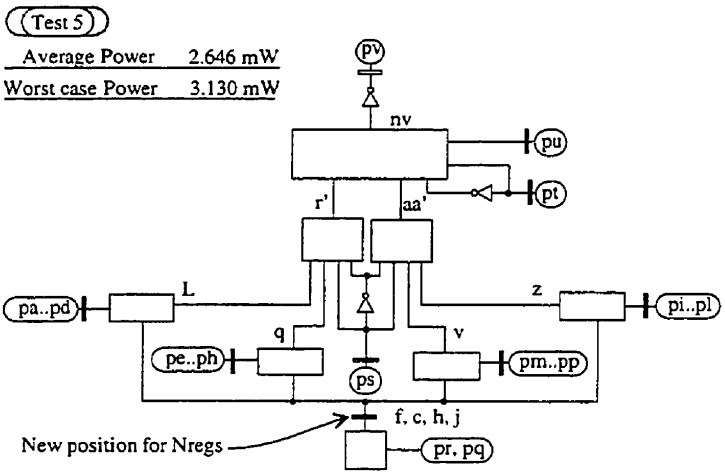
The table in Figure 6.7 summarizes the power estimates and the number of latches for each test circuit. The number of latches has been included to show that the total capacitance cannot be used to find the best performing circuit. In this benchmark, the smallest circuit is found in test 8, however such a circuit is in fact the worst case for the *maximum power* column.

The circuit in test 4 is the reference from where to compare against the dynamic logic version of the cm150a benchmark. Although the circuit in test 3 shows the lowest power in the *maximum power* column the circuit in test 4 is chosen.

| test | Power consumption | | Number of Latches | |
|---|---|---|---|---|
| | average | maximum | N-C$^2$MOS | P-C$^2$MOS |
| 0 | 2.702 mW | 2.994 mw | 21 | 21 |
| 1 | 2.924 mW | 3.166 mW | 21 | 23 |
| 2 | 2.492 mW | 2.878 mW | 21 | 7 |
| 3 | 2.430 mW | 2.842 mW | 21 | 4 |
| 4 | 2.394 mW | 2.878 mW | 21 | 1 |
| 5 | 2.646 mW | 3.130 mW | 23 | 1 |
| 6 | 2.634 mW | 3.588 mW | 7 | 1 |
| 7 | 2.624 mW | 3.616 mW | 4 | 1 |
| 8 | 2.592 mW | 3.620 mW | 1 | 1 |

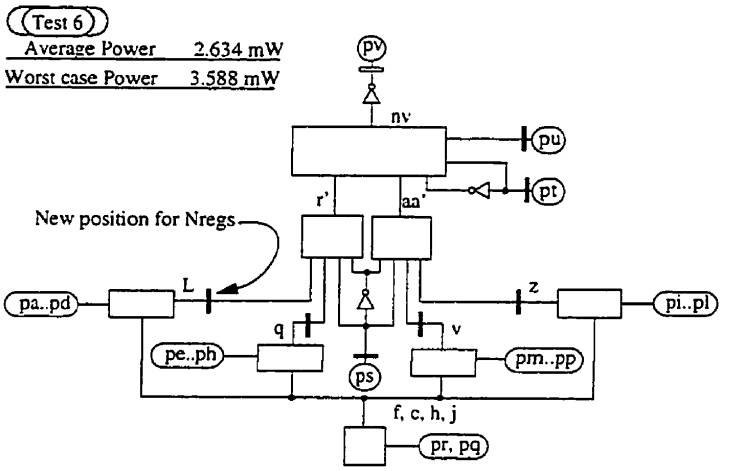
Power consumption for various static logic versions of the benchmark circuit *cm150a*.

Figure 6.7: Summary of tests 1 to 8.



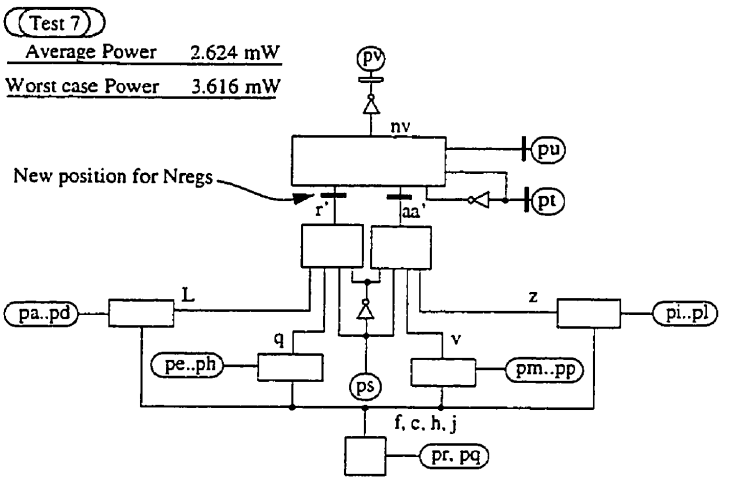

Static logic version of cm150a, N type latches are relocated to the output of *f, c, h and j*

Figure 6.8: Static Version: Test 5



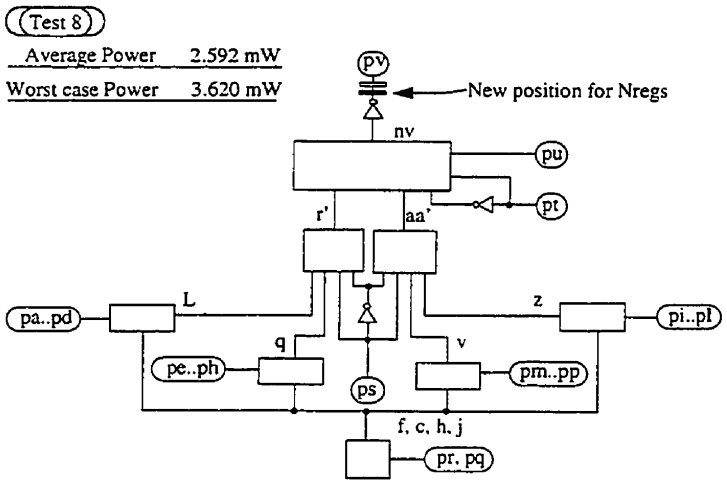

Static logic version of cm150a, N type latches are relocated to the output of *L, q, v, and z*

Figure 6.9: Static Version: Test 6




Static logic version of cm150a, N type latches are relocated to the output of *r' and aa'*

Figure 6.10: Static Version: Test 7




Static logic version of cm150a, N type latches are relocated to the output of *pv*

Figure 6.11: Static Version: Test 8

### 6.1.3 Dynamic Version: Introduce P Latches into the Circuit

The gate types defined for the new technique are able to perform complex logic functions regardless of their location within a circuit. This could inherently lead to a lower power consumption because there is no need to insert gates for the sole purpose of performing the latch function. However, such statement cannot be considered a rule of thumb as will be demonstrated by the following tests.

The test circuit number 4 of Figure 6.6 is the reference circuit from where a dynamic logic version will be built. This circuit is selected because it places the P latch at the primary output; a location that is the most efficient place for the P latch according to the results in test 4 and 8, described in Figure 6.6 and Figure 6.11. The location for the N latches is appropriate because it affects a large portion of the logic.

The test circuits in Figure 6.12 introduce the P latches into the logic and modify gates NV and PV to be driven by a CKHL(P) timing. The input to NV is described by behavior *Out-N*, which is the output generated by gates under the opposite CKLH(N) timing. The input behaviors to the gates in the CKHL(P) timing must be compatible and therefore their input behavior is *crossed* from a CKHL(P) to a CKLH(N) timing denoted by "(*Out-N*)".

The rules indicate that the behavior "(*Out-N*)" can drive the gate types specified in the set *Border-P* $\equiv$ {P, All-N-L, P-C$^2$MOS}, from this set only the P and P-C$^2$MOS gate types are tested. The All-N-L gate type is not tested because this is a non-inverting gate and it would require two more inverters: one for bringing the output behavior to be *Out-P* and another inverter to set the right logic output.

Finally, in Figure 6.12 the output behavior is generated by a P-C$^2$MOS gate because this is the only gate type specified in the set *Dyn-Out-P* of the rules.

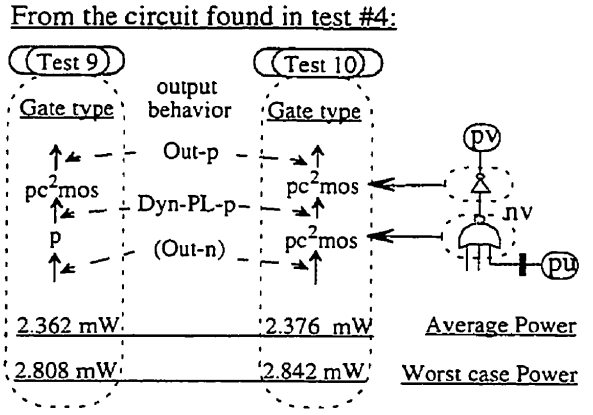



Figure 6.12: Dynamic Version: Tests 9 and 10

### 6.1.4 Dynamic Version: Introduce N Latches into the Circuit

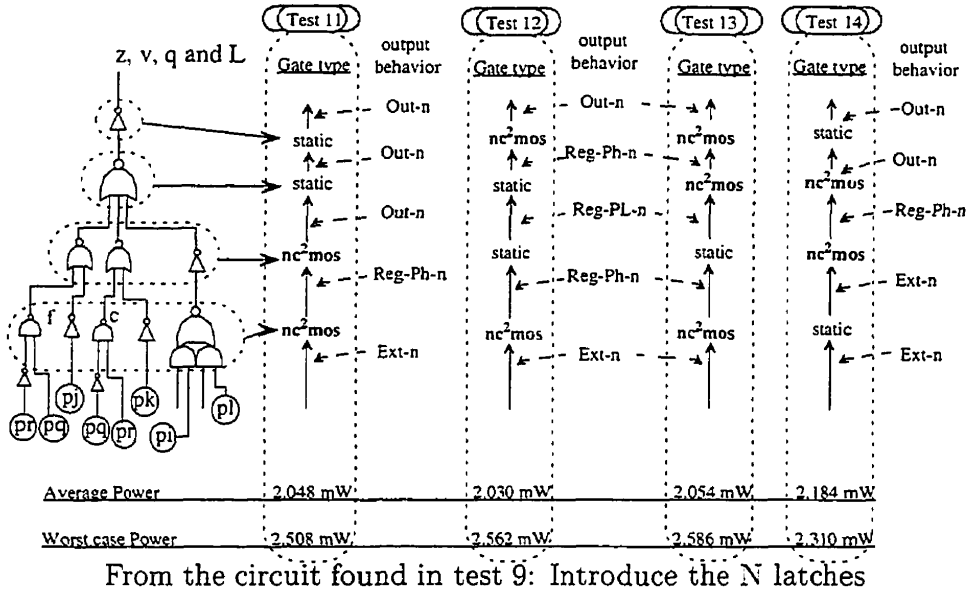
The circuit found for test 9 is the starting point from which the N type latches are introduced to the circuit in Figure 6.13. The input behavior to the N type latches is *Ext-N* and their output is *Out-N*. With this behavior requirements, an inspection of the new technique shows that there should be an N–C$^2$MOS gate followed by either:

- An N–C$^2$MOS gate which has a *Reg-PH-N* input behavior and a *Out-N* output behavior, or
- One or more structures formed by a static gate followed by either another static gate or an N–C$^2$MOS gate, with input and output behaviors of *Reg-PH-N*. This structure is finally connected to an N–C$^2$MOS which has an output behavior of *Out-N*.

In any case, both items indicate a structure with an even number of gates from input to output.

The largest subcircuit for introducing the N latches in Figure 6.13 covers gates $z, v, q$, and $L$ and reaches up to four levels deep towards the inputs.

The test circuits number 11, 12, 13 and 14 have all possible arrangements of dynamic gates according to the new technique as described by the previous two items.




From the circuit found in test 9: Introduce the N latches into the circuit. Input behaviors are *Ext-N* and the output is *Out-N*. These test circuits explore all possibilities allowed by the new technique under such conditions.

Figure 6.13: Dynamic Version: Tests 11, 12, 13 and 14

The extent of the pipes due to tests 11, 12 and 9 is illustrated in Figure 6.14.

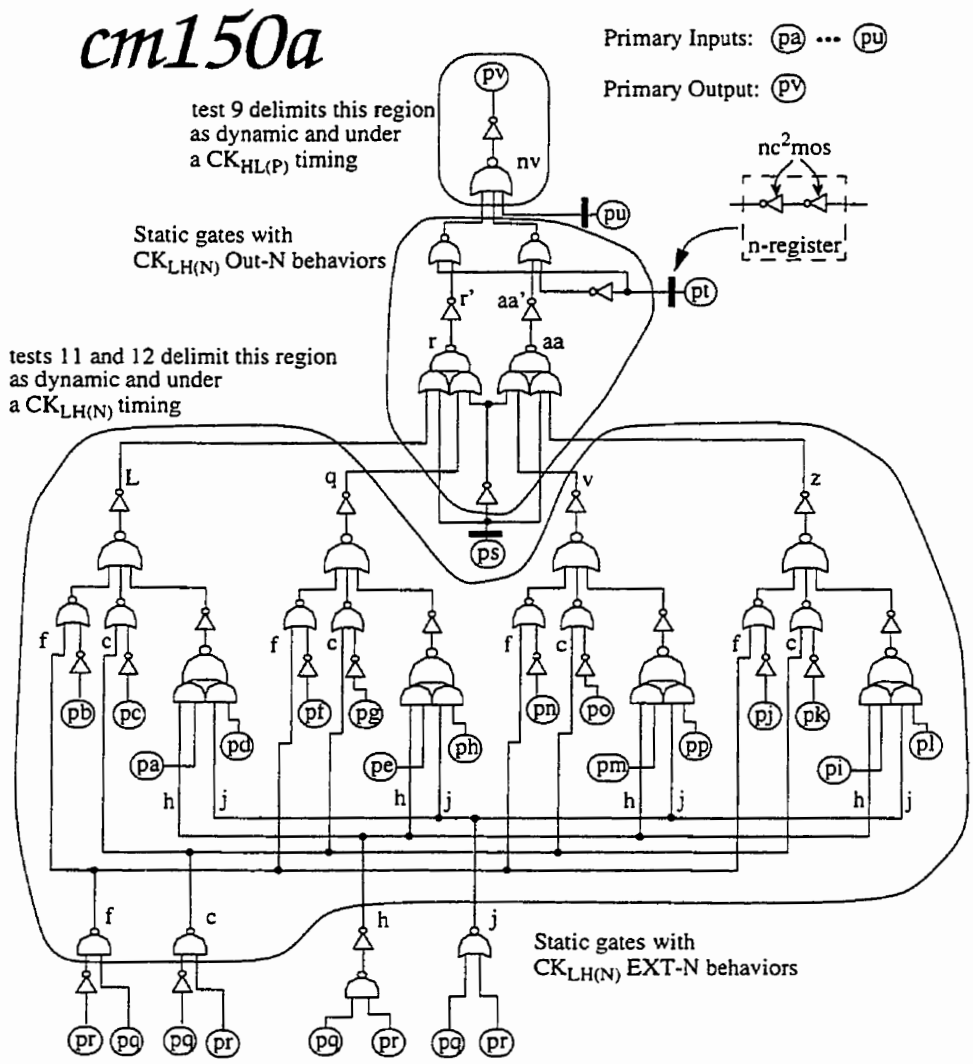



Figure 6.14: cm150a version from tests 11, 12 and 9

### 6.1.5 Dynamic Version: Insert Dynamic Gates into the CKhl(p) Timing Region

The best performing circuits in Figure 6.13 are tests 11 and 12 and are used to explore further alternatives for dynamic logic. In Figure 6.15, the logic gates $r', aa', r$ *and* $aa$ provide an even number of gates from the output of $z, q, v$ *and* $L$ to the output of $r'$ *and* $aa'$. The output behavior from $r'$ *and* $a'$ will be propagated up to gate NV where it's mixed with the "(*Out-N*)" behavior of *PU*'s N latch. NV's output is now different from what was expected before in test 11, and for this reason it is no longer able to drive a P type dynamic gate. The circuit in Figure 6.15 with test label number 15, is the same circuit from test 11 but it replaces the gate type for NV from P to P–C$^2$MOS.

#### Test circuits from test 11 and 15

The following test circuits 16 to 21 explore all the alternatives provided by the set *Border-P* $\equiv$ {P, All–N–L, P–C$^2$MOS} for gates $aa$ and $r$. These are the behaviors allowed when entering a CKHL(P) timing.

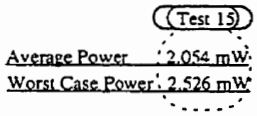
The $r'$ and $aa'$ gates have the input behavior of *Dyn-PL-P* and have no restriction on the output behavior. The reason for this freedom is because the P–C$^2$MOS gates used for NV and PV can have any input behavior for NV and still provide at the output of PV the required *Out-P* behavior. This effect was also observed by the developers of the TSPC technique who were aiming to eliminate the odd/even inversion constraint set by the NORA technique.

As a consequence of such freedom the feasible gate types for $r'$ and $aa'$ are in the set {*Dyn-PL-PH-P* $\cup$ *Dyn-Out-P*} $\equiv$ {Static, L/H, P–C$^2$MOS}.

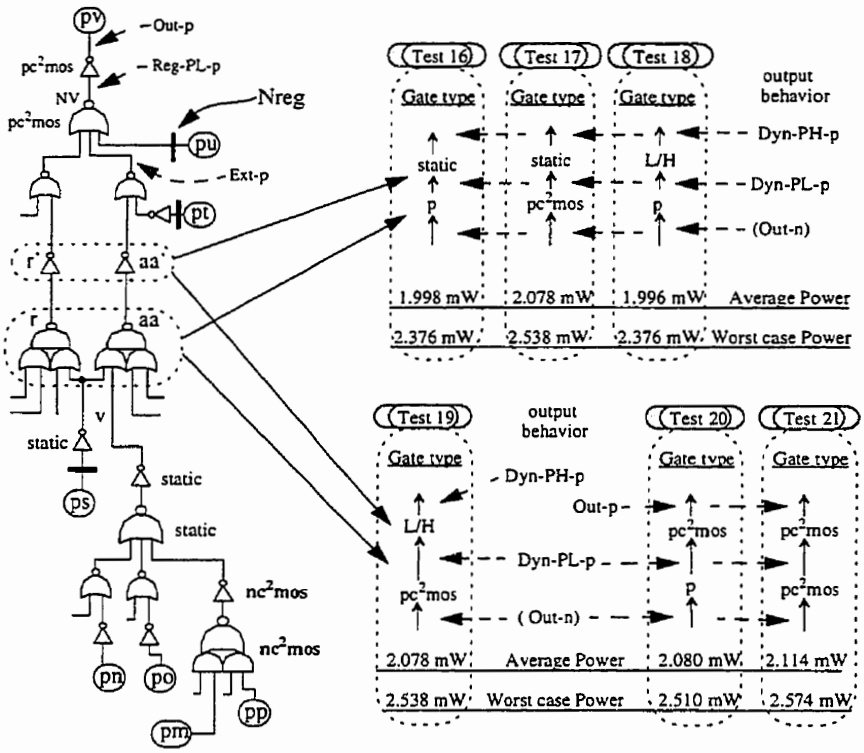
Another possible test derived from the circuit in test 15 uses the set *Dyn-PL-PL-P* $\equiv$ {All–N–L}. The non-inverting function of an All–N–L gate requires some changes to the circuit structure to allow for the non-inverting versions of gates $r$ and $aa$. The test circuit number 22 in Figure 6.16 uses All–N–L gates and it has the best power performance.

From the circuit in test #11:

(change NV to pc$^2$mos type because it will bring Ext-p behaviors back to Out-p at the output of PV)

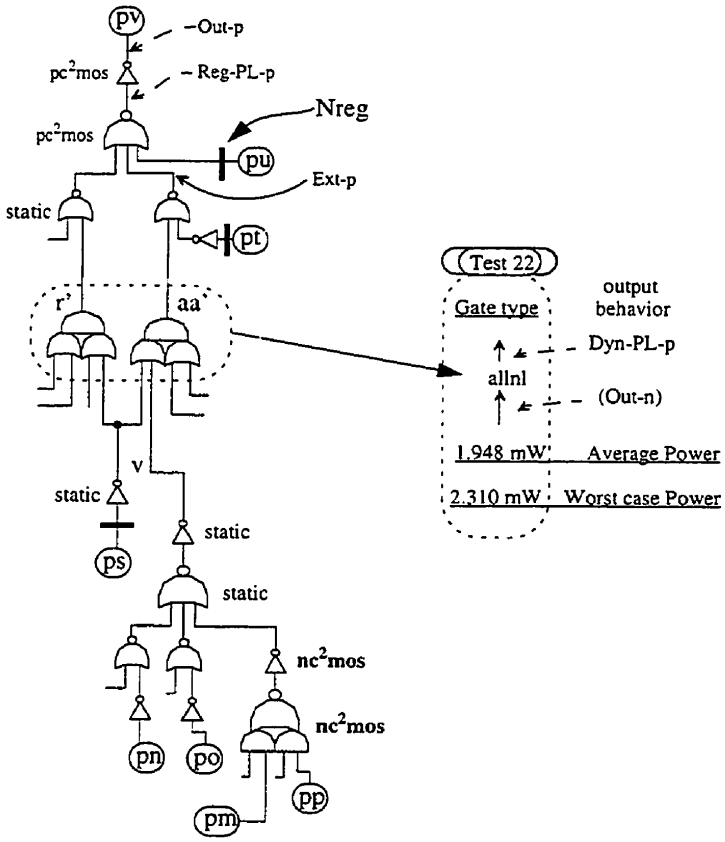



From the circuit in test #15:




From the circuit found in test 11: Insert more dynamic gates into the CKHL(P) section.

Figure 6.15: Dynamic Version: Tests 15 to 21.

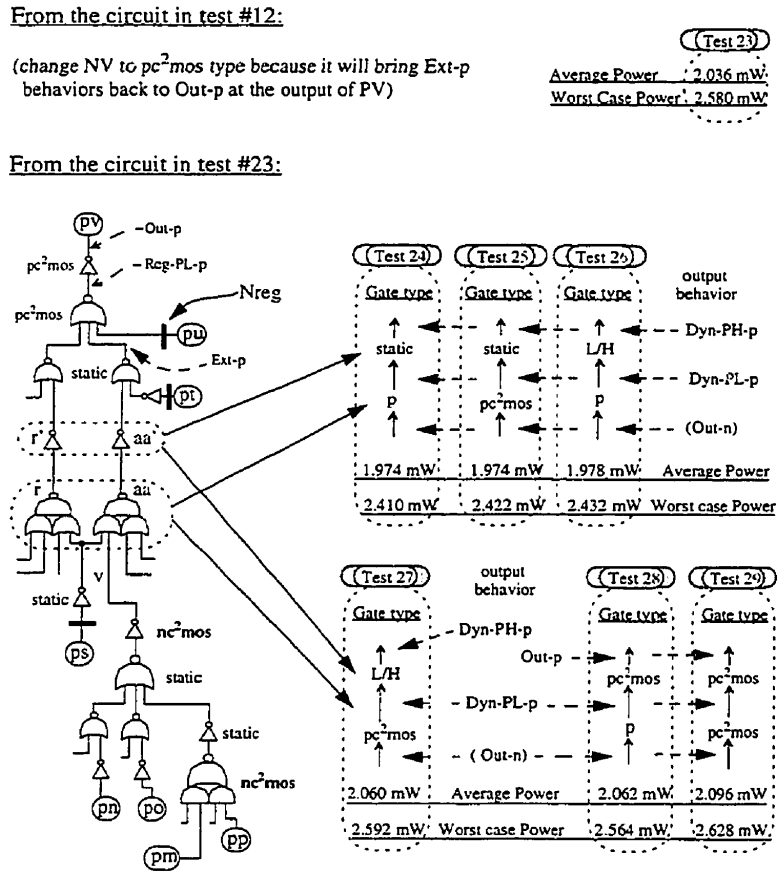



From the circuit in test 15: Insert non-inverting gates into the CKHL(P) section.

Figure 6.16: Dynamic Version: Test 22.

### Test circuits from test 12

The same search is now indicated by Figure 6.15 and Figure 6.16 but based on the circuit found for test 12. First, in Figure 6.17 the test circuit number 23 replaces the NV gate, in test 12, by a P–C$^2$MOS type because together NV and PV become a structure accepting any behavior within the CKHL(P) timing, and provides an output behavior of *Out-P*.

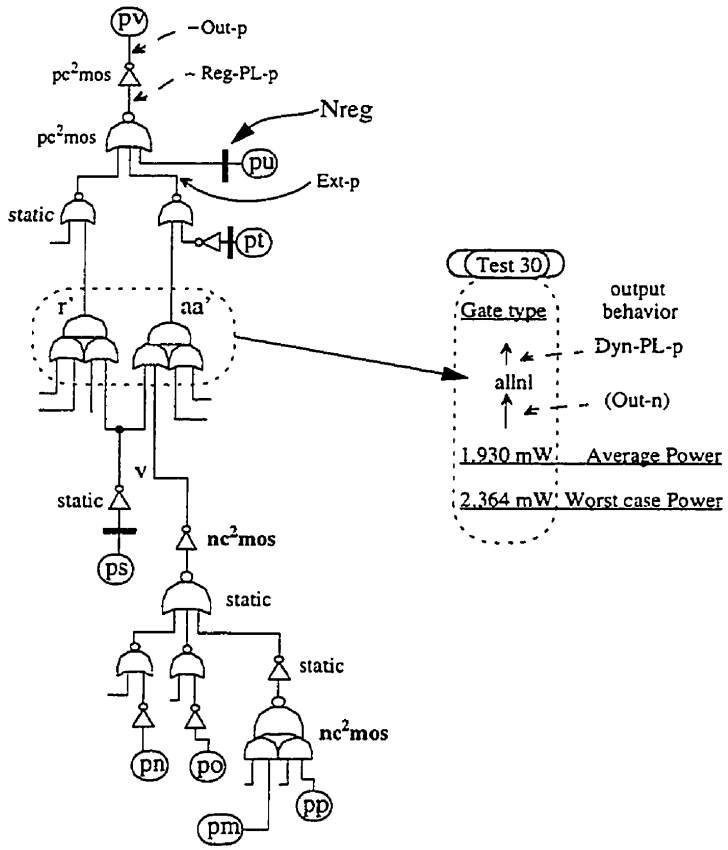



From the circuit found in test 23: Insert more dynamic gates into the CKHL(P) section.

Figure 6.17: Dynamic Version: Tests 23 to 29.

The test circuits number 24 to 29 explore all possible types for the inverting gate implementation of $r, r', aa$ *and* $aa'$ the same way as in Figure 6.15.

The best overall circuits are those in test 22 and 30.




From the circuit in test 23: Insert non-inverting gates into the CKHL(P) section.

Figure 6.18: Dynamic Version: Test 30.

### Final version for the benchmark circuit cm150a

The final circuit of Figure 6.19 is the circuit described by test 30. This circuit shows a number of mixed techniques. For example the N type latches made of two N–C$^2$MOS gates and the P–C$^2$MOS gates at the output are typical of a TSPC circuit. The gates $r$ and $aa$ are used as simple non-inverting dynamic gates, and their interconnection to the output P–C$^2$MOS gates is well understood by the new technique. Finally, the arrangement of the N–C$^2$MOS gates for the subcircuits at $L, q, v$ and $z$ cannot be identified with any other technique because it splits what used to be an inseparable latch structure.

A final comparison is provided in Figure 6.20 to illustrate how timing is affected. The delays given are an arbitrary value based on the number of inputs. The clock period is measured by adding the units from the primary input to the output along the critical path. The result for this benchmark circuit is that the delay for the dynamic-logic circuit is shorter than the static circuit. Both circuits are driven by the same input signals and generate the same output, and, in addition, the dynamic version consumes less power.

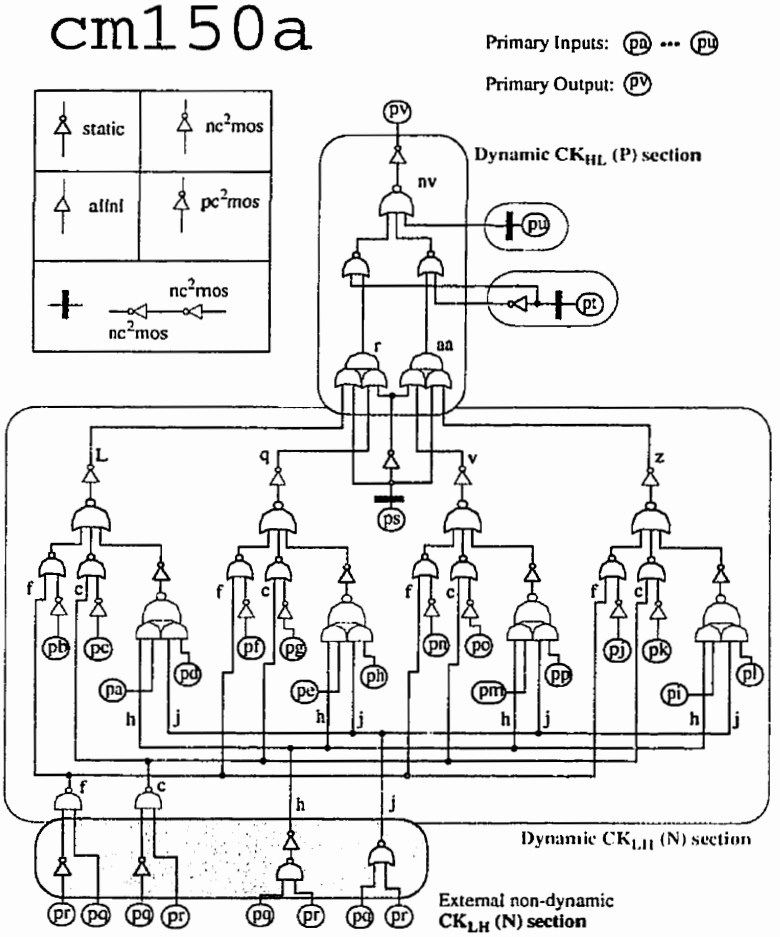
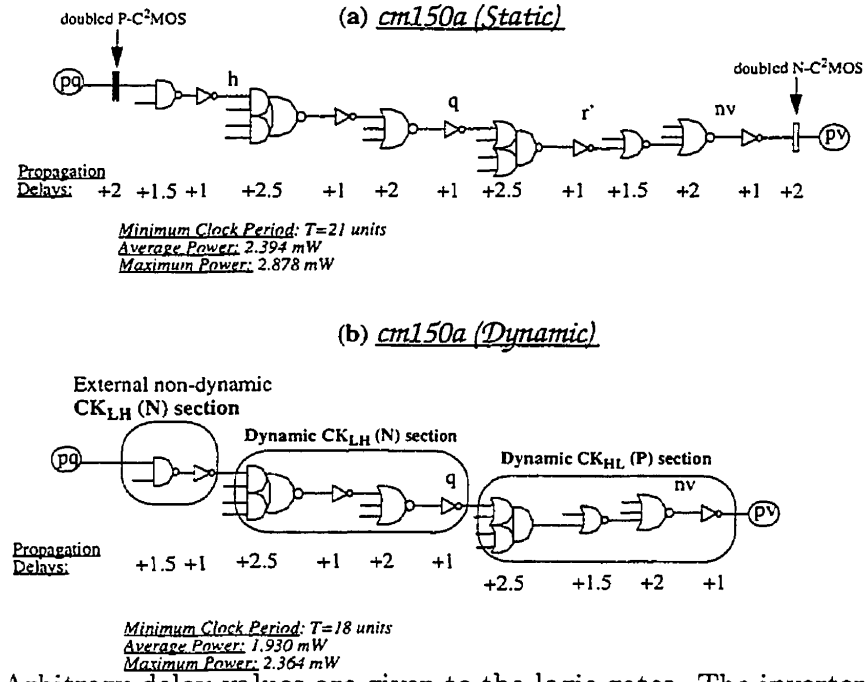



Figure 6.19: cm150a final circuit.




Arbitrary delay values are given to the logic gates. The inverter has a delay of 1 unit, a two input gate has 1.5 units, three input gates have 2 units, four input gates 2.5 units and the latches have 2 units. (a) the static circuit propagates the inputs signals thru the clock-independent gates in 21 units, whereas (b) the dynamic circuit, which has two pipes, provides the final answer after 9*2=18 units.

Figure 6.20: Critical path delay for the cm150a

## 6.2 Benchmark "trapezoid"

The benchmark circuit *trapezoid* was created during the development of an experimental synthesizer for dynamic logic. However the optimization efforts presented in this section are the result of a non-automated optimization. The benchmark circuit is illustrated in Figure 6.21 and it has 12 inputs and 3 outputs and will not have any further logic gate alterations.

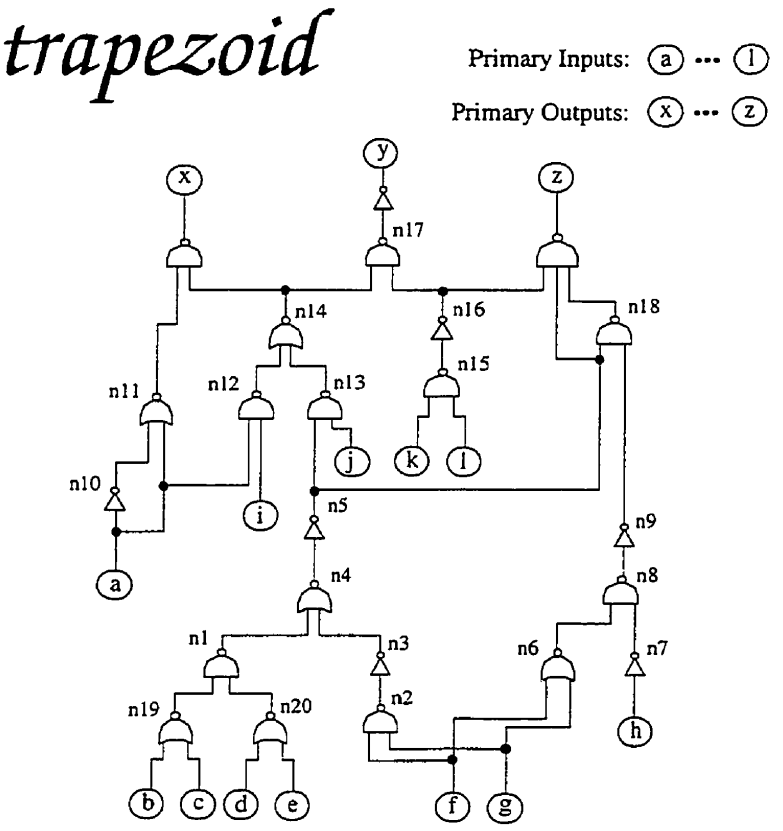



Figure 6.21: Test circuit: the *trapezoid* benchmark without registers

### 6.2.1 Static version: Relocate Latches

The static logic version of the circuit in Figure 6.23 indicates the estimated average and worst power consumption for the same benchmark circuit using TSPC latches at the primary inputs. The following versions of *trapezoid* relocate the P type latches at the output because, it was demonstrated with the *cm150a* benchmark, that power reductions are influenced mostly by the total capacitance when relocating them. Subsequent versions of *trapezoid* in Figure 6.24 to Figure 6.26 explore the placement for N type latches.

The table in Figure 6.22 summarizes the power estimates and the number of latches for each test circuit. The most efficient version is the test circuit number 2 in Figure 6.25 and it has the least power consumed by a static version of the *trapezoid* benchmark.

| test | Power consumption | | Number of Latches | |
|---|---|---|---|---|
| | average | maximum | N–C$^2$MOS | P–C$^2$MOS |
| 0 | 1.454 mW | 1.496 mw | 12 | 12 |
| 1 | 1.212 mW | 1.308 mW | 12 | 3 |
| 2 | 1.076 mW | 1.172 mW | 6 | 3 |
| 3 | 1.164 mW | 1.478 mW | 3 | 3 |

Power consumption for various static logic versions of the benchmark circuit *trapezoid.*

Figure 6.22: Summary of initial tests for the Trapezoid Benchmark

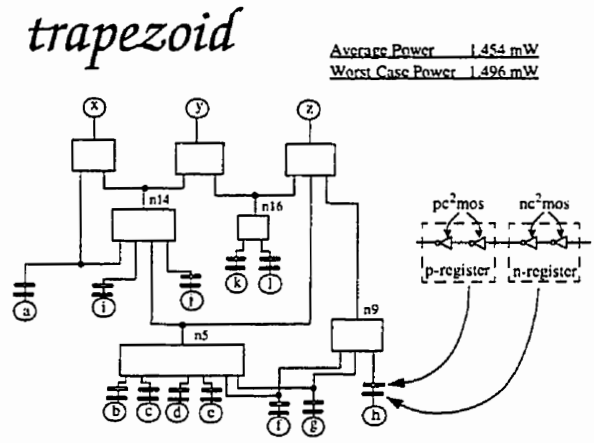



Figure 6.23: Static logic version of *trapezoid* with all latches located at the primary inputs

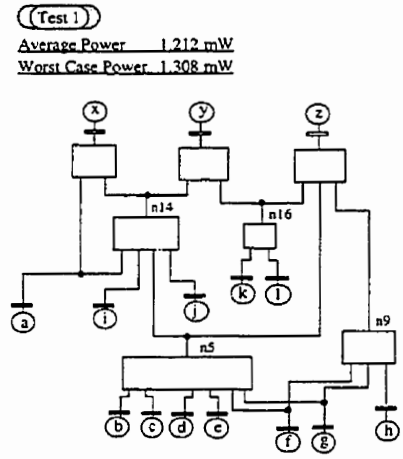



Figure 6.24: Static version: Test 1

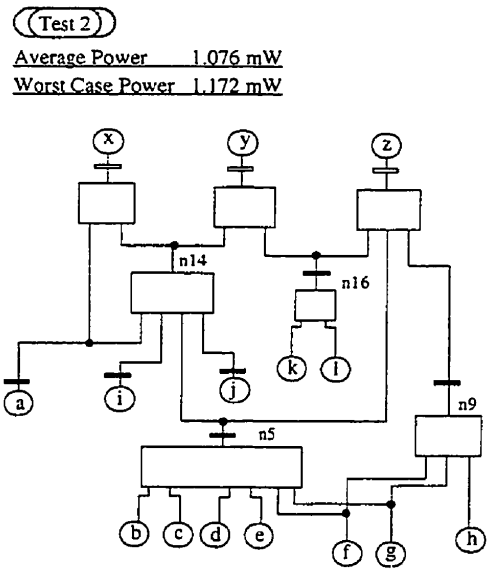



Figure 6.25: Static version: Test 2

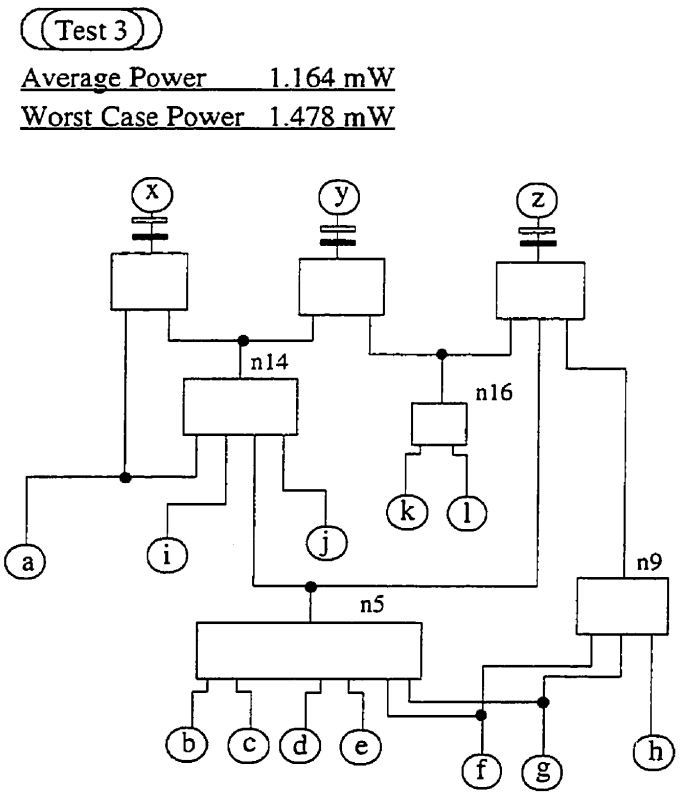
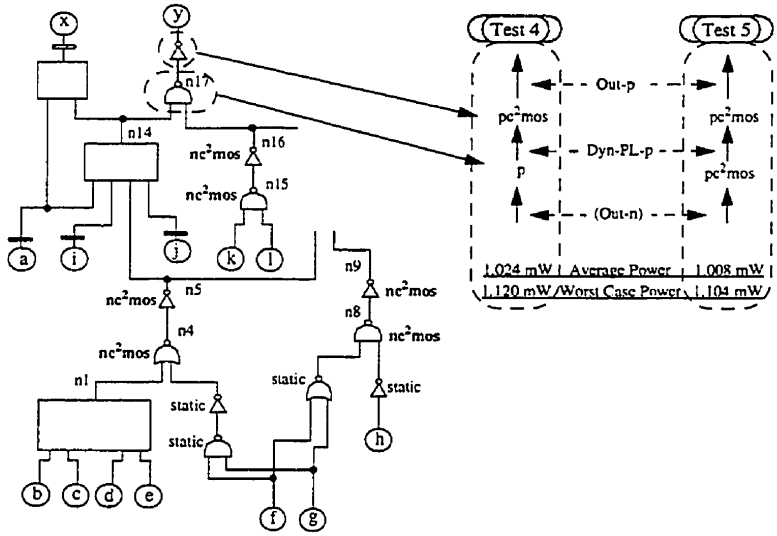



Figure 6.26: Static version: Test 3

### 6.2.2 Dynamic Version: Introduce Latches into the Circuit

Similarly to the previous benchmark, the N and P type latches are introduced into the circuit. The obvious gate type assignments are introduced immediately, and the latches at nodes n5, n9, n16 and $y$ are assimilated. The gate types assigned to gate n17 have two possible gate types, the P and the P–C$^2$MOS gate types are tested in circuits 4 and 5 of Figure 6.27. These gate types are the only choice because their input behavior is *Out-N* and according to the new technique these are the only inverting gate types in the set *Border-P* $\equiv$ {P, All–N–L, P–C$^2$MOS}.




Introduce N and P latches into the circuit.

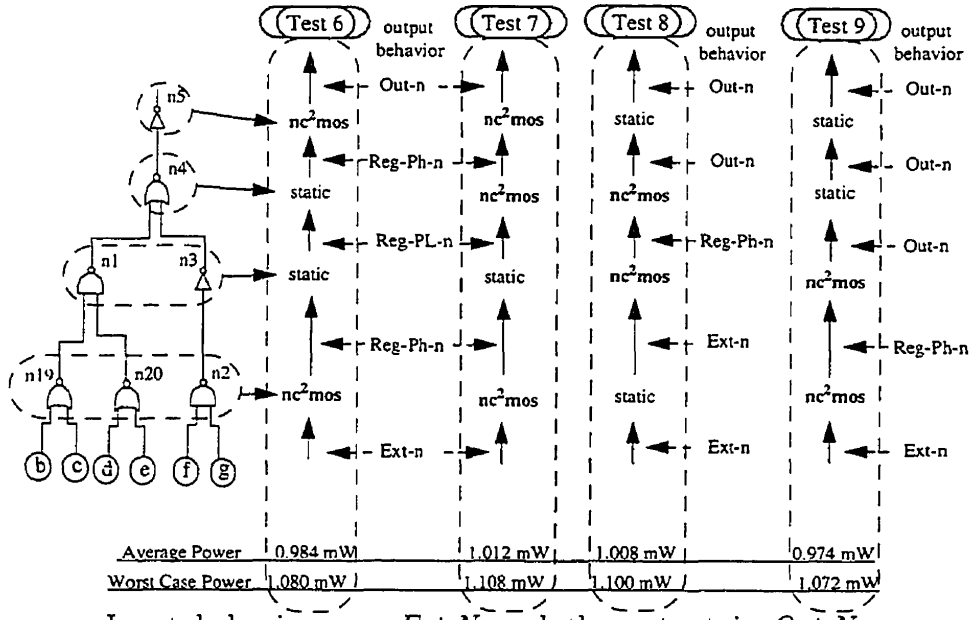
Figure 6.27: Dynamic version: Tests 4 and 5

The circuit found for test 5 is the reference from where further gate type arrangements are explored at nodes n5, n4, n1, n3, n19, n20 and n2. The input behavior to this subcircuit is *Ext-N* and the output is *Out-N*. As in the previous benchmark, the new technique shows that there should be an N–C$^2$MOS gate, according to the set *Ext-Reg-N*, followed by either:

- An N–C$^2$MOS gate which has a *Reg-PH-N* input behavior and a *Out-N* output behavior, or

- One or more structures formed by a static gate followed by either another static gate or an N–C$^2$MOS gate, with input and output behaviors of *Reg-PH-N*. This structure is finally connected to an N–C$^2$MOS which has an output behavior of *Out-N*.

Both items indicating a structure with an even number of gates from input to output. The test circuits numbered 6, 7, 8 and 9 in Figure 6.28 explore all possible arrangements as described by the previous two items.

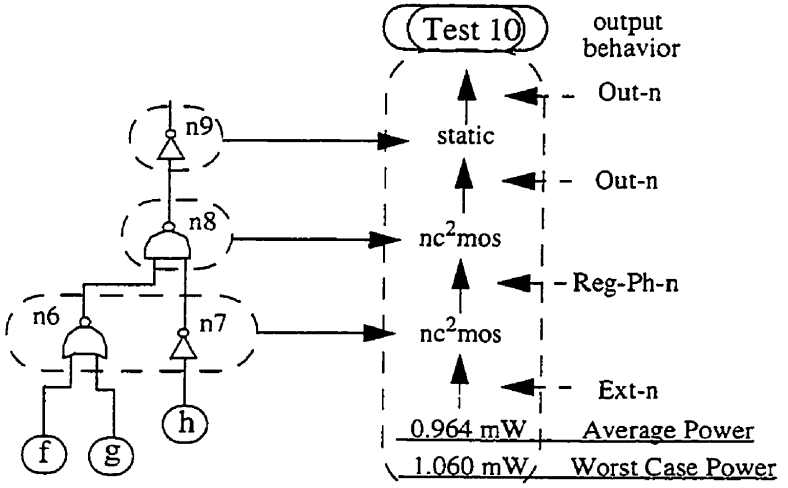



Input behaviors are *Ext-N* and the output is *Out-N*. These circuits explore all possibilities allowed by the new technique under such behavior conditions.

Figure 6.28: Dynamic version: tests 6, 7, 8 and 9

A relocation of boundaries from n8 to n6 and n7 leads to a similar analysis in the test circuit 10 in Figure 6.29, but its constrained to a depth of two

logic gates. The gate types chosen are the only alternative for using N–C$^2$MOS gates and it is derived from the circuit in test 9.




From the circuit in test 9: Relocate the boundaries from n8 to n6 and n7.

Figure 6.29: Dynamic version: Test 10

### 6.2.3 Gate Assignment and Odd Subcircuits

Using dynamic logic according to the rules in the new technique can be a task with various restrictions. There are ways to overcome such difficulties at the expense of loosing the ability to utilize precharged gates such as the N and All–N–L gate types. For example, the mixture of behaviors under a different category in a logic gate results for the most cases in an external behavior, and from there the only gate types allowed are Static and one of the modified-C$^2$MOS latches.

The circuit found by test 10 is illustrated again in Figure 6.30, from this updated version, the test circuits 11 and 12 in Figure 6.31 are an attempt to introduce the P latch into gates X, n11 and n14. The input behavior to n11 and n14 is the crossed behavior (*Out-N*) and is suitable to drive P

dynamic gates. The gate selection is unfortunate for gate n17 because this gate had its type selected on the basis of a previous assumption, where the input behavior is (*Out-N*), and now the input behaviors have changed to *Dyn-PL-P* and (*Out-N*). After evaluation of the non-inverting AND, the output behavior is *Ext-P* and the only possible gate types for n17 and Y which generates the desired *Out-P* behavior is of P–C$^2$MOS for both.

The conclusion for gate n17 is the gate type remains the same as before but its output behavior has changed.

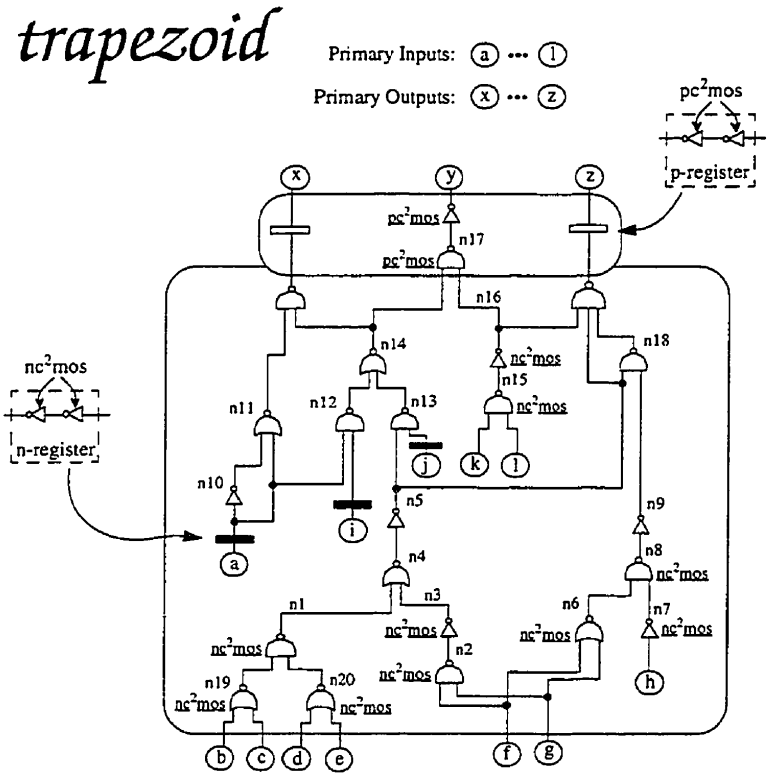
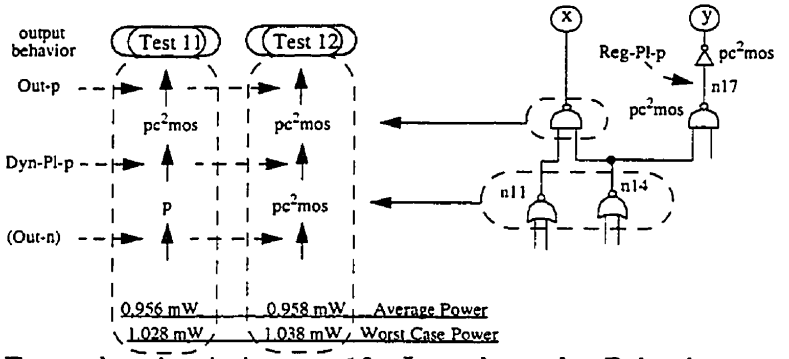



Figure 6.30: Trapezoid benchmark as described by test circuit 10

Another attempt at improving this circuit is exploring the allocation of latches by the definition for boundaries, defined in Section 3.4.3. The test circuit number 13 in Figure 6.32 provides a good example. The gates n14,




From the circuit in test 10: Introduce the P latch into X, n11 and n14. The behavior in n17 changes from *Dyn-PL-P*to *Reg-PL-P*.

Figure 6.31: Dynamic version: Tests 11 and 12

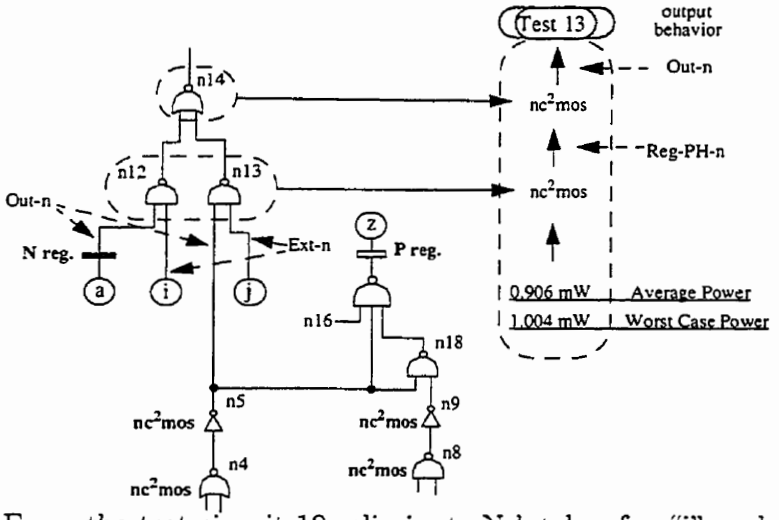
n12 and n13 are given the N–C$^2$MOS type and are now replacing the N latches for two of the primary inputs.

The N latch for "a" is not eliminated and the relationship of its output behavior with the timing in n12 doesn't indicate any further boundary crossings. This is an unusual manipulation of a circuit because the normal procedure would follow the rules indicated by retiming theory [LS91] and it would force the removal of the latch in "a", however it is demonstrated with this example that dynamic logic can take advantage of such anomalies.

Similarly in gate n13, the input behaviors are the primary input "j", which no longer has an N latch, and n5 which has the output behavior *Out-N*. The timing relationship between n5 and n13 indicate no boundary crossings and yet the mixture of behaviors and subsequent output behaviors create a replacement for the N latch from the primary input "j" to the output n14. Furthermore, without changing the subcircuit in n5 there is still a boundary crossing along the path from n5 to the P latch in Z.

The circuit in test 13 is the best performing and its reduced power is attributed mostly to a lower capacitance after eliminating the N latches for "i" and "j".

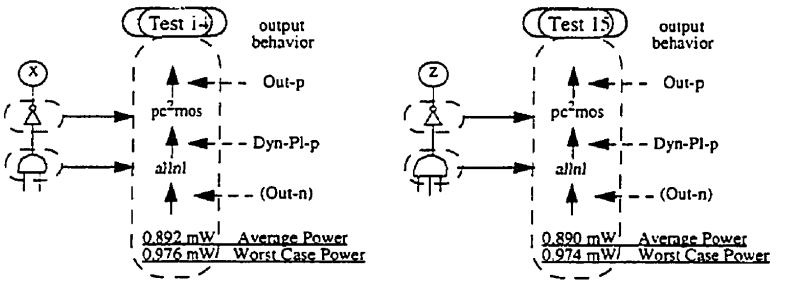
Finally, the test circuits in Figure 6.33 explore the advantages of using




From the test circuit 10: eliminate N latches for "i" and "j".

Figure 6.32: Dynamic version: Test 13

non-inverting gates and are based on the circuit for test 13. The All–N–L gates require simple changes to accommodate their non-inverting logic. The results of these tests are similar to those obtained for test 22 of the cm150a benchmark, where using the All–N–L gates caused better results.




From the circuit in test 13, a) Introduce All–N–L gates in X, and b) Introduce All–N–L gates in Z.

Figure 6.33: Dynamic version: tests 14 and 15

### Final version for the benchmark circuit *trapezoid*

The final circuit of Figure 6.34 is the circuit described by test 15. The circuit shows a selection of gate types that seems to violate retiming rules but is able to properly hold and propagate data. This stems from the better understanding of the edge-triggered behavior for latches by Principle 3.2 and Definition 3.4 of the new technique.

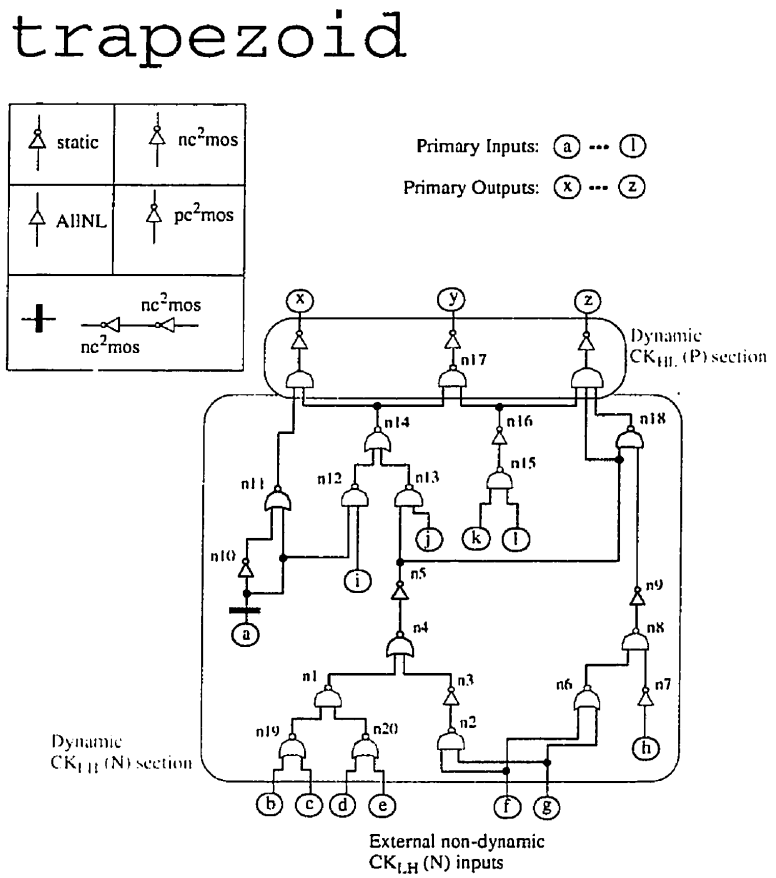
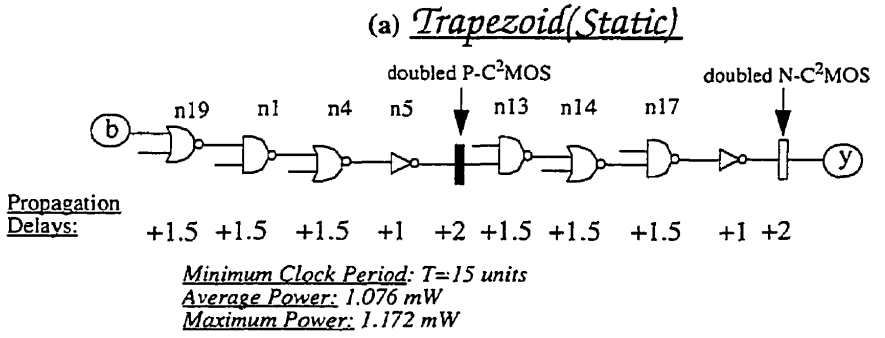


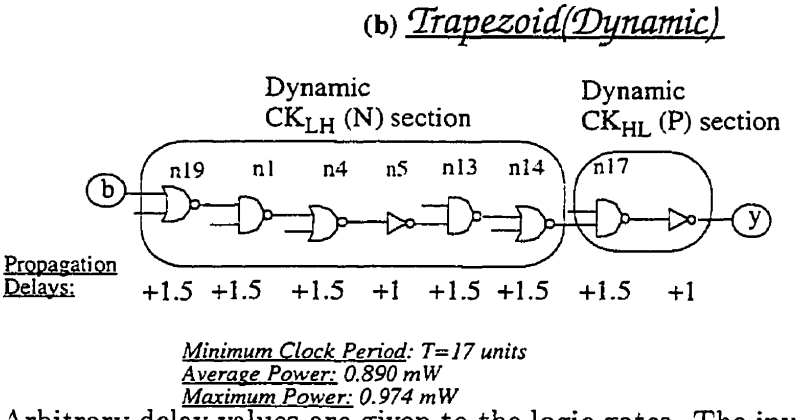

Figure 6.34: *trapezoid* final circuit.

A final comparison is provided in Figure 6.35 to illustrate how timing is affected. The delays given are an arbitrary value based on the number of

inputs. The clock period is measured by adding the units from the primary input to the output along the critical path. The result for this benchmark circuit is that the delay for the static-logic version is shorter than the delay for the dynamic circuit. Both circuits are driven by the same input signals and generate the same output however the dynamic version consumes less power.

The obtained delay numbers in Figure 6.35 indicate an unfair comparison of the power estimates, because slowing down the faster circuit brings down its power consumption. These power estimates can be adjusted based on the ratio of their delays $x = 15/17$, and the power for the static-logic circuit will be of $0.9494mW$ and $1.0341mW$ instead. The arbitrary nature of the delay numbers and the variations of power consumption indicate the limitations of this power estimator tool.



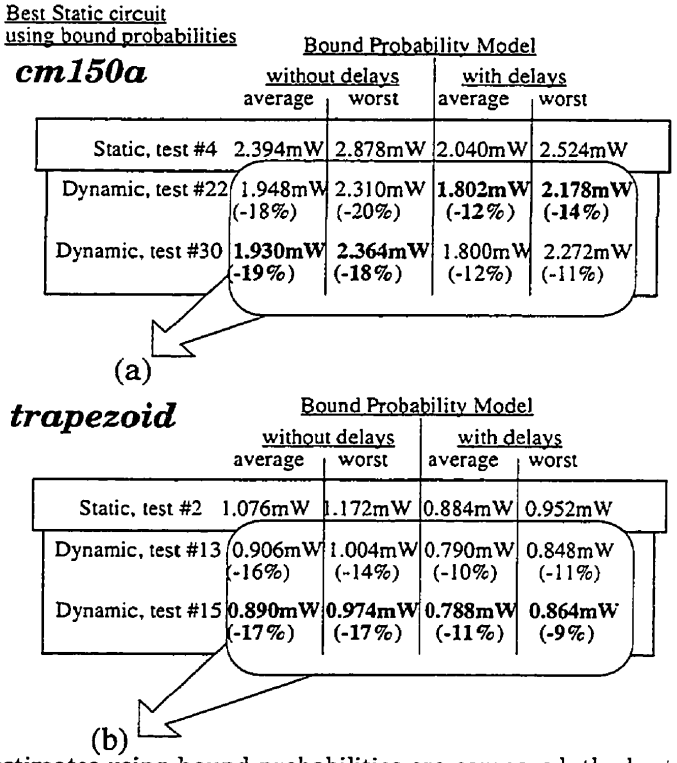

Arbitrary delay values are given to the logic gates. The inverter has a delay of 1 unit, a two input gate has 1.5 units, three input gates have 2 units, four input gates 2.5 units and the latches have 2 units. (a) the static circuit propagates the inputs signals thru the clock-independent gates in 15 units, whereas (b) the dynamic circuit, which has two pipes, provides the final answer after 8.5*2=17 units.

Figure 6.35: Critical path delay for Trapezoid

## 6.3 Summary

The power consumption of the benchmark circuits has been estimated without delays and under a model where bound probabilities are assigned whenever dependencies are generated within the circuit. However there are other perspectives to consider, for example, the effect of delays has been included for all test circuits of both benchmarks and the results from the best performing circuits are compared in Figure 6.36.

These tables indicate that dynamic logic can improve the benchmark circuits by 19% and 17%, for the *cm150a* and *trapezoid* respectively. If the faster circuits are further optimized by slowing their propagation delay along the critical path these dynamic circuits show improvement estimates of 30% and 6%, for the *cm150a* and *trapezoid* respectively. These numbers are obtained after using the ratio of their propagation delays along the critical path, $x = 18/21$ and $x = 15/21$, and reduces the power estimate for the faster circuits. However these propagation delays are arbitrary. Precise timing estimates is a requirement for this type of comparisons where differences in the propagation delay are significant.




Power estimates using bound probabilities are compared, the best Static Logic circuit versus the best Dynamic Logic version of the same. (a) the *cm150a* benchmark shows that the circuit found in test #22 is the best dynamic version whenever delays are considered. (b) the *trapezoid* benchmark shows that the same test #15 is the best dynamic version whether delays are being considered or not.

Figure 6.36: Static vs Dynamic Logic (best Static circuit under bound probabilities)

# Chapter 7

# Conclusion

This thesis has presented the analysis for a group of dynamic logic gates with the purpose of finding their interconnection rules at the gate level. The analysis introduces the principles and definitions that help determine when the behavior of a circuit is logically correct. This analysis is made at a generalized level, appropriate for Dynamic logic circuits, and helps to understand the behavioral relationships between circuits.

The main application of the behavioral relationships of circuits is the development of a new technique for Dynamic Logic circuits. The circuit design rules of this technique were derived for a set of fundamental gates extracted from a target group of Dynamic Logic techniques. Subsequently, the type of circuits obtained with the new technique are also described by these techniques. The difference is that many other structural possibilities are available, every circuit arrangement that is obtained with the new technique is fully justified and guaranteed its logic operation. In addition, the new technique shows a clear behavioral relationship with other types of logic circuits.

Two benchmark circuits were modified from a complete Static Logic implementation to a new version where a mixture of Static and Dynamic Logic gates are being used. The purpose of this benchmark exercise is to demonstrate how to apply the new concepts and to show the flexibility of the new technique. Several transformations of these circuits were evaluated for power consumption using estimates from a probabilistic method. The results ob-

tained with these measurements show that given the right circumstances a Dynamic Logic circuit can consume less power. The limitations of the measurement model must be taken into account for other applications. Nevertheless, the performance model and the benchmark circuits indicate that by choosing the right families Dynamic Logic circuits can be improved.

## 7.1 Contributions

The circuit design theory for Dynamic Logic is an assortment of techniques where each was introduced for a different purpose. New advantages have been found in some components of these techniques and have been presented either as separate gate optimizations or as completely new techniques. The research presented here is the taxonomy of some of these gates and techniques. This type of analysis brings a commonality and shows when a gate type can be connected to another. The following list states the contributions of this research found during the development of this taxonomy for Dynamic gates,

- A new waveform model definition. This model simplifies the description of input and output signals based on a 4-quadrant timing framework.
- A method to forecast the shape of the output waveforms. This method is introduced with the Waveform Response Graphs and it is the essence for the new technique and the performance measurement model.
- The gate interconnection rules are presented in terms of waveform behaviors, together with their classification and relationship to the fundamental gates. This approach tells what inputs can be used and with which gates in order to:
  - Prevent premature discharge of stored data.
  - Avoid the generation and propagation of glitches for precharged gates.
  - Shift data by 180 degrees to form registers as in static logic.

  The gate optimizations found in the literature are compatible with these rules and are classified as part of the technique. Their inclusion

is an outline of previous work but they provide a useful link between the new technique and previous efforts.

- Use of the cutting-algorithm for circuit activity estimates in a performance model. A new employment of this algorithm which was originally developed for testing applications.

- Followed a method applying the gate interconnection rules to modify two circuits into new versions. An assortment of circuits are obtained which, together with a generalized cost function, demonstrate that the new concepts can be part of an optimizing tool.

## 7.2 Observations

The application of these concepts to Computer Aided Design tools is possible because of the simplicity of the gate interconnection rules. The waveform behaviors are used to determine the gate types which can be driven by any structure as long as they have the right behavior. Therefore, a tool within the synthesis process performing the Technology Mapping of a circuit can be developed. The additional computation time is within the linear domain because there is no need to explore structural arrangements but instead their behavioral compatibility. That is the essence of the algorithms for Technology Mapping.

Technology Mapping follows a depth-first order, therefore the optimum gates are found from the primary input to the primary output nodes of a circuit. In a Technology Mapping algorithm the number of logic gate possibilities is independent of the number of gate types but proportional to the number of gates which generate a different behavior. For example, a Static logic circuit is optimized by finding the best circuit providing the output $f$ and $\overline{f}$ for every node. A Dynamic Logic version would have to provide the same two signals but for every waveform behavior category. Therefore the total number of gates to compare is 13 times larger. This is the number of gates found from the rules graph in Chapter 3 which have the Dyn- and Out- behaviors at their input. These are the behaviors driving precharged dynamic gates.

The Cover stage in Technology Mapping is designed to provide a single best solution for the given output behavior for every node in the circuit. Once made, the selection of these gates and their output behavior is not altered because the depth-first order in Technology Mapping. A Static Logic circuit is built by using the optimized gates, either $f$ or $\overline{f}$, for every node in the circuit. The same can be done for Dynamic gates using the behavior categories of the new technique. Therefore the complexity of a Technology Mapping tool for Dynamic gates is directly proportional to that of traditional methods.

The factor of 13 is a rather arbitrary number since it doesn't account for the size or complexity of a circuit. A custom set of gates can be derived to accommodate for the nature and size of the given application. Therefore, the analysis presented by this research is the beginning of a new and versatile type of tool.

The following is an account of what is comprised by the gate interconnection rules of the new technique:

- Analyzed the fundamental gate types N, P, Static, N–C$^2$MOS, P–C$^2$MOS, All–N–L and All–P–L.
- Opposite-Edge input transitions which can generate glitches are prevented. The new technique is based on behavior classification and is able to detect this type of destructive transitions.
- Loss of Precharged Levels due to races between clock and data is prevented. The new technique is able to detect this type of destructive input signals.
- Considered storage characterization for latches. The output waveforms of latches have characteristics which are fully identified and utilized by the new technique.

The waveform model, the Waveform Response Graphs, the interconnection rules, and the power estimation method are all subject to the following limitations:

- The operation of the fundamental gates and their response is based on a single-phase clock signal.

- Effects due to slow transition times are not considered. These effects can deviate the behavior of a circuit from that predicted with the Waveform Response Graphs. However there are optimizations from other techniques that have addressed this problem. Their introduction to the new technique is outlined in Chapter 3.
- The number of transitions, or other transient events, within a single clock cycle is restricted to four. However, the definition of what type and how many transients occur is not congealed to four. Alternative frameworks can be introduced to obtain more comprehensive results. An example is the "extended timing model" and the "transient shift" models of Chapter 5.
- Noise parameters are not considered.
- The following types of circuits are not covered: Differential Cascode-Voltage Switch Logic, Ternary Dynamic logic, Differential Ternary Dynamic logic, or Pass-Transistor logic gates.

## 7.3 Future Work

Future work related to this thesis falls in two categories. First are the improvements to the concepts presented in this thesis, and second are the type of future research that will benefit from this thesis.

### Custom Improvements

Improvements to the new technique are possible because this thesis is the first attempt to effectively mix static with dynamic gates by considering the waveform behavior of circuits. There are other dynamic logic techniques which were not considered because of the extra complexity they would introduce to the presented framework. For instance, some dynamic techniques within the Pass-Transistor logic and the Complementary Voltage-Switch logic show a similar behavior to that in static logic gates and their utilization is similar to that in Domino logic. However, these logic circuits are also realized by other techniques and their inclusion deserves an appropriate amount of research

resources to guarantee a full characterization within the concepts presented in this thesis.

The power measurement method can also be improved. The addition of better algorithms to estimate bound probabilities, which create a tighter range, is expected to be beneficial. Because the accuracy of the estimated activity is dependent on how close these bound probabilities are from the true value.

Further improvements to the power measurement model are expected with a more complex set of symbols to describe glitches. For instance, the definition given for glitches was given in general terms to identify anomalous transients within a region. A customized set of symbols can identify, for instance, the effects by charge leakage in a node and those from multiple transitions within a region. The extra symbols and their introduction to the Waveform Response Graphs will give a better forecast of waveform shapes. Similarly, the introduction of more regions within a clock cycle will help predict more accurately the timing of transients and their influence to the various gate types.

### Future Research

The new technique has been used to optimize a circuit after it has been synthesized for static logic. If better knowledge is obtained regarding the layout/simulation of gates by means of a custom library then a better strategy can be followed. For instance, it has been suggested that during the synthesis of a circuit the Technology Mapping and Logic Minimization stages should be merged into a single stage named Coherent Technology Mapping [SA+93]. Therefore, a custom synthesis tool that selects the appropriate dynamic and static gates from a circuit described at a higher level will, subsequently, explore more effectively for the optimum circuit.

Recent developments have also indicated that high performance dynamic circuits are using the same fundamental gates presented in this thesis [S+98] [FB98]. The circuit in the Alpha chip, for example, optimized its power consumption with a relaxed clocking strategy and a timing analysis method which is dependant on the shape of the input waveforms. Clearly, the method to forecast waveform shapes presented in this thesis can be adapted to pro-

vide, together with slope estimates of signals, fast and accurate timing measurements.

Finally, the addition of features such as scan test to a circuit, which is part of the synthesis process, is a significant topic that deserves an appropriate amount of research resources. The interconnection rules provided in this thesis are among the fundamental concepts to consider for the automated synthesis of dynamic logic including test features.

# Bibliography


[AA75] P. Agrawal and V.D. Agrawal. Probabilistc analysis of random test generation method for irredundant combinational logic networks. *IEEE Transactions on Computers*, pages 691–695, July 1975.

[D+92] Daniel W. Dobberpuhl et al. A 200–mhz 64–b dual–issue CMOS microprocessor. *IEEE Journal of Solid–State Circuits*, 27(11):1555–1567, November 1992.

[Dag91] Michel Dagenais. Efficient algorithmic decomposition of transistor groups into series, bridge and parallel combinations. *IEEE Transactions on Circuits and Systems*, 38(6):569–581, June 1991.

[ERPR95] J.H. Edmondson, P. Rubinfeld, D. Preston, and V. Rajagopalan. Superscalar instruction execution in the 21164 alpha microprocessor. *IEEE Micro*, pages 33–43, April 1995.

[FB98] H. Fair and D. Bailey. Clocking design and analysis for a 600mhz alpha microprocessor. In *IEEE International Solid-State Circuits Conference*, pages 398–399, February 1998.

[FL84] V. Friedman and S. Liu. Dynamic logic cmos circuits. *IEEE Journal Solid-State Circuits*, sc–19:263–266, April 1984.

[GE96] Richard X. Gu and Mohammed I. Elmasry. All-n-logic high–speed true–single–phase dynamic CMOS logic. *IEEE Journal of Solid-State Circuits*, 31(2):221–229, February 1996.

[GM83] Nelson F. Goncalves and Hugo J. De Man. NORA: A racefree dynamic CMOS technique for pipelined logic structures. *IEEE Journal of Solid-State Circuits*, 18(3):261–266, June 1983.

[JKS87] Yuan Jiren, Ingemar Karlsson, and Christer Svensson. A true single-phase-clock dynamic CMOS circuit technique. *IEEE Journal of Solid-State Circuits*, 22(5):899–901, October 1987.

[JS89] Yuan Jiren and Christer Svensson. High-speed CMOS circuit technique. *IEEE Journal of Solid-State Circuits*, 24(1):62–70, February 1989.

[KLL82] R. H. Krambeck, Charles M. Lee, and Hung-Fai Stephen Law. High-speed compact circuits with CMOS. *IEEE Journal of Solid-State Circuits*, 17(3):614–619, June 1982.

[LS91] Charles E. Leiserson and James B. Saxe. Retiming synchronous circuitry. *Algorithmica*, 6:5–35, 1991.

[M+96] Fumio Murabayashi et al. 2.5v cmos circuit techniques for a 200mhz superscalar risc processor. *IEEE Journal of Solid-State Circuits*, pages 972–979, July 1996.

[Man89] Udi Manber. *Introduction to Algorithms.* Addison Wesley, 1989.

[Mic94] Giovanni De Micheli. *Synthesis and Optimization of Digital Circuits.* McGraw-Hill, 1994.

[Rab96] Jan M. Rabaey. *"Digital Integrated Circuits"*. Prentice Hall, 1996.

[S+92] Ellen M. Sentovich et al. "SIS: A system for sequential circuit synthesis". Technical report, Department of Electrical Engineering and Computer Science, University of California, Berkeley, May 1992. Memorandum No UCB/ERL M92/41.

[S+98] J. Silberman et al. A 1.0ghz single-issue 64b powerpc integer processor. In *IEEE International Solid-State Circuits Conference*, pages 230–231, February 1998.

[SA+93] Khalid Sakouti, Pierre Abouzeid, et al. Coherent optimisation strategies for multilevel synthesis. *IEICE Transactions on Information and Systems*, E76-D(9):1093-1101, September 1993.

[SDB84] Jacob Savir, Gary S. Ditlow, and Paul H. Bardell. Random pattern testability. *IEEE Transactions on Computers*, pages 79-90, January 1984.

[Sho88] Masakazu Shoji. CMOS *Digital Circuit Technology*. Prentice Hall, 1988.

[SL94] Christer Svensson and Dake Liu. A power estimation tool and prospects of power savings in CMOS VLSI chips. In *IWLPD Workshop Proceedings*, pages 171-176. 1994.

[SOA73] Yasoji Suzuki, Kaichiro Odagawa, and Toshio Abe. Clocked CMOS calculator circuitry. *IEEE Journal of Solid-State Circuits*, 8(6):462-469, 1973.

[Yea98] Gary K. Yeap. *Practical Low Power Digital VLSI Design*. Kluwer Academic Press, 1998.

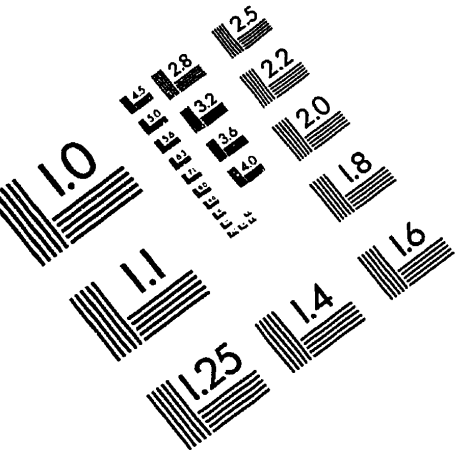
[YS97] Jiren Yuan and Christer Svensson. New single-clock cmos latches and flipflops with improved speed and power saving. *IEEE Journal of Solid-State Circuits*, pages 62-69, January 1997.

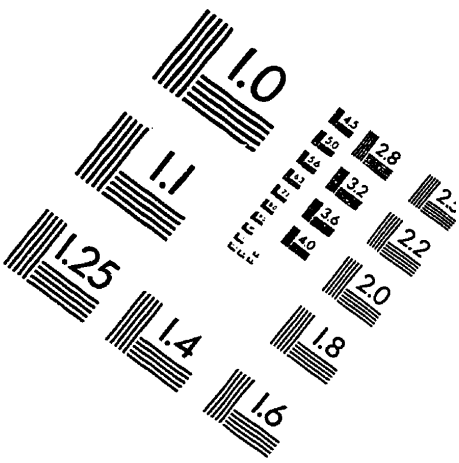
[YSL93] J-R Yuan, C. Svensson, and P. Larsson. New domino logic precharged by clock and data. *Electronics Letters*, 29(25):2188-2189, December 1993.

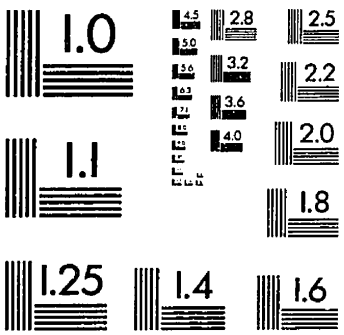
[ZAEB93] Jiabi Zhu and Mostafa Abd-El-Barr. On the optimization of mos circuits. *IEEE Transactions on Circuits and Systems-I:Fundamental Theory and Applications*, 40(6):412-422, June 1993.

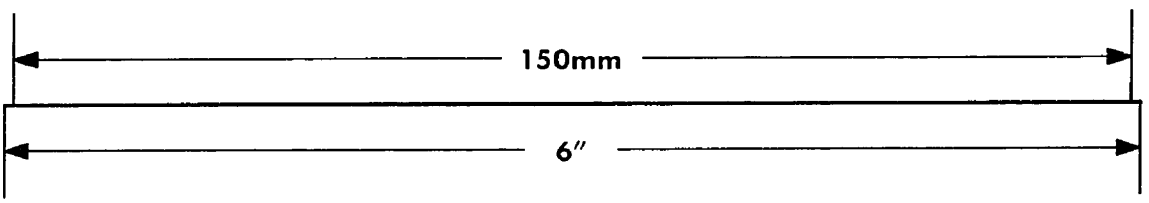
1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
7.1
8.0
9.0
10.0
11.0
12.5
14.0
16.0
18.0
20.0
22.0
25.0
28.0
32.0
36.0
40.0
45.0
50.0
56.0
63.0
71.0
80.0
90.0
100.0
112.0
125.0
140.0
160.0
180.0
200.0
220.0
250.0
280.0
320.0
360.0
400.0
450.0
500.0
560.0
630.0
710.0
800.0
900.0
1000.0
1120.0
1250.0
1400.0
1600.0
1800.0
2000.0
2200.0
2500.0
2800.0
3200.0
3600.0
4000.0
4500.0
5000.0
5600.0
6300.0
7100.0
8000.0
9000.0
10000.0
11200.0
12500.0
14000.0
16000.0
18000.0
20000.0
22000.0
25000.0
28000.0
32000.0
36000.0
40000.0
45000.0
50000.0
56000.0
63000.0
71000.0
80000.0
90000.0
100000.0
112000.0
125000.0
140000.0
160000.0
180000.0
200000.0
220000.0
250000.0
280000.0
320000.0
360000.0
400000.0
450000.0
500000.0
560000.0
630000.0
710000.0
800000.0
900000.0
1000000.0
1120000.0
1250000.0
1400000.0
1600000.0
1800000.0
2000000.0
2200000.0
2500000.0
2800000.0
3200000.0
3600000.0
4000000.0
4500000.0
5000000.0
5600000.0
6300000.0
7100000.0
8000000.0
9000000.0
10000000.0
11200000.0
12500000.0
14000000.0
16000000.0
18000000.0
20000000.0
22000000.0
25000000.0
28000000.0
32000000.0
36000000.0
40000000.0
45000000.0
50000000.0
56000000.0
63000000.0
71000000.0
80000000.0
90000000.0
100000000.0
112000000.0
125000000.0
140000000.0
160000000.0
180000000.0
200000000.0
220000000.0
250000000.0
280000000.0
320000000.0
360000000.0
400000000.0
450000000.0
500000000.0
560000000.0
630000000.0
710000000.0
800000000.0
900000000.0
1000000000.0
1120000000.0
1250000000.0
1400000000.0
1600000000.0
1800000000.0
2000000000.0
2200000000.0
2500000000.0
2800000000.0
3200000000.0
3600000000.0
4000000000.0
4500000000.0
5000000000.0
5600000000.0
6300000000.0
7100000000.0
8000000000.0
9000000000.0
10000000000.0
11200000000.0
12500000000.0
14000000000.0
16000000000.0
18000000000.0
20000000000.0
22000000000.0
25000000000.0
28000000000.0
32000000000.0
36000000000.0
40000000000.0
45000000000.0
50000000000.0
56000000000.0
63000000000.0
71000000000.0
80000000000.0
90000000000.0
100000000000.0
112000000000.0
125000000000.0
140000000000.0
160000000000.0
180000000000.0
200000000000.0
220000000000.0
250000000000.0
280000000000.0
320000000000.0
360000000000.0
400000000000.0
450000000000.0
500000000000.0
560000000000.0
630000000000.0
710000000000.0
800000000000.0
900000000000.0
1000000000000.0
1120000000000.0
1250000000000.0
1400000000000.0
1600000000000.0
1800000000000.0
2000000000000.0
2200000000000.0
2500000000000.0
2800000000000.0
3200000000000.0
3600000000000.0
4000000000000.0
4500000000000.0
5000000000000.0
5600000000000.0
6300000000000.0
7100000000000.0
8000000000000.0
9000000000000.0
10000000000000.0
11200000000000.0
12500000000000.0
14000000000000.0
16000000000000.0
18000000000000.0
20000000000000.0
22000000000000.0
25000000000000.0
28000000000000.0
32000000000000.0
36000000000000.0
40000000000000.0
45000000000000.0
50000000000000.0
56000000000000.0
63000000000000.0
71000000000000.0
80000000000000.0
90000000000000.0
100000000000000.0
112000000000000.0
125000000000000.0
140000000000000.0
160000000000000.0
180000000000000.0
200000000000000.0
220000000000000.0
250000000000000.0
280000000000000.0
320000000000000.0
360000000000000.0
400000000000000.0
450000000000000.0
500000000000000.0
560000000000000.0
630000000000000.0
710000000000000.0
800000000000000.0
900000000000000.0
1000000000000000.0
1120000000000000.0
1250000000000000.0
1400000000000000.0
1600000000000000.0
1800000000000000.0
2000000000000000.0
2200000000000000.0
2500000000000000.0
2800000000000000.0
3200000000000000.0
3600000000000000.0
4000000000000000.0
4500000000000000.0
5000000000000000.0
5600000000000000.0
6300000000000000.0
7100000000000000.0
8000000000000000.0
9000000000000000.0
10000000000000000.0
11200000000000000.0
12500000000000000.0
14000000000000000.0
16000000000000000.0
18000000000000000.0
20000000000000000.0
22000000000000000.0
25000000000000000.0
28000000000000000.0
32000000000000000.0
36000000000000000.0
40000000000000000.0
45000000000000000.0
50000000000000000.0
56000000000000000.0
63000000000000000.0
71000000000000000.0
80000000000000000.0
90000000000000000.0
100000000000000000.0
112000000000000000.0
125000000000000000.0
140000000000000000.0
160000000000000000.0
180000000000000000.0
200000000000000000.0
220000000000000000.0
250000000000000000.0
280000000000000000.0
320000000000000000.0
360000000000000000.0
400000000000000000.0
450000000000000000.0
500000000000000000.0
560000000000000000.0
630000000000000000.0
710000000000000000.0
800000000000000000.0
900000000000000000.0
1000000000000000000.0
1120000000000000000.0
1250000000000000000.0
1400000000000000000.0
1600000000000000000.0
1800000000000000000.0
2000000000000000000.0
2200000000000000000.0
2500000000000000000.0
2800000000000000000.0
3200000000000000000.0
3600000000000000000.0
4000000000000000000.0
4500000000000000000.0
5000000000000000000.0
5600000000000000000.0
6300000000000000000.0
7100000000000000000.0
8000000000000000000.0
9000000000000000000.0
10000000000000000000.0
11200000000000000000.0
12500000000000000000.0
14000000000000000000.0
16000000000000000000.0
18000000000000000000.0
20000000000000000000.0
22000000000000000000.0
25000000000000000000.0
28000000000000000000.0
32000000000000000000.0
36000000000000000000.0
40000000000000000000.0
45000000000000000000.0
50000000000000000000.0
56000000000000000000.0
63000000000000000000.0
71000000000000000000.0
80000000000000000000.0
90000000000000000000.0
100000000000000000000.0
112000000000000000000.0
125000000000000000000.0
140000000000000000000.0
160000000000000000000.0
180000000000000000000.0
200000000000000000000.0
220000000000000000000.0
250000000000000000000.0
280000000000000000000.0
320000000000000000000.0
360000000000000000000.0
400000000000000000000.0
450000000000000000000.0
500000000000000000000.0
560000000000000000000.0
630000000000000000000.0
710000000000000000000.0
800000000000000000000.0
900000000000000000000.0
1000000000000000000000.0
1120000000000000000000.0
1250000000000000000000.0
1400000000000000000000.0
1600000000000000000000.0
1800000000000000000000.0
2000000000000000000000.0
2200000000000000000000.0
2500000000000000000000.0
2800000000000000000000.0
3200000000000000000000.0
3600000000000000000000.0
4000000000000000000000.0
4500000000000000000000.0
5000000000000000000000.0
5600000000000000000000.0
6300000000000000000000.0
7100000000000000000000.0
8000000000000000000000.0
9000000000000000000000.0
10000000000000000000000.0
11200000000000000000000.0
12500000000000000000000.0
14000000000000000000000.0
16000000000000000000000.0
18000000000000000000000.0
20000000000000000000000.0
22000000000000000000000.0
25000000000000000000000.0
28000000000000000000000.0
32000000000000000000000.0
36000000000000000000000.0
40000000000000000000000.0
45000000000000000000000.0
50000000000000000000000.0
56000000000000000000000.0
63000000000000000000000.0
71000000000000000000000.0
80000000000000000000000.0
90000000000000000000000.0
100000000000000000000000.0
112000000000000000000000.0
125000000000000000000000.0
140000000000000000000000.0
160000000000000000000000.0
180000000000000000000000.0
200000000000000000000000.0
220000000000000000000000.0
250000000000000000000000.0
280000000000000000000000.0
320000000000000000000000.0
360000000000000000000000.0
400000000000000000000000.0
450000000000000000000000.0
500000000000000000000000.0
560000000000000000000000.0
630000000000000000000000.0
710000000000000000000000.0
800000000000000000000000.0
900000000000000000000000.0
1000000000000000000000000.0
1120000000000000000000000.0
1250000000000000000000000.0
1400000000000000000000000.0
1600000000000000000000000.0
1800000000000000000000000.0
2000000000000000000000000.0
2200000000000000000000000.0
2500000000000000000000000.0
2800000000000000000000000.0
3200000000000000000000000.0
3600000000000000000000000.0
4000000000000000000000000.0
4500000000000000000000000.0
5000000000000000000000000.0
5600000000000000000000000.0
6300000000000000000000000.0
7100000000000000000000000.0
8000000000000000000000000.0
9000000000000000000000000.0
10000000000000000000000000.0
11200000000000000000000000.0
12500000000000000000000000.0
14000000000000000000000000.0
16000000000000000000000000.0
18000000000000000000000000.0
20000000000000000000000000.0
22000000000000000000000000.0
25000000000000000000000000.0
28000000000000000000000000.0
32000000000000000000000000.0
36000000000000000000000000.0
40000000000000000000000000.0
45000000000000000000000000.0
50000000000000000000000000.0
56000000000000000000000000.0
63000000000000000000000000.0
71000000000000000000000000.0
80000000000000000000000000.0
90000000000000000000000000.0
100000000000000000000000000.0
112000000000000000000000000.0
125000000000000000000000000.0
140000000000000000000000000.0
160000000000000000000000000.0
180000000000000000000000000.0
200000000000000000000000000.0
220000000000000000000000000.0
250000000000000000000000000.0
280000000000000000000000000.0
320000000000000000000000000.0
360000000000000000000000000.0
400000000000000000000000000.0
450000000000000000000000000.0
500000000000000000000000000.0
560000000000000000000000000.0
630000000000000000000000000.0
710000000000000000000000000.0
800000000000000000000000000.0
900000000000000000000000000.0
1000000000000000000000000000.0
1120000000000000000000000000.0
1250000000000000000000000000.0
1400000000000000000000000000.0
1600000000000000000000000000.0
1800000000000000000000000000.0
2000000000000000000000000000.0
2200000000000000000000000000.0
2500000000000000000000000000.0
2800000000000000000000000000.0
3200000000000000000000000000.0
3600000000000000000000000000.0
4000000000000000000000000000.0
4500000000000000000000000000.0
5000000000000000000000000000.0
5600000000000000000000000000.0
6300000000000000000000000000.0
7100000000000000000000000000.0
8000000000000000000000000000.0
9000000000000000000000000000.0
10000000000000000000000000000.0
11200000000000000000000000000.0
12500000000000000000000000000.0
14000000000000000000000000000.0
16000000000000000000000000000.0
18000000000000000000000000000.0
20000000000000000000000000000.0
22000000000000000000000000000.0
25000000000000000000000000000.0
28000000000000000000000000000.0
32000000000000000000000000000.0
36000000000000000000000000000.0
40000000000000000000000000000.0
45000000000000000000000000000.0
50000000000000000000000000000.0
56000000000000000000000000000.0
63000000000000000000000000000.0
71000000000000000000000000000.0
80000000000000000000000000000.0
90000000000000000000000000000.0
100000000000000000000000000000.0
112000000000000000000000000000.0
125000000000000000000000000000.0
140000000000000000000000000000.0
160000000000000000000000000000.0
180000000000000000000000000000.0
200000000000000000000000000000.0
220000000000000000000000000000.0
250000000000000000000000000000.0
280000000000000000000000000000.0
320000000000000000000000000000.0
360000000000000000000000000000.0
400000000000000000000000000000.0
450000000000000000000000000000.0
500000000000000000000000000000.0
560000000000000000000000000000.0
630000000000000000000000000000.0
710000000000000000000000000000.0
800000000000000000000000000000.0
900000000000000000000000000000.0
1000000000000000000000000000000.0
1120000000000000000000000000000.0
1250000000000000000000000000000.0
1400000000000000000000000000000.0
1600000000000000000000000000000.0
1800000000000000000000000000000.0
2000000000000000000000000000000.0
2200000000000000000000000000000.0
2500000000000000000000000000000.0
2800000000000000000000000000000.0
3200000000000000000000000000000.0
3600000000000000000000000000000.0
4000000000000000000000000000000.0
4500000000000000000000000000000.0
5000000000000000000000000000000.0
5600000000000000000000000000000.0
6300000000000000000000000000000.0
7100000000000000000000000000000.0
8000000000000000000000000000000.0
9000000000000000000000000000000.0
10000000000000000000000000000000.0
11200000000000000000000000000000.0
12500000000000000000000000000000.0
14000000000000000000000000000000.0
16000000000000000000000000000000.0
18000000000000000000000000000000.0
20000000000000000000000000000000.0
22000000000000000000000000000000.0
25000000000000000000000000000000.0
28000000000000000000000000000000.0
32000000000000000000000000000000.0
36000000000000000000000000000000.0
40000000000000000000000000000000.0
45000000000000000000000000000000.0
50000000000000000000000000000000.0
56000000000000000000000000000000.0
63000000000000000000000000000000.0
71000000000000000000000000000000.0
80000000000000000000000000000000.0
90000000000000000000000000000000.0
100000000000000000000000000000000.0
112000000000000000000000000000000.0
125000000000000000000000000000000.0
140000000000000000000000000000000.0
160000000000000000000000000000000.0
180000000000000000000000000000000.0
200000000000000000000000000000000.0
220000000000000000000000000000000.0
250000000000000000000000000000000.0
280000000000000000000000000000000.0
320000000000000000000000000000000.0
360000000000000000000000000000000.0
400000000000000000000000000000000.0
450000000000000000000000000000000.0
500000000000000000000000000000000.0
560000000000000000000000000000000.0
630000000000000000000000000000000.0
710000000000000000000000000000000.0
800000000000000000000000000000000.0
900000000000000000000000000000000.0
1000000000000000000000000000000000.0
1120000000000000000000000000000000.0
1250000000000000000000000000000000.0
1400000000000000000000000000000000.0
1600000000000000000000000000000000.0
1800000000000000000000000000000000.0
2000000000000000000000000000000000.0
2200000000000000000000000000000000.0
2500000000000000000000000000000000.0
2800000000000000000000000000000000.0
3200000000000000000000000000000000.0
3600000000000000000000000000000000.0
4000000000000000000000000000000000.0
4500000000000000000000000000000000.0
5000000000000000000000000000000000.0
5600000000000000000000000000000000.0
6300000000000000000000000000000000.0
7100000000000000000000000000000000.0
8000000000000000000000000000000000.0
9000000000000000000000000000000000.0
10000000000000000000000000000000000.0
11200000000000000000000000000000000.0
12500000000000000000000000000000000.0
14000000000000000000000000000000000.0
16000000000000000000000000000000000.0
18000000000000000000000000000000000.0
20000000000000000000000000000000000.0
22000000000000000000000000000000000.0
25000000000000000000000000000000000.0
28000000000000000000000000000000000.0
32000000000000000000000000000000000.0
36000000000000000000000000000000000.0
40000000000000000000000000000000000.0
45000000000000000000000000000000000.0
50000000000000000000000000000000000.0
56000000000000000000000000000000000.0
63000000000000000000000000000000000.0
71000000000000000000000000000000000.0
80000000000000000000000000000000000.0
90000000000000000000000000000000000.0
100000000000000000000000000000000000.0
112000000000000000000000000000000000.0
125000000000000000000000000000000000.0
140000000000000000000000000000000000.0
160000000000000000000000000000000000.0
180000000000000000000000000000000000.0
200000000000000000000000000000000000.0
220000000000000000000000000000000000.0
250000000000000000000000000000000000.0
280000000000000000000000000000000000.0
320000000000000000000000000000000000.0
360000000000000000000000000000000000.0
400000000000000000000000000000000000.0
450000000000000000000000000000000000.0
500000000000000000000000000000000000.0
560000000000000000000000000000000000.0
630000000000000000000000000000000000.0
710000000000000000000000000000000000.0
800000000000000000000000000000000000.0
900000000000000000000000000000000000.0
1000000000000000000000000000000000000.0
1120000000000000000000000000000000000.0
1250000000000000000000000000000000000.0
1400000000000000000000000000000000000.0
1600000000000000000000000000000000000.0
1800000000000000000000000000000000000.0
2000000000000000000000000000000000000.0
2200000000000000000000000000000000000.0
2500000000000000000000000000000000000.0
2800000000000000000000000000000000000.0
3200000000000000000000000000000000000.0
3600000000000000000000000000000000000.0
4000000000000000000000000000000000000.0
4500000000000000000000000000000000000.0
5000000000000000000000000000000000000.0
5600000000000000000000000000000000000.0
6300000000000000000000000000000000000.0
7100000000000000000000000000000000000.0
8000000000000000000000000000000000000.0
9000000000000000000000000000000000000.0
10000000000000000000000000000000000000.0
11200000000000000000000000000000000000.0
12500000000000000000000000000000000000.0
14000000000000000000000000000000000000.0
16000000000000000000000000000000000000.0
18000000000000000000000000000000000000.0
20000000000000000000000000000000000000.0
22000000000000000000000000000000000000.0
25000000000000000000000000000000000000.0
28000000000000000000000000000000000000.0
32000000000000000000000000000000000000.0
36000000000000000000000000000000000000.0
40000000000000000000000000000000000000.0
45000000000000000000000000000000000000.0
50000000000000000000000000000000000000.0
56000000000000000000000000000000000000.0
63000000000000000000000000000000000000.0
71000000000000000000000000000000000000.0
80000000000000000000000000000000000000.0
90000000000000000000000000000000000000.0
100000000000000000000000000000000000000.0
112000000000000000000000000000000000000.0
125000000000000000000000000000000000000.0
140000000000000000000000000000000000000.0
160000000000000000000000000000000000000.0
180000000000000000000000000000000000000.0
200000000000000000000000000000000000000.0
220000000000000000000000000000000000000.0
250000000000000000000000000000000000000.0
280000000000000000000000000000000000000.0
320000000000000000000000000000000000000.0
360000000000000000000000000000000000000.0
400000000000000000000000000000000000000.0
450000000000000000000000000000000000000.0
500000000000000000000000000000000000000.0
560000000000000000000000000000000000000.0
630000000000000000000000000000000000000.0
710000000000000000000000000000000000000.0
800000000000000000000000000000000000000.0
900000000000000000000000000000000000000.0
1000000000000000000000000000000000000000.0
1120000000000000000000000000000000000000.0
1250000000000000000000000000000000000000.0
1400000000000000000000000000000000000000.0
1600000000000000000000000000000000000000.0
1800000000000000000000000000000000000000.0
2000000000000000000000000000000000000000.0
2200000000000000000000000000000000000000.0
2500000000000000000000000000000000000000.0
2800000000000000000000000000000000000000.0
3200000000000000000000000000000000000000.0
3600000000000000000000000000000000000000.0
4000000000000000000000000000000000000000.0
4500000000000000000000000000000000000000.0
5000000000000000000000000000000000000000.0
5600000000000000000000000000000000000000.0
6300000000000000000000000000000000000000.0
7100000000000000000000000000000000000000.0
8000000000000000000000000000000000000000.0
9000000000000000000000000000000000000000.0
10000000000000000000000000000000000000000.0
11200000000000000000000000000000000000000.0
12500000000000000000000000000000000000000.0
14000000000000000000000000000000000000000.0
16000000000000000000000000000000000000000.0
18000000000000000000000000000000000000000.0
20000000000000000000000000000000000000000.0
22000000000000000000000000000000000000000.0
25000000000000000000000000000000000000000.0
28000000000000000000000000000000000000000.0
32000000000000000000000000000000000000000.0
36000000000000000000000000000000000000000.0
40000000000000000000000000000000000000000.0
45000000000000000000000000000000000000000.0
50000000000000000000000000000000000000000.0
56000000000000000000000000000000000000000.0
63000000000000000000000000000000000000000.0
71000000000000000000000000000000000000000.0
80000000000000000000000000000000000000000.0
90000000000000000000000000000000000000000.0
100000000000000000000000000000000000000000.0
112000000000000000000000000000000000000000.0
125000000000000000000000000000000000000000.0
140000000000000000000000000000000000000000.0
160000000000000000000000000000000000000000.0
180000000000000000000000000000000000000000.0
200000000000000000000000000000000000000000.0
220000000000000000000000000000000000000000.0
250000000000000000000000000000000000000000.0
280000000000000000000000000000000000000000.0
320000000000000000000000000000000000000000.0
360000000000000000000000000000000000000000.0
400000000000000000000000000000000000000000.0
450000000000000000000000000000000000000000.0
500000000000000000000000000000000000000000.0
560000000000000000000000000000000000000000.0
630000000000000000000000000000000000000000.0
710000000000000000000000000000000000000000.0
800000000000000000000000000000000000000000.0
900000000000000000000000000000000000000000.0
1000000000000000000000000000000000000000000.0
1120000000000000000000000000000000000000000.0
1250000000000000000000000000000000000000000.0
1400000000000000000000000000000000000000000.0
1600000000000000000000000000000000000000000.0
1800000000000000000000000000000000000000000.0
2000000000000000000000000000000000000000000.0
2200000000000000000000000000000000000000000.0
2500000000000000000000000000000000000000000.0
2800000000000000000000000000000000000000000.0
3200000000000000000000000000000000000000000.0
3600000000000000000000000000000000000000000.0
4000000000000000000000000000000000000000000.0
4500000000000000000000000000000000000000000.0
5000000000000000000000000000000000000000000.0
5600000000000000000000000000000000000000000.0
6300000000000000000000000000000000000000000.0
7100000000000000000000000000000000000000000.0
8000000000000000000000000000000000000000000.0
9000000000000000000000000000000000000000000.0
10000000000000000000000000000000000000000000.0
11200000000000000000000000000000000000000000.0
12500000000000000000000000000000000000000000.0
14000000000000000000000000000000000000000000.0
16000000000000000000000000000000000000000000.0
18000000000000000000000000000000000000000000.0
20000000000000000000000000000000000000000000.0
22000000000000000000000000000000000000000000.0
25000000000000000000000000000000000000000000.0
28000000000000000000000000000000000000000000.0
32000000000000000000000000000000000000000000.0
36000000000000000000000000000000000000000000.0
40000000000000000000000000000000000000000000.0
45000000000000000000000000000000000000000000.0
50000000000000000000000000000000000000000000.0
56000000000000000000000000000000000000000000.0
63000000000000000000000000000000000000000000.0
71000000000000000000000000000000000000000000.0
80000000000000000000000000000000000000000000.0
90000000000000000000000000000000000000000000.0
100000000000000000000000000000000000000000000.0
112000000000000000000000000000000000000000000.0
125000000000000000000000000000000000000000000.0
140000000000000000000000000000000000000000000.0
160000000000000000000000000000000000000000000.0
180000000000000000000000000000000000000000000.0
200000000000000000000000000000000000000000000.0
220000000000000000000000000000000000000000000.0
250000000000000000000000000000000000000000000.0
280000000000000000000000000000000000000000000.0
320000000000000000000000000000000000000000000.0
360000000000000000000000000000000000000000000.0
400000000000000000000000000000000000000000000.0
450000000000000000000000000000000000000000000.0
500000000000000000000000000000000000000000000.0
560000000000000000000000000000000000000000000.0
630000000000000000000000000000000000000000000.0
710000000000000000000000000000000000000000000.0
800000000000000000000000000000000000000000000.0
900000000000000000000000000000000000000000000.0
1000000000000000000000000000000000000000000000.0
1120000000000000000000000000000000000000000000.0
1250000000000000000000000000000000000000000000.0
1400000000000000000000000000000000000000000000.0
1600000000000000000000000000000000000000000000.0
1800000000000000000000000000000000000000000000.0
2000000000000000000000000000000000000000000000.0
2200000000000000000000000000000000000000000000.0
2500000000000000000000000000000000000000000000.0
2800000000000000000000000000000000000000000000.0
3200000000000000000000000000000000000000000000.0
3600000000000000000000000000000000000000000000.0
4000000000000000000000000000000000000000000000.0
4500000000000000000000000000000000000000000000.0
5000000000000000000000000000000000000000000000.0
5600000000000000000000000000000000000000000000.0
6300000000000000000000000000000000000000000000.0
7100000000000000000000000000000000000000000000.0
8000000000000000000000000000000000000000000000.0
9000000000000000000000000000000000000000000000.0
10000000000000000000000000000000000000000000000.0
11200000000000000000000000000000000000000000000.0
12500000000000000000000000000000000000000000000.0
14000000000000000000000000000000000000000000000.0
16000000000000000000000000000000000000000000000.0
18000000000000000000000000000000000000000000000.0
20000000000000000000000000000000000000000000000.0
22000000000000000000000000000000000000000000000.0
25000000000000000000000000000000000000000000000.0
28000000000000000000000000000000000000000000000.0
32000000000000000000000000000000000000000000000.0
36000000000000000000000000000000000000000000000.0
40000000000000000000000000000000000000000000000.0
45000000000000000000000000000000000000000000000.0
50000000000000000000000000000000000000000000000.0
56000000000000000000000000000000000000000000000.0
63000000000000000000000000000000000000000000000.0
71000000000000000000000000000000000000000000000.0
80000000000000000000000000000000000000000000000.0
90000000000000000000000000000000000000000000000.0
100000000000000000000000000000000000000000000000.0
112000000000000000000000000000000000000000000000.0
125000000000000000000000000000000000000000000000.0
140000000000000000000000000000000000000000000000.0
160000000000000000000000000000000000000000000000.0
180000000000000000000000000000000000000000000000.0
200000000000000000000000000000000000000000000000.0
220000000000000000000000000000000000000000000000.0
250000000000000000000000000000000000000000000000.0
280000000000000000000000000000000000000000000000.0
320000000000000000000000000000000000000000000000.0
360000000000000000000000000000000000000000000000.0
400000000000000000000000000000000000000000000000.0
450000000000000000000000000000000000000000000000.0
500000000000000000000000000000000000000000000000.0
560000000000000000000000000000000000000000000000.0
630000000000000000000000000000000000000000000000.0
710000000000000000000000000000000000000000000000.0
800000000000000000000000000000000000000000000000.0
900000000000000000000000000000000000000000000000.0
1000000000000000000000000000000000000000000000000.0
1120000000000000000000000000000000000000000000000.0
1250000000000000000000000000000000000000000000000.0
1400000000000000000000000000000000000000000000000.0
1600000000000000000000000000000000000000000000000.0
1800000000000000000000000000000000000000000000000.0
2000000000000000000000000000000000000000000000000.0
2200000000000000000000000000000000000000000000000.0
2500000000000000000000000000000000000000000000000.0
2800000000000000000000000000000000000000000000000.0
3200000000000000000000000000000000000000000000000.0
3600000000000000000000000000000000000000000000000.0
4000000000000000000000000000000000000000000000000.0
4500000000000000000000000000000000000000000000000.0
5000000000000000000000000000000000000000000000000.0
5600000000000000000000000000000000000000000000000.0
6300000000000000000000000000000000000000000000000.0
7100000000000000000000000000000000000000000000000.0
8000000000000000000000000000000000000000000000000.0
9000000000000000000000000000000000000000000000000.0
10000000000000000000000000000000000000000000000000.0
11200000000000000000000000000000000000000000000000.0
12500000000000000000000000000000000000000000000000.0
14000000000000000000000000000000000000000000000000.0
16000000000000000000000000000000000000000000000000.0
18000000000000000000000000000000000000000000000000.0
20000000000000000000000000000000000000000000000000.0
22000000000000000000000000000000000000000000000000.0
25000000000000000000000000000000000000000000000000.0
28000000000000000000000000000000000000000000000000.0
32000000000000000000000000000000000000000000000000.0
36000000000000000000000000000000000000000000000000.0
40000000000000000000000000000000000000000000000000.0
45000000000000000000000000000000000000000000000000.0
50000000000000000000000000000000000000000000000000.0
56000000000000000000000000000000000000000000000000.0
63000000000000000000000000000000000000000000000000.0
71000000000000000000000000000000000000000000000000.0
80000000000000000000000000000000000000000000000000.0
90000000000000000000000000000000000000000000000000.0
100000000000000000000000000000000000000000000000000.0
112000000000000000000000000000000000000000000000000.0
125000000000000000000000000000000000000000000000000.0
140000000000000000000000000000000000000000000000000.0
160000000000000000000000000000000000000000000000000.0
180000000000000000000000000000000000000000000000000.0
200000000000000000000000000000000000000000000000000.0
220000000000000000000000000000000000000000000000000.0
250000000000000000000000000000000000000000000000000.0
280000000000000000000000000000000000000000000000000.0
320000000000000000000000000000000000000000000000000.0
360000000000000000000000000000000000000000000000000.0
400000000000000000000000000000000000000000000000000.0
450000000000000000000000000000000000000000000000000.0
500000000000000000000000000000000000000000000000000.0
560000000000000000000000000000000000000000000000000.0
630000000000000000000000000000000000000000000000000.0
710000000000000000000000000000000000000000000000000.0
800000000000000000000000000000000000000000000000000.0
900000000000000000000000000000000000000000000000000.0
1000000000000000000000000000000000000000000000000000.0
1120000000000000000000000000000000000000000000000000.0
1250000000000000000000000000000000000000000000000000.0
1400000000000000000000000000000000000000000000000000.0
1600000000000000000000000000000000000000000000000000.0
1800000000000000000000000000000000000000000000000000.0
2000000000000000000000000000000000000000000000000000.0
2200000000000000000000000000000000000000000000000000.0
2500000000000000000000000000000000000000000000000000.0
2800000000000000000000000000000000000000000000000000.0
3200000000000000000000000000000000000000000000000000.0
3600000000000000000000000000000000000000000000000000.0
4000000000000000000000000000000000000000000000000000.0
4500000000000000000000000000000000000000000000000000.0
5000000000000000000000000000000000000000000000000000.0
5600000000000000000000000000000000000000000000000000.0
6300000000000000000000000000000000000000000000000000.0
7100000000000000000000000000000000000000000000000000.0
8000000000000000000000000000000000000000000000000000.0
9000000000000000000000000000000000000000000000000000.0
10000000000000000000000000000000000000000000000000000.0
11200000000000000000000000000000000000000000000000000.0
12500000000000000000000000000000000000000000000000000.0
14000000000000000000000000000000000000000000000000000.0
16000000000000000000000000000000000000000000000000000.0
18000000000000000000000000000000000000000000000000000.0
20000000000000000000000000000000000000000000000000000.0
22000000000000000000000000000000000000000000000000000.0
25000000000000000000000000000000000000000000000000000.0
28000000000000000000000000000000000000000000000000000.0
32000000000000000000000000000000000000000000000000000.0
36000000000000000000000000000000000000000000000000000.0
40000000000000000000000000000000000000000000000000000.0
45000000000000000000000000000000000000000000000000000.0
50000000000000000000000000000000000000000000000000000.0
56000000000000000000000000000000000000000000000000000.0
63000000000000000000000000000000000000000000000000000.0
71000000000000000000000000000000000000000000000000000.0
80000000000000000000000000000000000000000000000000000.0
90000000000000000000000000000000000000000000000000000.0
100000000000000000000000000000000000000000000000000000.0
112000000000000000000000000000000000000000000000000000.0
125000000000000000000000000000000000000000000000000000.0
140000000000000000000000000000000000000000000000000000.0
160000000000000000000000000000000000000000000000000000.0
180000000000000000000000000000000000000000000000000000.0
200000000000000000000000000000000000000000000000000000.0
220000000000000000000000000000000000000000000000000000.0
250000000000000000000000000000000000000000000000000000.0
280000000000000000000000000000000000000000000000000000.0
320000000000000000000000000000000000000000000000000000.0
360000000000000000000000000000000000000000000000000000.0
400000000000000000000000000000000000000000000000000000.0
450000000000000000000000000000000000000000000000000000.0
500000000000000000000000000000000000000000000000000000.0
560000000000000000000000000000000000000000000000000000.0
630000000000000000000000000000000000000000000000000000.0
710000000000000000000000000000000000000000000000000000.0
800000000000000000000000000000000000000000000000000000.0
900000000000000000000000000000000000000000000000000000.0
1000000000000000000000000000000000000000000000000000000.0
1120000000000000000000000000000000000000000000000000000.0
1250000000000000000000000000000000000000000000000000000.0
1400000000000000000000000000000000000000000000000000000.0
1600000000000000000000000000000000000000000000000000000.0
1800000000000000000000000000000000000000000000000000000.0
2000000000000000000000000000000000000000000000000000000.0
2200000000000000000000000000000000000000000000000000000.0
2500000000000000000000000000000000000000000000000000000.0
2800000000000000000000000000000000000000000000000000000.0
3200000000000000000000000000000000000000000000000000000.0
3600000000000000000000000000000000000000000000000000000.0
4000000000000000000000000000000000000000000000000000000.0
4500000000000000000000000000000000000000000000000000000.0
5000000000000000000000000000000000000000000000000000000.0
5600000000000000000000000000000000000000000000000000000.0
6300000000000000000000000000000000000000000000000000000.0
7100000000000000000000000000000000000000000000000000000.0
8000000000000000000000000000000000000000000000000000000.0
9000000000000000000000000000000000000000000000000000000.0
10000000000000000000000000000000000000000000000000000000.0
11200000000000000000000000000000000000000000000000000000.0
12500000000000000000000000000000000000000000000000000000.0
14000000000000000000000000000000000000000000000000000000.0
16000000000000000000000000000000000000000000000000000000.0
18000000000000000000000000000000000000000000000000000000.0
20000000000000000000000000000000000000000000000000000000.0
22000000000000000000000000000000000000000000000000000000.0
25000000000000000000000000000000000000000000000000000000.0
28000000000000000000000000000000000000000000000000000000.0
32000000000000000000000000000000000000000000000000000000.0
36000000000000000000000000000000000000000000000000000000.0
40000000000000000000000000000000000000000000000000000000.0
45000000000000000000000000000000000000000000000000000000.0
50000000000000000000000000000000000000000000000000000000.0
56000000000000000000000000000000000000000000000000000000.0
63000000000000000000000000000000000000000000000000000000.0
71000000000000000000000000000000000000000000000000000000.0
80000000000000000000000000000000000000000000000000000000.0
90000000000000000000000000000000000000000000000000000000.0
100000000000000000000000000000000000000000000000000000000.0
112000000000000000000000000000000000000000000000000000000.0
125000000000000000000000000000000000000000000000000000000.0
140000000000000000000000000000000000000000000000000000000.0
160000000000000000000000000000000000000000000000000000000.0
180000000000000000000000000000000000000000000000000000000.0
200000000000000000000000000000000000000000000000000000000.0
220000000000000000000000000000000000000000000000000000000.0
250000000000000000000000000000000000000000000000000000000.0
280000000000000000000000000000000000000000000000000000000.0
320000000000000000000000000000000000000000000000000000000.0
360000000000000000000000000000000000000000000000000000000.0
400000000000000000000000000000000000000000000000000000000.0
450000000000000000000000000000000000000000000000000000000.0
500000000000000000000000000000000000000000000000000000000.0
560000000000000000000000000000000000000000000000000000000.0
630000000000000000000000000000000000000000000000000000000.0
710000000000000000000000000000000000000000000000000000000.0
800000000000000000000000000000000000000000000000000000000.0
900000000000000000000000000000000000000000000000000000000.0
1000000000000000000000000000000000000000000000000000000000.0
1120000000000000000000000000000000000000000000000000000000.0
1250000000000000000000000000000000000000000000000000000000.0
1400000000000000000000000000000000000000000000000000000000.0
1600000000000000000000000000000000000000000000000000000000.0
1800000000000000000000000000000000000000000000000000000000.0
2000000000000000000000000000000000000000000000000000000000.0
2200000000000000000000000000000000000000000000000000000000.0
2500000000000000000000000000000000000000000000000000000000.0
2800000000000000000000000000000000000000000000000000000000.0
3200000000000000000000000000000000000000000000000000000000.0
3600000000000000000000000000000000000000000000000000000000.0
4000000000000000000000000000000000000000000000000000000000.0
4500000000000000000000000000000000000000000000000000000000.0
5000000000000000000000000000000000000000000000000000000000.0
5600000000000000000000000000000000000000000000000000000000.0
6300000000000000000000000000000000000000000000000000000000.0
7100000000000000000000000000000000000000000000000000000000.0
8000000000000000000000000000000000000000000000000000000000.0
9000000000000000000000000000000000000000000000000000000000.0
10000000000000000000000000000000000000000000000000000000000.0
11200000000000000000000000000000000000000000000000000000000.0
12500000000000000000000000000000000000000000000000000000000.0
14000000000000000000000000000000000000000000000000000000000.0
16000000000000000000000000000000000000000000000000000000000.0
18000000000000000000000000000000000000000000000000000000000.0
20000000000000000000000000000000000000000000000000000000000.0
22000000000000000000000000000000000000000000000000000000000.0
25000000000000000000000000000000000000000000000000000000000.0
28000000000000000000000000000000000000000000000000000000000.0
32000000000000000000000000000000000000000000000000000000000.0
36000000000000000000000000000000000000000000000000000000000.0
40000000000000000000000000000000000000000000000000000000000.0
45000000000000000000000000000000000000000000000000000000000.0
50000000000000000000000000000000000000000000000000000000000.0
56000000000000000000000000000000000000000000000000000000000.0
63000000000000000000000000000000000000000000000000000000000.0
71000000000000000000000000000000000000000000000000000000000.0
80000000000000000000000000000000000000000000000000000000000.0
90000000000000000000000000000000000000000000000000000000000.0
100000000000000000000000000000000000000000000000000000000000.0
112000000000000000000000000000000000000000000000000000000000.0
125000000000000000000000000000000000000000000000000000000000.0
140000000000000000000000000000000000000000000000000000000000.0
160000000000000000000000000000000000000000000000000000000000.0
180000000000000000000000000000000000000000000000000000000000.0
200000000000000000000000000000000000000000000000000000000000.0
220000000000000000000000000000000000000000000000000000000000.0
250000000000000000000000000000000000000000000000000000000000.0
280000000

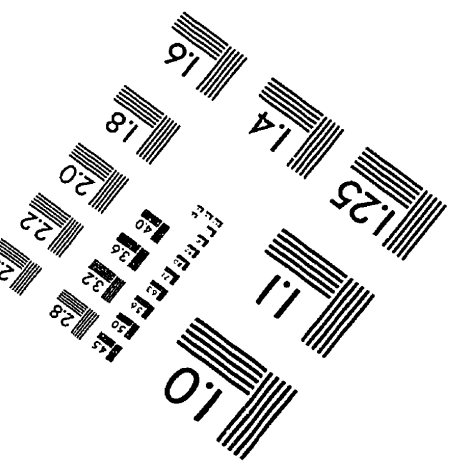
# IMAGE EVALUATION
# TEST TARGET (QA-3)











APPLIED IMAGE . Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989