High Performance Dynamics Desi
of Arithmetic Circuits

Zaher Owda

MASTER THESIS

—♦—

Ioannina, July 2010

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF IOANNINA

ΥΨΗΛΗΣ ΑΠΟΔΟΣΗΣ ΔΥΝΑΜΙΚΗ ΣΧΕΔΙΑΣΗ ΑΡΙΘΜΗΤΙΚΩΝ ΚΥΚΛΩΜΑΤΩΝ

Η
ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΕΞΕΙΔΙΚΕΥΣΗΣ

Υποβάλλεται στην

ορισθείσα από την Γενική Συνέλευση Ειδικής Σύνθεσης
του Τμήματος Πληροφορικής
Εξεταστική Επιτροπή

από τον

ZAHER OWDA

ως μέρος των Υποχρεώσεων

για τη λήψη

του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΣΤΗΝ ΠΛΗΡΟΦΟΡΙΚΗ

ΜΕ ΕΞΕΙΔΙΚΕΥΣΗ ΣΤΙΣ ΤΕΧΝΟΛΟΓΙΕΣ-ΕΦΑΡΜΟΓΕΣ

Ιούλιος 2010

HIGH PERFORMANCE DYNAMIC DESIGNS OF ARITHMETIC CIRCUITS

MASTER THESIS

ZAHER OWDA

POST GRADUATE DIPLOMA IN COMPUTER SCIENCE

IN APPLICATIONS AND TECHNOLOGIES

July 2010

DEDICATION

This thesis is dedicated to my parents.

ACKNOWLEDGEMENTS

I need to thank my family and friends for their unabated support, their encouragement, patience and understanding.

I am most grateful to my supervisor, Assist. Prof. Yiorgos Tsiatouhas for his guidance throughout this thesis and for giving me the opportunity to work on an interesting subject, such as this one.

Special thanks to the Assist. Prof. Themistoklis Haniotakis from the University of Patras and Prof. Kostas Efstathiou from the TEI of Athens for the productive cooperation we had on dynamic circuit and the Manchester adder designs respectively.

I would like to extend my deepest expressions of gratitude to all the people that have assisted me in the completion of this thesis.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Zaher Owda, MSc, Computer Science Department, University of Ioannina,Greece. February, 2010. High Performance Dynamic Designs Of Arithmetic Circuits.

Thesis Supervisor: Tsiatouhas Yiorgos.

A desirable characteristic of VLSI circuits is high speed operation. The use of dynamic circuit design techniques can provide high speed operation at lower silicon area requirements, compared to full static CMOS designs. Another common design technique in order to achieve high operating speed is the use of pipeline schemes. However, the higher the required operating frequency, the higher the number of stages we must implement in the pipeline. In addition, a limiting factor in cases with a large number of stages, are the restrictions imposed from the required memory elements. These memory elements not only increase the silicon area of the implementation but also restrict the maximum achievable frequency due to their internal delays. In this Thesis, a memory-less pipeline design style is proposed, where the combinational part is implemented with dynamic circuits that offer the desirable high speed operation while the memory elements are eliminated due to an intelligent clocking scheme. Thus, the proposed design technique provide the advantage of high performance operation and at the same time compares favourably to pre-existing approaches with respect to silicon overhead and power requirements. According to the experimental results on dynamic designs of the Kogge-Stone carry-lookahead adder topology, the proposed technique can improve the propagation delay of the evaluation phase up to 76.96% and 59.11% over the pertinent Domino and Wave Domino dynamic designs.

Furthermore, an efficient implementation of an 8-bit Manchester carry chain adder in multi-output domino CMOS logic is introduced in this work. The carries of this adder are computed in parallel by two independent 4-bit carry chains, one for the odd and one for the even carries respectively. This adder module can be used for the

implementation of wider adders leading to significant operating speed improvement compared to the corresponding adders based on alternative Manchester carry chain adder modules proposed in the open literature. The simulation results on a 64-bit Manchester adder provided propagation delay improvements up to 35.08% over earlier designs.

ΕΚΤΕΤΑΜΕΝΗ ΠΕΡΙΛΗΨΗ

Zaher Owda, MSc, Τμήμα Πληροφορικής, Πανεπιστήμιο Ιωαννίνων, Ιούλιος 2010.

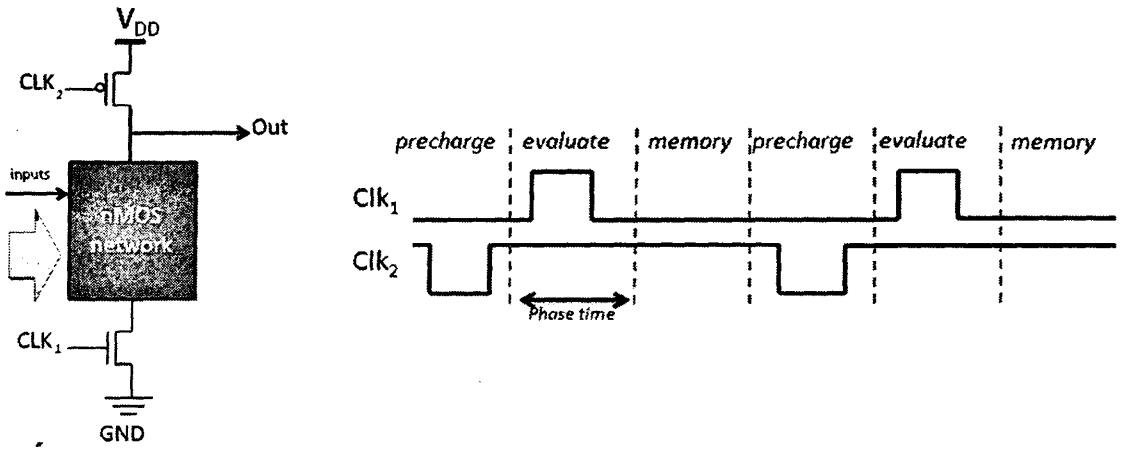
Υψηλής Απόδοσης Δυναμική Σχεδίαση Αριθμητικών Κυκλωμάτων.

Επιβλέπων : Τσιατούχας Γεώργιος.

Ένα επιθυμητό χαρακτηριστικό των VLSI κυκλωμάτων είναι η λειτουργία με υψηλές ταχύτητες. Η χρήση δυναμικών τεχνικών σχεδιασμού κυκλωμάτων μπορεί να παρέχουν υψηλής ταχύτητας λειτουργία εκμαιεύοντας μικρότερη επιφάνεια πυριτίου, σε σύγκριση με την πλήρη στατική CMOS τεχνική σχεδίασης. Μια άλλη κοινή τεχνική σχεδιασμού ώστε να επιτευχθούν υψηλές ταχύτητες λειτουργίας είναι η χρήση δομών διοχέτευσης. Ωστόσο, όσο μεγαλύτερη είναι η απαιτούμενη συχνότητα λειτουργίας, τόσο μεγαλύτερος είναι ο αριθμός των σταδίων που θα πρέπει να εφαρμόσουμε στην δομή διοχέτευσης. Επιπλέον, ένας περιοριστικός παράγοντας σε υλοποιήσεις με μεγάλο αριθμό σταδίων, είναι οι περιορισμοί που επιβάλλονται από τα απαιτούμενα στοιχεία μνήμης. Αυτά τα στοιχεία μνήμης δεν αυξάνουν μόνο την επιφάνεια του πυριτίου της υλοποιούμενης δομής, αλλά επίσης περιορίζουν και την μέγιστη εφικτή συχνότητα λειτουργίας, λόγω των εσωτερικών τους καθυστερήσεων. Σε αυτή την εργασία, προτείνεται μια τεχνική σχεδίασης δομών διοχέτευσης χωρίς την χρήση στοιχείων μνήμης, όπου το συνδυαστικό μέρος υλοποιείται με δυναμικά κυκλώματα που προσφέρουν υψηλή ταχύτητα λειτουργίας, ενώ τα στοιχεία μνήμης έχουν εξαλειφθεί με την χρήση ενός ευφυούς συστήματος χρονισμού. Έτσι, η προτεινόμενη τεχνική παρέχει το πλεονέκτημα της υψηλής απόδοσης λειτουργίας και ταυτόχρονα μικρότερη κατανάλωση ισχύος και μείωση της επιφάνειας πυριτίου σε σύγκριση με προϋπάρχουσες υψηλών επιδόσεων τεχνικές.

Αρχικός σκοπός μας είναι η δημιουργία μιας δυναμικής λογικής οικογένειας, όπου σε κάθε πύλη θα ενσωματώνεται η δυνατότητα λειτουργίας της και ως μνήμης. Για να το επιτύχουμε αυτό εισάγουμε ένα δεύτερο ρολόι στην λειτουργία μιας δυναμικής πύλης

κατά τρόπο τέτοιο ώστε το PMOS τρανζίστορ προφόρτισης και το NMOS τρανζίστορ υπολογισμού να οδηγούνται από διαφορετικό σήμα ρολογιού. Ο νέος τρόπος λειτουργίας δεν απαιτεί την παρουσία του αντιστροφέα, καθότι επιτυγχάνεται η εξάλειψη των συνθηκών ανταγωνισμού (*race conditions*), που ταλαιπωρούν την τυπική δυναμική λογική. Στο σχήμα που ακολουθεί, παρουσιάζονται η τοπολογία της προτεινόμενης δυναμικής λογικής και οι κυματομορφές των δύο σημάτων ρολογιού με τις οποίες καταφέρνουμε να εισαγάγουμε την επιπλέων φάση μνήμης στη λειτουργία της πύλης.

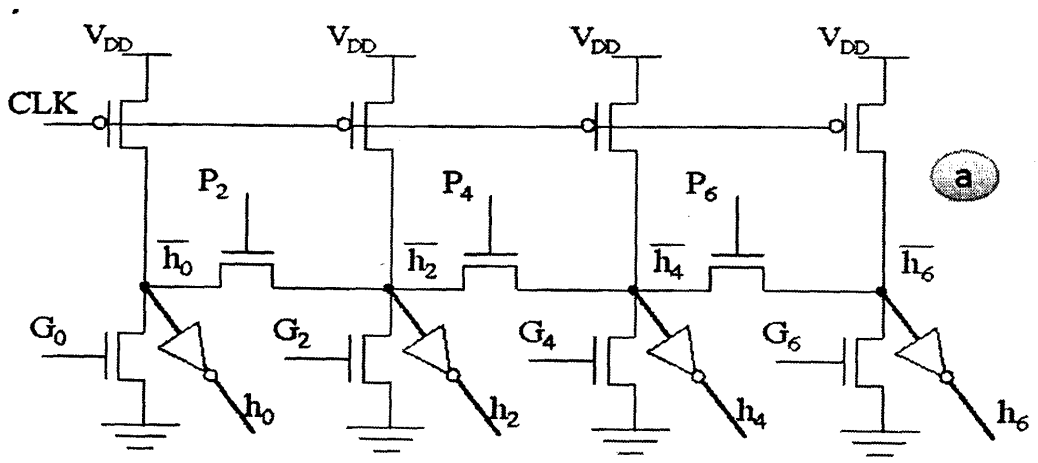


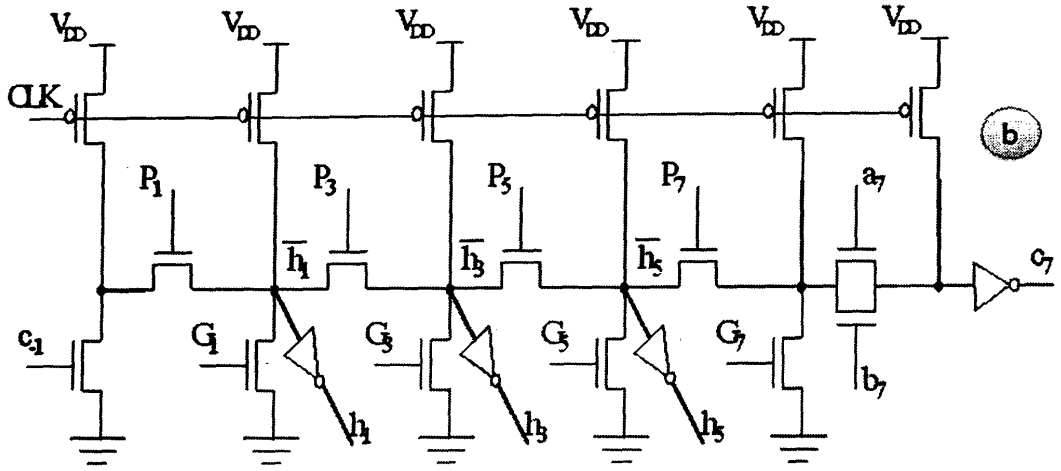
Η τοπολογία της προτεινόμενης δυναμικής λογικής και οι τρεις φάσεις λειτουργίας με τη χρήση δύο σημάτων ρολογιού.

Η προτεινόμενη δυναμική λογική σχεδίασης εφαρμόστηκε στην σχεδίαση της βαθμίδας πρόβλεψης κρατουμένου ενός 16-bit Kogge-Stone, αθροιστή με χρήση έξι φάσεων ρολογιού (τρία ρολόγια και τα συμπληρώματά τους). Η νέα σχεδίαση προσφέρει σύμφωνα με τα αποτελέσματα των προσομοιώσεων, βελτίωση στην συχνότητα λειτουργίας μέχρι και 76.96% σε σχέση με την τυπική Domino σχεδίαση και μέχρι 59,11% σε σχέση με την πολύ υψηλών επιδόσεων Wave Domino τεχνική σχεδίασης. Η βελτίωση στην ταχύτητα λειτουργίας έχει ένα κόστος στην μέση κατανάλωση ενέργειας μέχρι 28.49% ως προς την τυπική Domino και μείωση στην κατανάλωση κατά 29,54% ως προ της Wave Domino, όμως το γινόμενο ενέργεια × κύκλος ρολογιού βελτιώνεται και στις δύο περιπτώσεις πάνω από 70%. Οι προηγούμενες επιδόσεις απαίτησαν επιφάνεια πυριτίου αυξημένη κατά 8,7% ως προς την τυπική Domino και μειωμένη κατά 9% ως προς τη Wave Domino τεχνική.

Ακολούθως στην εργασία, παρουσιάζεται μια αποτελεσματική δυναμική υλοποίηση του Manchester αθροιστή με χρήση πολλαπλών εξόδων Domino CMOS λογικής. Τα κρατούμενα του αθροιστή υπολογίζονται παράλληλα και από δύο ανεξάρτητες αλυσίδες κρατουμένου. Αυτό επιφέρει σημαντική βελτίωση της ταχύτητας λειτουργίας σε σύγκριση με τις αντίστοιχες τοπολογίες υλοποίησης αθροιστών οι οποίες βασίζονται στη Manchetser δομή.

Η καινούρια προτεινόμενη τοπολογία βασίζεται στις εξισώσεις του Manchester αθροιστή μετασχηματίζοντάς τες έτσι ώστε να δίνεται η δυνατότητα υπολογισμού των άρτιων κρατουμένων παράλληλα με τα περιττά. Αυτός ο διαχωρισμός επιτρέπει π.χ. την υλοποίηση του 8-bit Manchester αθροιστή με δυο ανεξάρτητες παράλληλες αλυσίδες των 4-bit, όπου η πρώτη αλυσίδα υπολογίζει τα άρτια κρατούμενα και η δεύτερη υπολογίζει τα περιττά. Γίνεται φανερό πως ο υπολογισμός των κρατουμένων επιταχύνεται σημαντικά με τη χρήση της νέας τεχνικής. Στο σχήμα που ακολουθεί παρουσιάζονται οι δυο αλυσίδες ενός 8-bit Manchester αθροιστή για τον υπολογισμό των νέων κρατούμενων h_0-h_6 και του τελικού κρατούμενου εξόδου c_7.





α) Η αλυσίδα των άρτιων κρατούμενων. β) Η αλυσίδα των περιττών κρατούμενων

Τα αποτελέσματα των προσομοιώσεων των Manchester σχεδιασμών σύμφωνα με την προτεινόμενη τεχνική ήταν εντυπωσιακά σε ότι αφορά την ταχύτητα. Η καθυστέρηση διάδοσης μειώνεται έως και 35,08% σε σχέση με τη συμβατική τοπολογία ενός 64-bit Manchester αθροιστή. Η νέα τοπολογία επιτυγχάνει υψηλές επιδόσεις στην ταχύτητα λειτουργίας, πληρώνοντας ωστόσο το απαιτούμενο κόστος σε καταναλισκόμενη ενέργεια ανά κύκλο ρολογιού, το οποίο μεταφράζεται σε αύξηση 44.29% συγκριτικά με το συμβατικό 64-bit Manchester αθροιστή. Όμως το γινόμενο ενέργεια × κύκλο ρολογιού είναι αυξημένο μόνο κατά 14,37% με τάσεις μείωσης όσο αυξάνουν τα bit του αθροιστή.

CHAPTER 1 INTRODUCTION

CHAPTER 1. INTRODUCTION

1.1. Scope

Over decades, the semiconductor industry is continuously increasing its research efforts to provide higher performance microelectronic circuits and systems. Dynamic circuit design techniques can provide the desirable high speed operation at lower silicon area requirements compared to other CMOS design techniques.
The scope in this thesis is to present new high performance dynamic design techniques suitable for the implementation of high speed arithmetic circuits, where the pertinent demands are crucial, and to compare them with existing solutions in the open literature.

1.2. Manuscript organization

This manuscript is organized as follows. In chapter 2 an introduction to the CMOS logic design is provided, where the complementary static CMOS logic and the common dynamic and Domino CMOS logic design styles are discussed.

Next, in chapter 3 an introduction to the theory of CMOS adders design is given, that is followed by a detailed presentation of the Carry Lookahead adder

topologies, especially focusing on the Kooge-Stone Lookahead adder as well as on the Manchester Carry Chain adder.

A new dynamic design technique and its enhanced performance version is introduced and analyzed in chapter 4. This technique is applied on a Kooge-Stone adder design and simulation results are provided in comparison to the corresponding Domino and Wave Domino designs.

Next, in chapter 5, the architecture of a new high performance double carry chain Manchester adder is presented, which is based on a multi output Domino topology. Comparisons among the proposed Manchester carry chain design and earlier Manchester topologies in the open literature are discussed. Finally, the conclusions are drawn in chapter 6.

CHAPTER 2. CMOS LOGIC DESIGN

2.1. Introduction to CMOS

Complementary Metal Oxide Semiconductor (CMOS) is a technology for integrated circuits construction; CMOS technology is used in a wide range of circuit designs such as microcontrollers, microprocessors, static RAM, and digital logic design [3-7]. An important characteristic of this technology is the low static power consumption, compared to earlier technologies, since power is mainly drown only when the internal circuit nodes are changing state. Moreover, CMOS technology permits the implementation of high density logic functions in a chip.

CMOS circuits use both types of semiconductor field effect transistors, the PMOS (positive polarity) and the NMOS (negative polarity) transistors, in order to implement logic gates and consequently digital circuits. The most common measures to evaluate a circuit design are: the surface, the speed (performance), the energy consumption, the reliability and the generated noise [1]. Considering the above measures, static CMOS design offers low noise sensitivity and high reliability at acceptable speeds and relatively low power consumption.

2.2. Complementary Static CMOS logic

The circuits that are designed using CMOS technology are separated into two categories depending on whether they store or not a previous response of the circuit as a subsequent input: the combinational logic circuits, and the sequential logic circuits [1]. In combinational logic the output is defined by the current input signals, without any type of feedback from the output to the circuit input. On the contrary, the output of the sequential circuits depends on both the current input and the pervious response of the circuit (which is called "circuit state"). Consequently, the circuit consists of a combinational logic part and a register which holds the circuit state.

A static CMOS gate is a combination of two networks, the pull up network that consists of pMOS transistors and it is called pMOS network and the pull down network which is composed of nMOS transistors and is called nMOS network, as it is shown in Fig. 2.1. These two networks are structured in a mutually executive fashion such that one and only one of the networks is conducting in steady state. In this way, once the transients have settled, a path always exists between V_{dd} and the output F for a high output "1", or between Gnd (ground) and F for a low output "0". This is equivalent to stating that the output node is always a low-impedance node in steady state.

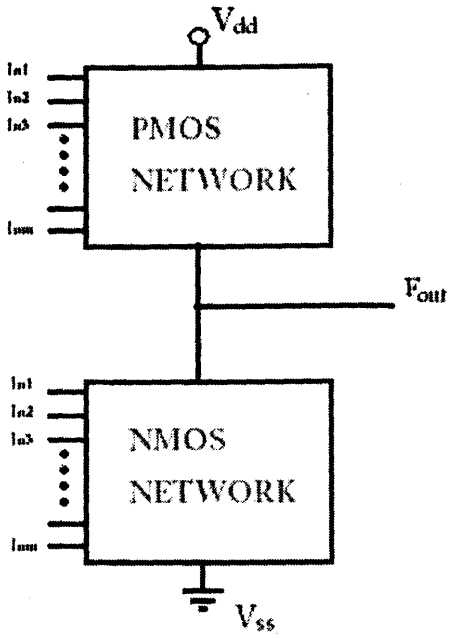


Figure 2.1 Static CMOS gate

2.3. Dynamic CMOS logic

An alternative use of CMOS technology in digital circuit design, targeting to provide increased performance, is the dynamic logic [1], [2]. According to the typical dynamic design style, the gate output is periodically precharged to high through a single pMOS control transistor (precharge transistor). This phase in the circuit operation is called precharge phase. Then, in-between the precharge phases, an nMOS network is exploited to calculate the gate response according to the input data. In case that a logic low value is required at the output an active path in the nMOS network discharges the output while in case that a logic high value is required no path is formed in the nMOS network to discharge the output which simply remains charged to V_{DD}. This phase in the circuit operation is called

evaluation phase. An additional nMOS control transistor (evaluation transistor) isolates the nMOS network from the ground and ensures that no discharge path is formed through the nMOS network during the precharge phase. During the evaluation phase the pMOS precharge transistor is inactive. Thus, during this phase, no low to high transitions can take place at the output. This implies that during the evaluation phase if an input combination discharges the output, the latter will remain discharged regardless of the input combinations that may follow during the same evaluation phase. Consequently, we must ensure that only a single and valid input combination is applied during each evaluation phase. A clock signal is used to drive the control transistors and form the two circuit operating phases.

In Fig. 2.2 the topology of a dynamic gate is presented. During the precharge phase, CLK value becomes "0" and so the output is precharged to V_{dd} independently of the input values, because the evaluation transistor is turned off. When the CLK turned to "1" during the evaluation, a conducting path is created between the output and the Gnd (ground) ground if the function that has been implemented in the nMOS network is true. Otherwise, the output remains at the precharged state of V_{dd}.

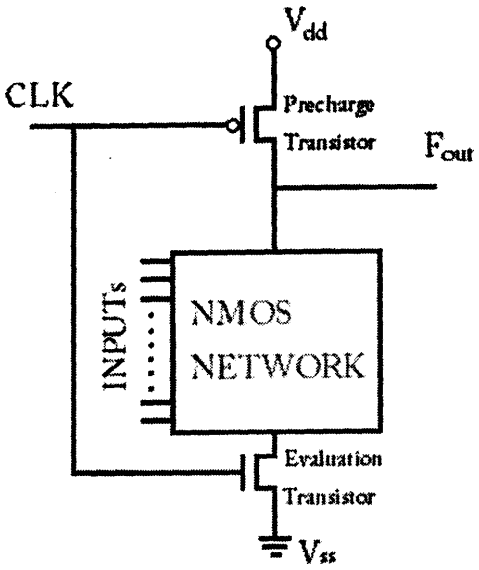


Figure 2.2 Dynamic CMOS gate

Next, the main attributes of the dynamic gates are discussed:

The logical function is realised by the pull down network, which is designed the same way as in the static CMOS.

The number of transistors is almost half the corresponding number in static CMOS.

The size of the PMOS device is not important for the functionality of the gate.

High dynamic power consumption is reported, due to the clock signal CLK switching activity.

Faster switching speed is observed due to the decreased gate capacitance as a result of the small number of transistors (low input load) and the absence of short circuit current, because all the current that goes through the nMOS network mainly concerns the discharge of the output.

Consequently, the basic advantages of the dynamic logic are the increased speed and the reduced silicon area requirements with respect to the static CMOS logic [9], [10]. However the speed is affected by the presence of the evaluation transistor, which is used to prevent short-circuit power consumption.

2.3.1. Signal Integrity

Aiming to exploit the high performance efficiency of dynamic logic we have to consider a number of critical design issues, like leakage currents, charge sharing, capacitive coupling and clock power consumption, in order to ensure that the dynamic logic operates properly [1].

2.3.1.1. Leakage Current

The operation of a dynamic gate is based on the dynamic storing of the output value in a capacitor. Consequently, if the nMOS network isn't conductive in an ideal circuit the output must retain the precharge value during the evaluation phase. However, the normal transistor leakage current may lead to an erroneous circuit operation. The charge that is stored at the output capacitance leaks due to the various leakage current mechanisms, like the parasitic diode reverse current or the subthreshold leakage current of the transistors in the nMOS network. Consequently, a minimum clock rate must be guaranteed although this could not be characterized as a robust design. The problem worsens in the modern CMOS nanotechnologies.

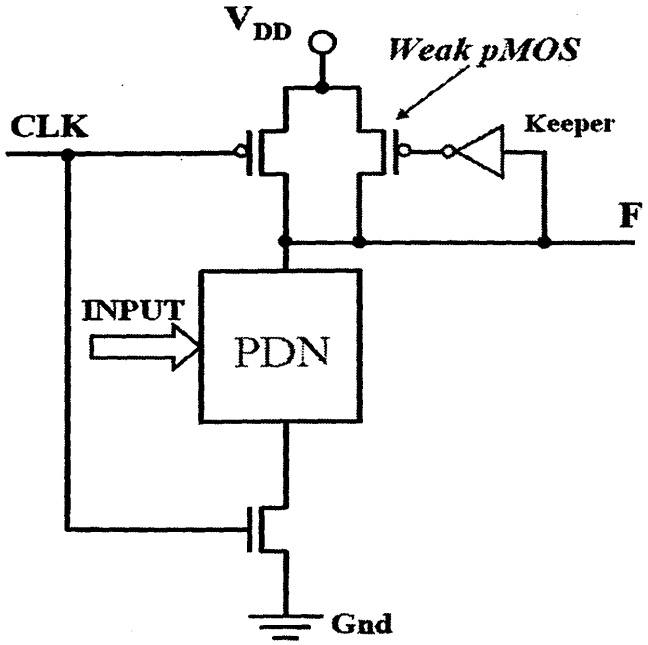


Figure 2.3 Adding a keeper to face leakage current

In order to confront the leak problem, a reduction of the output impedance, in the output node is suggested, during the evaluation. This is achieved by adding a feeder transistor (see Fig. 2.3). Its operation is to compensate for the charge loss due to the pull down leakage paths. To avoid the pull-up and pull-down network ratio problems associated with this style of circuits and the associated static power consumption, the keeper's resistance is made high (use of a high L). This allows the pull down devices to discharge the output node substantially below the switching threshold of the NOT gate and switch off the keeper.

2.3.1.2. Charge sharing

A subject of major importance concerning the design of dynamic logic is the charge redistribution (sharing). During the precharge phase the output node is set to high. Next assume that during the evaluation phase no conducting path is form

in the nMOS network from the output node to the ground. However, many paths may be formed from the output node to internal nodes of the nMOS network, depending on the input values. This may lead to charge redistribution between the parasitic capacitance of the output node and the internal parasitic capacitances of the nMOS network, which will reduce the voltage of the output node and cause reliability problems in the circuit operation. The most effective way to confront this situation is to precharge (during the precharge phase) important (high capacitance) internal nodes in order to prevent charge redistribution, although this will increase the cost and power consumption and will decrease the circuit performance.

2.3.1.3. Capacitive coupling

The rather high impedance of the output node exposes the circuit to the influence of capacitive coupling. Capacitive coupling appears when a capacitance exists between two signal lines. In dynamic logic serious capacity coupling can occur between the dynamic node and a wire routed over or next to the dynamic node. This may corrupt the logic state of the dynamic node especially when it is in a floating condition.

2.3.1.4. Clock Feedthrough

The feedthrough of the clock signal is a special case of capacitive coupling, which is related to the parasitic capacitance between the gate of the precharge transistor, which is fed by the clock signal, and the dynamic output node. This capacitive coupling makes the output signal of the dynamic node to rise above the value of V_{DD} during a low to high transition of the clock (Fig. 2.4), when the nMOS network is not in a conducting state. Consequently, the fast rising and falling edges of the clock couple onto the signal node, as it is shown in the simulation graph of

Fig. 2.4. The danger of the clock feedthrough is that it may force the normally reverse-biased junction diodes of the pMOS precharge transistor to become forward biased causing the injection of charges from the floating drain node to the substrate. This reduces the noise margins and may lead the circuit to an erroneous operation.

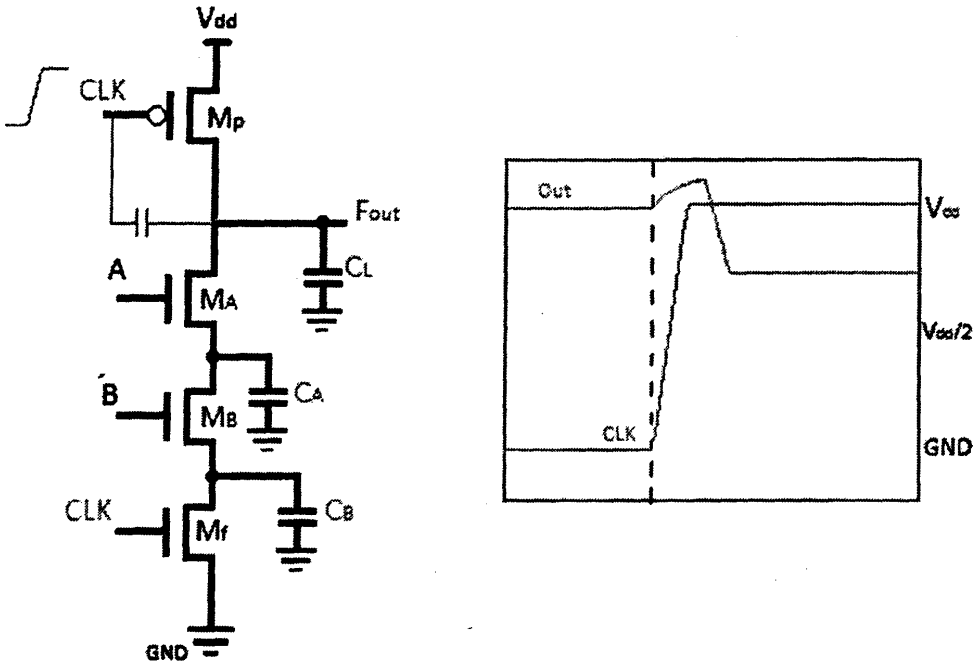


Figure 2.4 Demonstrating clock feedthrough effect

2.3.2. Cascading dynamic gates

Besides the signal integrity issues, there is a major problem that complicate the design of the dynamic circuits: straightforward cascading of dynamic gates to create multilevel logic structures does not work.

This problem exists because the output of the each gate of a level (and thus the input of the next level) is precharged to high. This may result to the unintended discharge of the output at the beginning of the evaluation cycle due to race conditions. Changing all the input values to "0" during the precharge phase solves this issue. In this way we deactivate the transistors of the nMOS network after the precharge and we avoid the unintended discharge of the output node during the evaluation phase. The conclusion is that the input can make only one transition from "0" to "1" during the phase of evaluation in order to ensure the correct operation of the circuit.

Race conditions during the evaluation phase, when a dynamic gate drives another dynamic gate, are a known problem of dynamic logic. In that case the precharged node of the first gate can discharge the output of the following gate before the first gate is correctly evaluated [1]. In order to overcome this problem, the most common design technique is the Domino logic family.

2.3.3. Domino logic

Domino logic gates are composed of a dynamic gate which is followed by a static inverter, as it is shown in Fig. 2.5 [17]. As in any dynamic logic there are two phases in the Domino CMOS circuits operation:

Precharge phase: When the clock CLK signal is low, the PMOS precharge transistor is conductive and charges the dynamic node, which gives a logical "0" at the output of the inverter.

Evaluation phase: When the clock CLK signal is high, the PMOS transistor isn't conductive, while the NMOS evaluation transistor is conductive. This transistor allows the discharge or not of the dynamic node depending on the input values of the NMOS network. As a result the output turns to high.

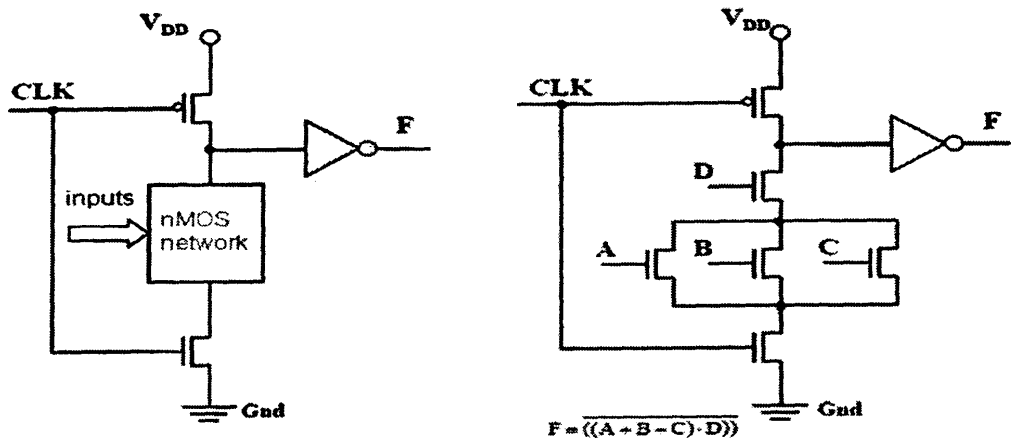


Figure 2.5 Domino CMOS logic

After evaluation, a Domino gate must be precharged before it can be used for a subsequent evaluation in the next clock cycle. According to the above discussion, all Domino gates in a circuit are precharged simultaneously. During the precharge phase the circuit wastes time since not useful computation takes place. Therefore, the operation of a Domino circuit is conventionally divided into two blocks with complementary clocks, so that when the first block evaluates the second block is in the precharge phase, while when the first block precharges the second is in the evaluation phase (see Fig. 2.7).

Let's consider a circuit consisting of cascaded Domino gates, where all gate input and output lines are set to low during the precharge. Then, during the evaluation phase the output of the first gate either remains "0" or makes a transition to "1", activating the next gate. This high value may continue its propagation along the gate chain, like the falling of the tiles in the well known structure of the domino game, so that's why this logic design style is called Domino logic.

A basic restriction of Domino logic is that we can only implement non-inverted logic since each dynamic gate is followed by a static inverter. However, there are

few solutions to this problem: a) the reorganization of the logic using simple Boolean transformations, like De Morgan's law, b) the use of Differential Domino Logic (DDL) as in Fig. 2.6 where the function *F* and its complement are realized, although this is design approach is characterized by increased cost, and c) the use of the NORA logic [36] which is an alternative dynamic design style where subsequent gates are realized using successionally pMOS and nMOS networks in the dynamic part without the insertion of the NOT gate at the output node. In that case the complement of the clock is also required.

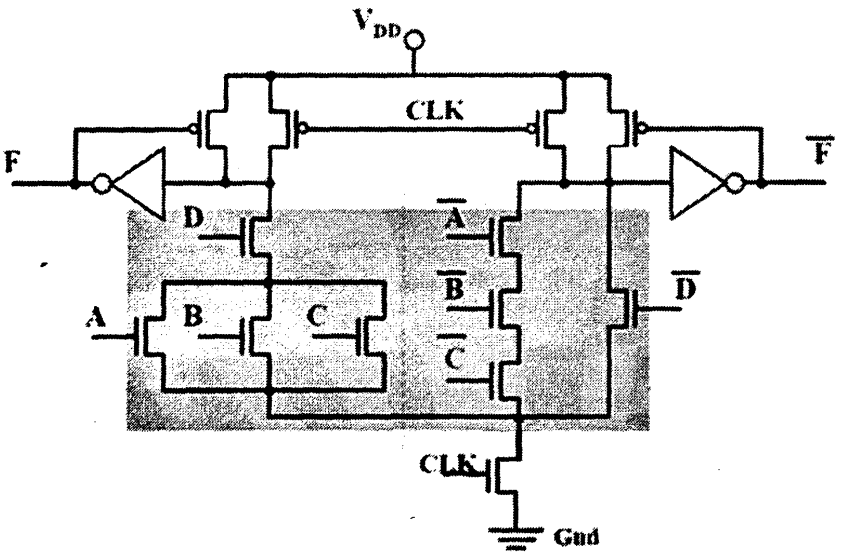


Figure 2.6 Differential Domino Logic (DDL)

2.3.4. Domino Circuits Operation

Designers are increasingly interested in faster circuit families, like Domino logic [12], [14]. Domino is a tempting choice because of the high speeds it achieves due to the decrease of the logical effort, which is a result of the standard pMOS network elimination. This is the reason it is used in critical sections of processing units like the arithmetic and logic units. The main disadvantage of the Domino logic is the high dynamic energy consumption due to the frequent alternations of the output values.

A real pipeline like that shown in Fig. 2.7 experiences clock skew. In the worst case, the dynamic gate and latch may have greatly skewed clocks. Therefore, the dynamic gate may not begin evaluation until the latest skewed clock, while the latch must set up before the earliest skewed clock. Hence, clock skew must be subtracted not just from each cycle, as it was in the case of a flip-flop based design, but from each half-cycle.

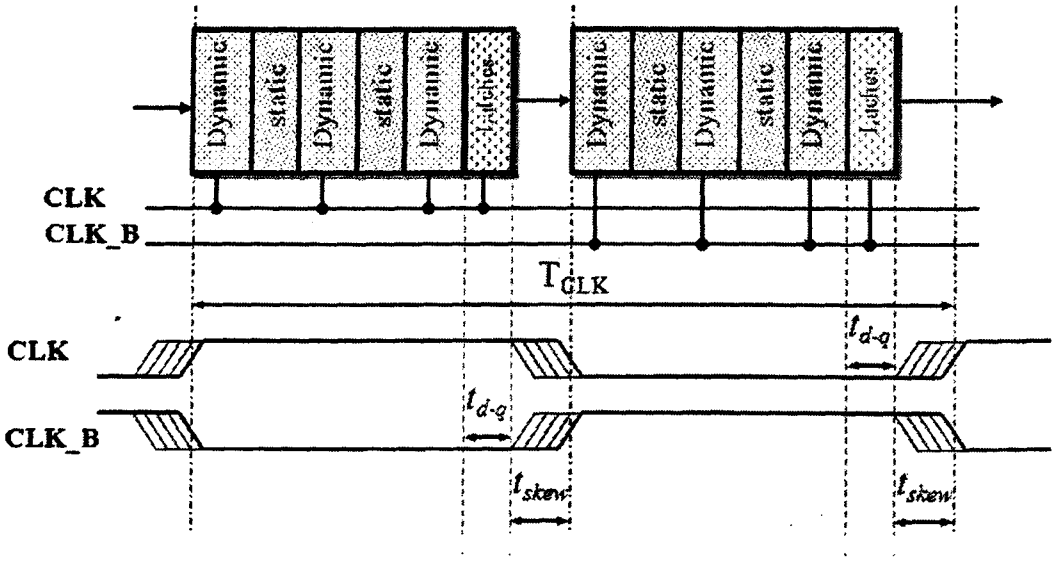


Figure 2.7 Classic domino including clock skew

Available Time for Logical Evaluation:

$$T_{logic} = T_{CLK} - 2t_{d\text{-}q} - 2t_{skew}$$

Traditional Domino pipelines also suffer from imbalanced logic. In summary, classic Domino circuits lose efficiency because they pay overhead for latch delay, clock skew and imbalanced logic [13].

2.3.4.1. Wave pipeline Domino logic

Wave pipelining is a technique to construct high-performance circuit designs which implements pipelining in logic without the use of intermediate latches or flip-flops. Wave pipelining can increase the clock frequency of practical circuits without increasing the number of internal storage elements. Using this technique, new set of input data can be applied to a combinational block before the previous responses are available at the output. In this way, pipelining of combinational logic blocks has been used effectively to maximize the utilization of the logic without inserting internal registers. This concept is applicable for single stage as well as for multi stage circuits [11], [13].

The basic problem with traditional Domino circuits is that data must arrive by the end of one half-cycle but will not depart until the beginning of the next half-cycle. Therefore, the circuits are infected by skew between the clocks and cannot borrow time. We can overcome this problem by using overlapping clocks, as shown in Figure 2.8. This figure presents a wave pipeline Domino clocking scheme with two overlapping clock phases. Instead of using one clock and its complement, we now use overlapping clocks CLK1 and CLK2 and we partition the logic into phases instead of half-cycles because in general we will allow more than two overlapping phases.

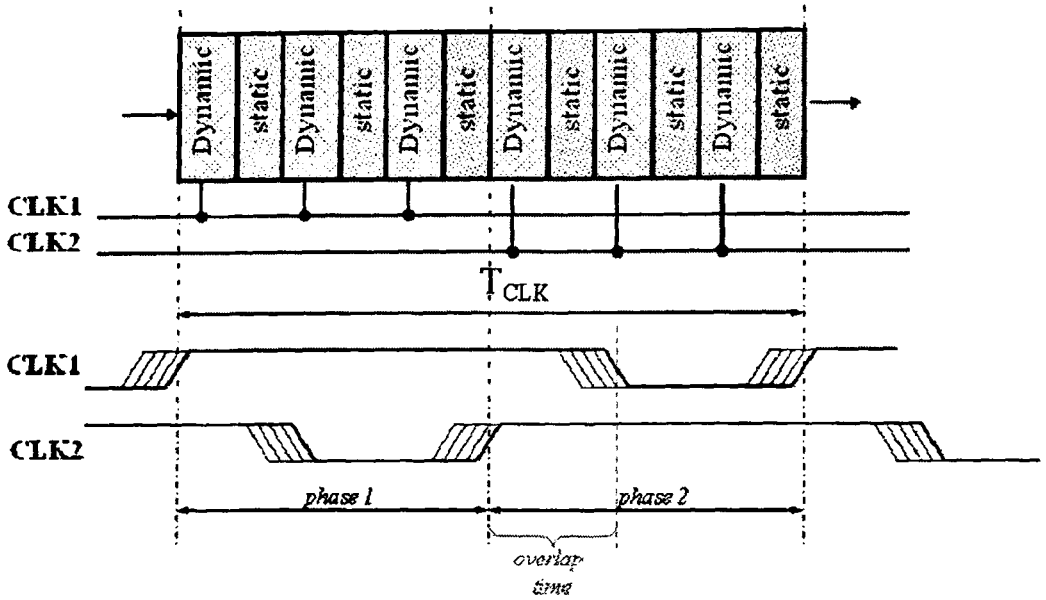


Figure 2.8 Wave pipeline Domino design

The clocks overlap enough so that even under worst-case clock skews, providing a minimum overlap, the first gate in the second phase has time to evaluate before the last gate in the first phase begins to precharge. As with static latches, the gates are guaranteed to be ready to operate when the data arrives even if skews cause modest variation in the arrival time of the clock. Therefore, the circuit is not affected by clock skew in the cycle time. Another advantage of wave pipeline Domino circuits is that latches are not necessary within the Domino pipeline. We ordinarily need latches to hold the result of the first phase's evaluation for use by the second phase when the first phase precharges. In wave pipeline domino, the overlapping clocks insure that the first gate in the second phase has enough time to evaluate before CLK1 falls and the first phase begins precharge. When the first phase precharges, the dynamic gates will pull high and therefore the static gates will fall low. This means that the input to the second phase falls low. The first gate of the second phase will remain at whatever value it evaluated to, based on the results of the first half-cycle, when its inputs fall low because both the nMOS

network and the precharge transistor will be off. Finally, wave pipeline domino circuits can allow time borrowing if the overlap between clock phases is larger than the clock skew. The guaranteed overlap is the nominal overlap minus uncertainty due to the clock skew. Gates in either Phase 1 or Phase 2 may evaluate during the overlap period, allowing time borrowing by letting gates that nominally evaluate during Phase 1 to run late into the second phase.

CHAPTER 3. ADDER CIRCUITS

3.1. Introduction to CMOS Adders' Design

An adder is a digital circuit that executes addition of numbers in many numerical representations, such as Binary-code, decimal e.t.c.. Manly, this circuit resides in the arithmetic logic unit where other operations are performed. The most common adders operate on binary numbers. The main requirement of digital computers is the ability to use logical functions to perform arithmetic operations. The basis of this is addition; if we can add two binary numbers, we can just as easily subtract them, or get a little further and perform multiplication and division.

3.2. Adder Types

Let's start by adding two binary bits. Since each bit has only two possible values, 0 or 1, there are only four possible combinations of inputs. These four possibilities, and the resulting sums, are shown in Table 3.1:

Table 3.1 Adding two binary bits

Inputs		Outputs	
A	B	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Circuits that perform this kind of addition on two one-bit binary numbers often written as A and B, are called Half Adders. As we can see in Fig. 2.1, a Half Adder can be built with the use of an XOR and an AND gate.

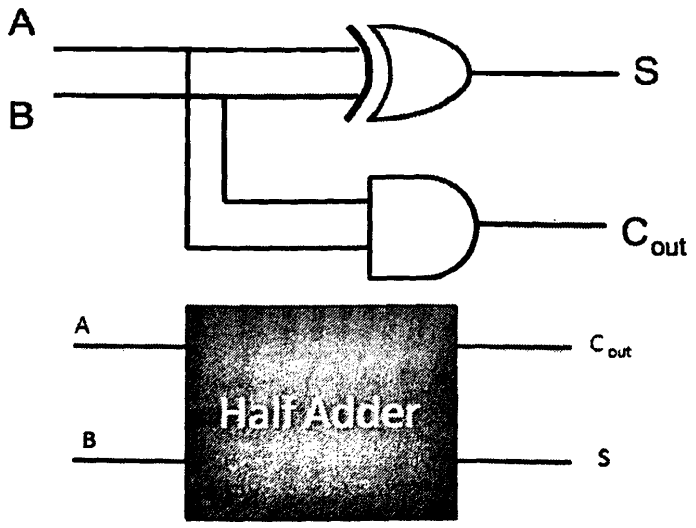


Figure 3.1 Half adder using an XOR and an AND gate.

The second adder type called Full Adder and is a logical circuit that performs an addition on three one-bit binary numbers. A full adder can be implemented in many different ways either at transistor level or gate level. An implementation, which is based on the next equations, is shown in Fig. 3.2:

$$S = (A \oplus B) \oplus C_{in}$$

$$C_{out} = (A \bullet B) + (C_{in} \bullet (A \oplus B))$$

where $\oplus$ is the XOR operation.

In this implementation, the final OR gate before the carry-out (C_{out}) output may be replaced by an XOR gate without altering the resulting operation.

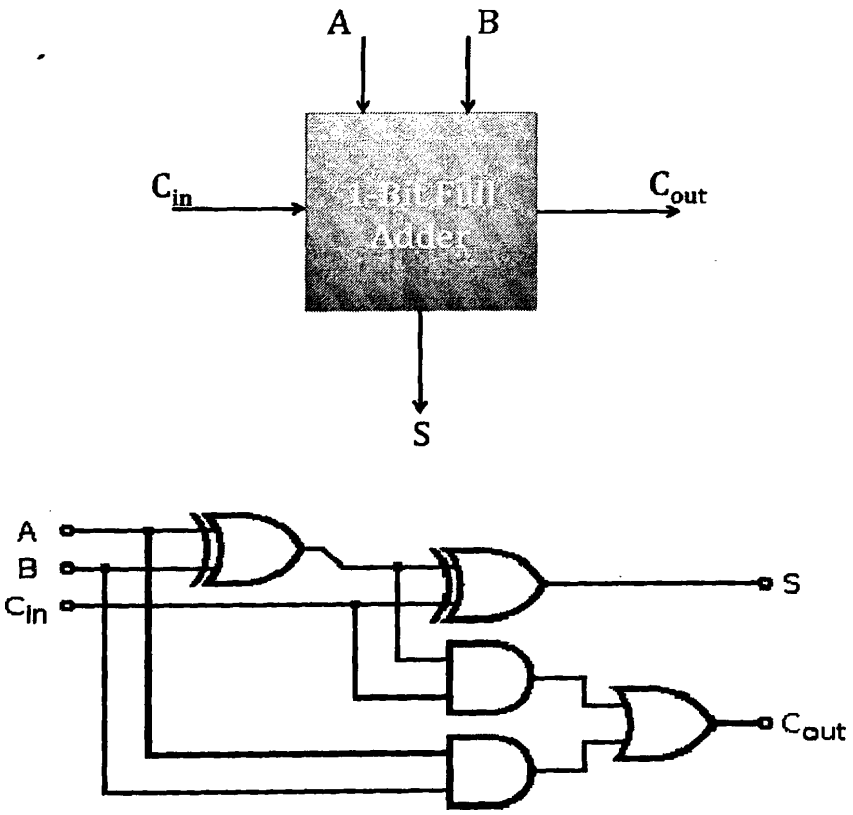


Figure 3.2 Schematic symbol for a 1-bit full adder and its gate level design

A full adder can be constructed from two half adders by connecting A and B to the input of one half adder, connecting the sum from that adder to an input of the second adder, connecting C_i to the other input with the sum output of the second half adder be the final sum and OR the two carry outputs to provide the final carry. Equivalently, S could be realized by a three-bit XOR of A, B, and C_i, and C_o could be realized by the three-bit majority function of A, B, and C_i. [1].

3.3. Multiple-bit adders

High speed adder architectures include the ripple carry adders, carry look-ahead (CLA) adders, carry-skip adders, carry-select adders, conditional sum adders, and combinations of these structures presented in [19-22]. High speed adders based on the CLA principle remain dominant, since the carry delay can be improved by calculating each stage in parallel.

3.3.1. Ripple Carry Adder

It is possible to create a logical circuit using multiple full adders to add N-bit numbers. Each full adder inputs a C_{in}, which is the C_{out} of the previous adder. This kind of adder is the *ripple carry adder*, since each carry bit "ripples" to the next full adder. We have to note that the first (and only the first) full adder may be replaced by a half adder.

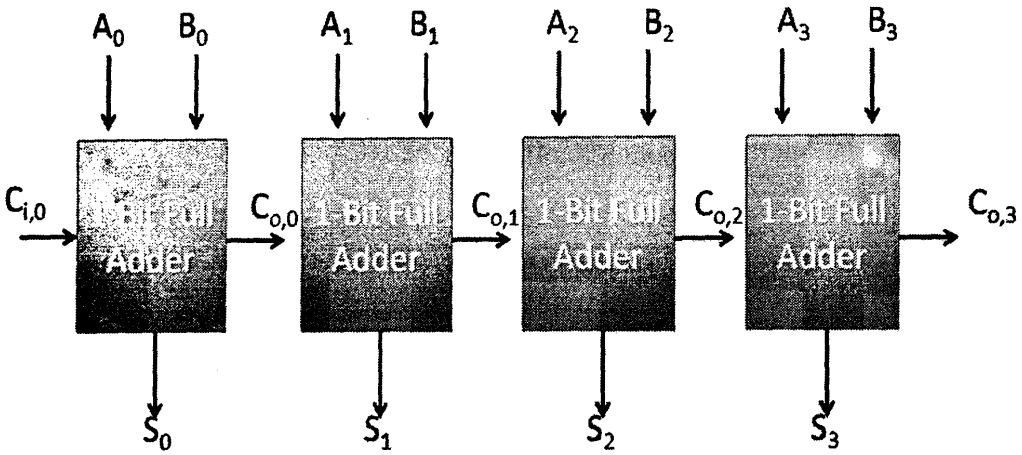


Figure 3.3 Four-bit ripple-carry adder topology

An N-bit adder can be constructed by cascading N full-adder circuits in series [17], by connecting $C_{0,k-1}$ to $C_{i,k}$ for k=1 to N-1, and setting the first carry-in $C_{i,0}$ to 0 (see Fig. 3.3). The delay through the circuit depends upon the number of the logic stages that must be traversed and is a function of the applied input signals. For some input signals, no rippling effect occurs at all, while for others, the carry has to ripple all the way from the least significant bit to the most significant bit. The propagation delay of each a structure is defined as the worst-case delay over all possible input patterns, also called the critical path.

The layout of ripple carry adder is simple, which allows for fast design time; however, the ripple carry adder is relatively slow, since each full adder must wait for the carry bit to be calculated from the previous full adder. The overall delay depends on the characteristics of the full-adders circuits; different CMOS implementation will produce different worst-case delay paths. The gate delay can easily be calculated by inspection of the full adder circuit. Each full adder requires three levels of logic.

In the case of ripple carry adder, the worst-case delay happens when a carry generated at the least significant bit position propagates all the way to the most significant bit position. This carry is finally consumed in the last stage to produce the sum. The delay is then proportional to the number of bits in the input words N and is approximated by:

$T_{adder} \approx (N - 1)t_{carry} + t_{sum}$

where t_{carry} and t_{sum} equal the propagation delays from C_i to C_0 and S, respectively [1].

3.3.2. Carry Look-Ahead Adders

A carry look-ahead adder improves speed by reducing the amount of time required to calculate carry bits [23]. The carry look-ahead adder calculates one or more carry bits before the sum, which reduces the wait time to calculate the result of the larger value bits. The Kogge-Stone adder and Brent-Kung adder are examples of this type of adder [17].

As mentioned above, carry look-ahead (CLA) adders are designed to overcome the latency introduced by the rippling effect of the carry bits. The CLA algorithm is based on the origin of the carry-out in the equation

$C_{i+1} = A_i \bullet B_i + C_i \bullet (A_i \oplus B_i)$

For the cases that gives $C_{i+1} = 1$, since either term may cause this output, we treat each one separately. First, if $A_i \bullet B_i = 1$, then $C_{i+1} = 1$. We define the generate term ($G_i = A_i \bullet B_i$), since the inputs are viewed as "generating" the carry-out bit. If $G_i = 1$, then we must have $A_i = B_i = 1$. The second term represents the case where inputs carry $C_i = 1$ may be "propagated" through the full-adder. This will happened if the propagate term ($P_i = A_i \oplus B_i$) is equal to '1': if $P_i = 1$ then $G_i = 0$ since the XOR operation produces a '1' iff the inputs are not equal. With these definitions, the equation for the carry-out bit is:

$C_{i+1} = G_i + P_i \bullet C_i$

Table 3.2 The basis of the carry look-ahead algorithm

	$G_i = A_i \bullet B_i$	$P_i = A_i \oplus B_i$
$A_i = B_i = 0$	0	0
$A_i = B_i = 1$	1	0
$A_i \neq B_i$	0	1

Table 3.2 shows the behaviour of the generate and propagate terms. The main idea of the CLA is to first calculate the values of P_i and G_i for every bit, then use them to find the carry bits C_{i+1}. Once these are found, the sum bits are given by $S_i = P_i \oplus C_i$ for every i. This avoids the need to ripple the carry bits serially down the chain.

Implementation Details

For each bit in a binary sequence to be added, the CLA logic will determine whether that bit pair will generate a carry or propagate a carry. This allows the circuit to "pre-process" the two numbers being added to determine the carry ahead of time. Then, when the actual addition is performed, there is no delay from waiting for the ripple carry effect. Below is a simple 4-bit generalized CLA circuit is discussed.

For the example provided, the logic for the generate (G) and propagate (P) values are given below. Note that the numeric value determines the signal, starting from 0 on the far left to 3 on the far right:

$C_1 = G_0 + P_0 \bullet C_0$

$C_2 = G_1 + P_1 \bullet C_1$

$C_3 = G_2 + P_2 \bullet C_2$

$C_4 = G_3 + P_3 \bullet C_3$

Substituting C_1 into C_2, then C_2 into C_3, then C_3 into C_4 yields the expanded equations:

$C_1 = G_0 + P_0 \bullet C_0$

$C_2 = G_1 + G_0 P_1 + C_0 P_0 P_1$

$C_3 = G_2 + G_1 P_2 + G_0 P_1 P_2 + C_0 P_0 P_1 P_2$

$C_4 = G_3 + G_2 P_3 + G_1 P_1 P_3 + G_0 P_1 P_2 P_3 + C_0 P_0 P_1 P_2 P_3$

These equations show that every carry bit can be found from the generate and propagate terms. Moreover, the algorithm yields nested expressions. The Carry CLA 4-bit adder can also be used in a higher-level circuit by having each CLA Logic circuit produce a propagate and generate signal to a higher-level CLA Logic circuit (see Fig. 3.4). The group propagate (*PG*) and group generate (*GG*) for a 4-bit CLA are:

$PG = P_0 P_1 P_2 P_3$

$GG = G_3 + G_2 P_3 + G_1 P_3 P_2 + G_0 P_3 P_2 P_1$

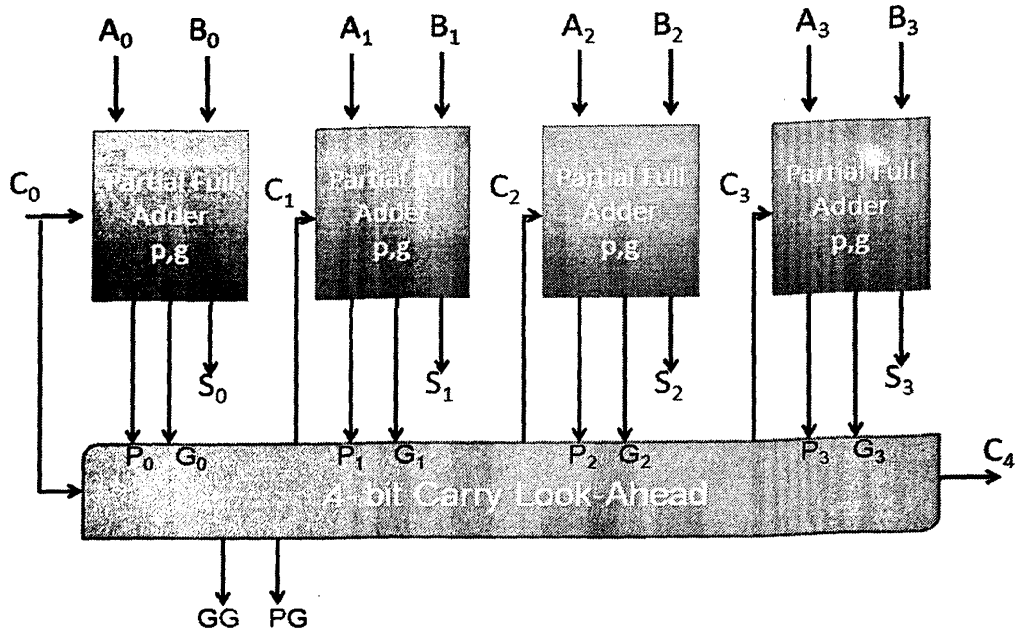


Figure 3.4 4-bit adder with Carry Look Ahead

3.4. Kogge-Stone Lookahead Adder

The Kogge-Stone adder is a parallel prefix form CLA adder. It has been developed by Peter M. Kogge and Harold S. Stone, which they published their work in 1973 [16]. It generates the carry signals in O(log n) time, and is widely considered the fastest adder design possible. It is the common design for high-performance adders in industry. Wiring congestion is often a problem for Kogge-Stone adders [17].

In order to build this adder, it is necessary to organize carry propagation and generation into recursive trees, by hierarchically decomposing the carry propagation into sub-groups of N bits:

$C_{o,0} = G_0 + P_0C_{i,0}$

$C_{o,1} = G_1 + P_1G_0 + P_1P_0C_{i,0} = (G_1 + P_1G_0) + (P_1P_0)C_{i,0} = G_{1:0} + P_{1:0}C_{i,0}$

$C_{o,2} = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_{i,0} = G_2 + P_2C_{o,1}$

$$C_{o,3} = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_{i,0}$$

$$= (G_3 + P_3G_2) + (P_3P_2)C_{o,1} = G_{3:2} + P_{3:2}C_{o,1}$$

$G_{i:j}$ and $P_{i:j}$ denote the generate and propagate functions, respectively, for a group of bits (from bit position i to j). $G_{i:j}$ equals "1" if the group generates carry, independent of the incoming carry. The block propagate $P_{i:j}$ is true if an incoming carry propagate through the complete group. For example, $G_{3:2}$, is equal to 1 when a carry either is equivalent to the bit position 3 or is generated at position 2 and propagated through position 3. In Fig. 3.5 an example of the structure of a 16-bit Kogge-Stone adder is provided, where carry at position 15 is computed by combining the results of blocks (0:7) and (8:15). Each of these, in turn, is composed hierarchically. For instance, (0:7) is the composition of (0:3) and (4:7), while (0:3) consists of (0:1) and (2:3), etc. The circuit of the 16-bit Kogge-Stone adder consists of three structural units. The first one is denoetd by the square symbol (□) and produces the generate and propagate signals, from the values of input signals according to the following equations:

$$P_i = A_i + B_i$$

$$G_i = A_iB_i$$

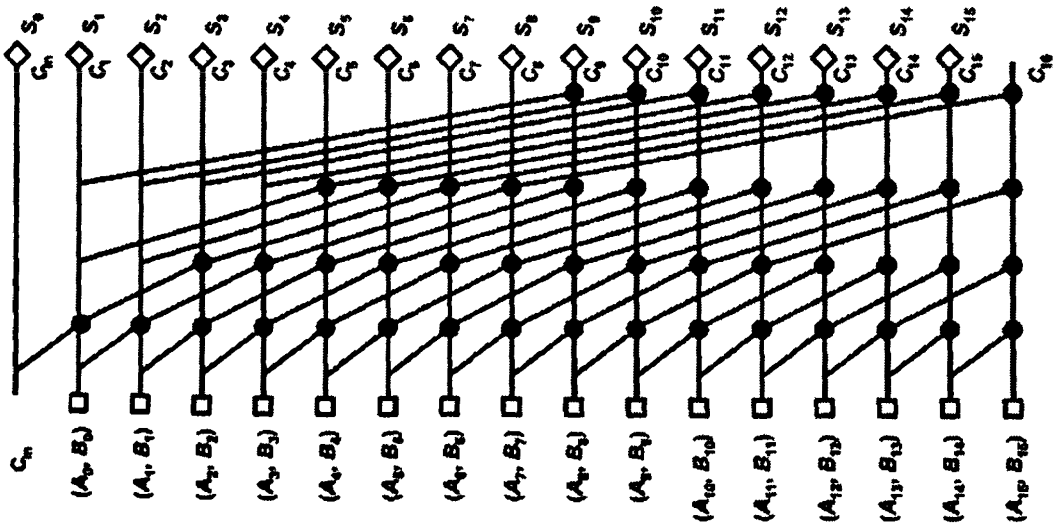


Figure 3.5 Structure of a 16-bit Kogge-Stone adder

The second structural unit, that is denoted by a black dot (•), is presented in Fig. 3.7 and represents two gates (AND, AND-OR), which calculate the block-level propagate and generate signals. This unit is used from second up to the fifth level. In the fifth level, the OR gate is not need. Since these gates are not located at the primary inputs side, the evaluation transistor is optional. During the precharge phase, all the outputs of the domino gate are guaranteed to be low, turning off any discharge path in the succeeding domino stage. Elimination of the foot switch in stages other than the first lowers effort of the gates and speed up the evaluation but increases power consumption. The transistor level implementations of the propagate and generate signals in Domino logic are given in Fig. 3.6

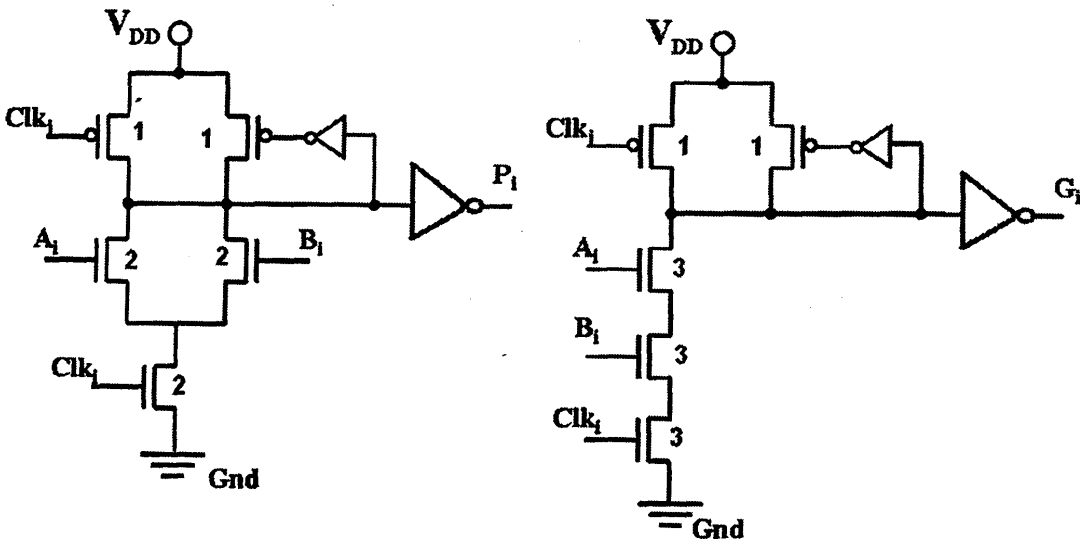


Figure 3.6 The Domino gate design of the first structural unit (AND, OR)

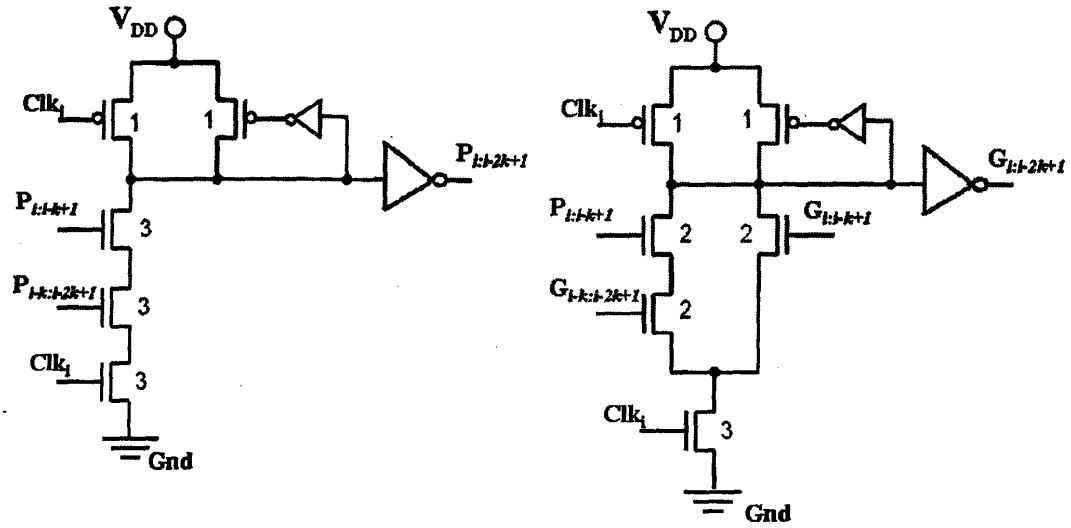


Figure 3.7 The Domino gate design of the second structural unit (AND, AND-OR)

Finally the third structural unit that is symbolized by a diamond (◊) is shown in the Fig. 3.8. This unit realizes the XOR gate that is required for the calculation of the final sum.. An additional XOR gate is used for each pair of inputs (a_i, b_i), which produces the propagate signal P_i for the calculation of the final sum.

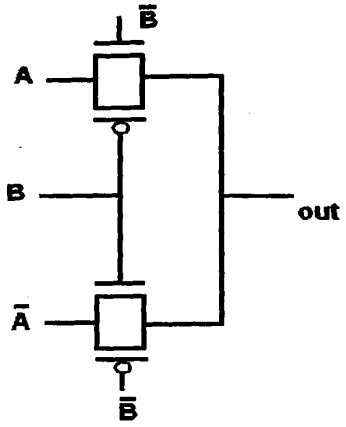


Figure 3.8 The XOR gate of third structural unit

3.5. Manchester Carry Chains (MCC)

The Manchester carry chain is a variation of the carry look-ahead adder that uses shared logic to lower the transistor count. As we know the logic for generating each carry contains all of the logic used to generate the previous carries [1], [18]. A Manchester carry chain generates the intermediate carries by tapping off nodes in the gate that calculates the most significant carry value. Not all logic families have these internal nodes, however, CMOS being a major example. Dynamic logic can support shared logic, as can transmission gate logic. One of the major drawbacks of the Manchester carry chain is that the capacitive load of all of these outputs, together with the resistance of the transistors causes the propagation delay to increase much more quickly than a regular carry look-ahead.Thus, a Manchester carry chain section generally won't exceed 4-bits [24].

The Manchester carry topology is based on building a switch-logic network for the basic equation:

$c_{i+1} = g_i + p_i c_i$

That can be cascaded to feed to successively stages. Consider a full adder with inputs a_i, b_i and c_i. We will use the generate and propagate expressions $g_i = a_i b_i$, $p_i = a_i \oplus b_i$ to introduce the term carry-kill that gets its name from the fact that if $k_i = 1$, then $p_i = g_i = 0$ so that $c_{i+1} = 0$; $k_i = 1$ thus "kills" the carry-out bit. This can be verified from the table below.

Table 3.3 Propagate, generate and carry-kill values

a_i	b_i	p_i	g_i	k_i
0	0	0	0	1
0	1	1	0	0
1	0	1	0	0
1	1	0	1	0

The Manchester carry topology is based on realized exploiting the behaviour described in Table 3.3. Since only one of the three quantities p_i, g_i and k_i can be high each time, we can construct the switch-level circuit using such a way so that one transistor is on (in conducting state) at a time as it is shown in Fig. 3.9.

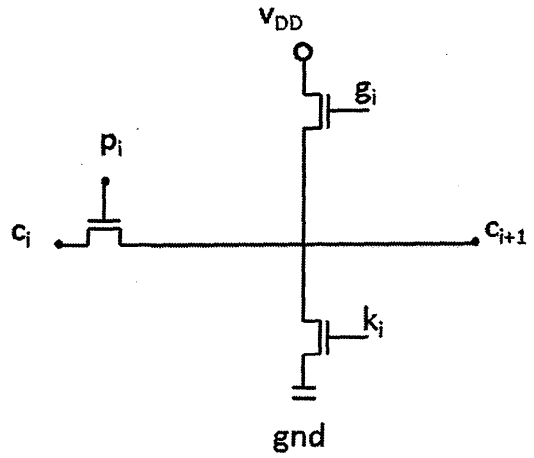


Figure 3.9 Switching network for the carry-out equation

Two of several different Manchester carry circuit implementations are shown in Fig. 3.10. The operation of the static logic gate is much complicated than the dynamic circuit.

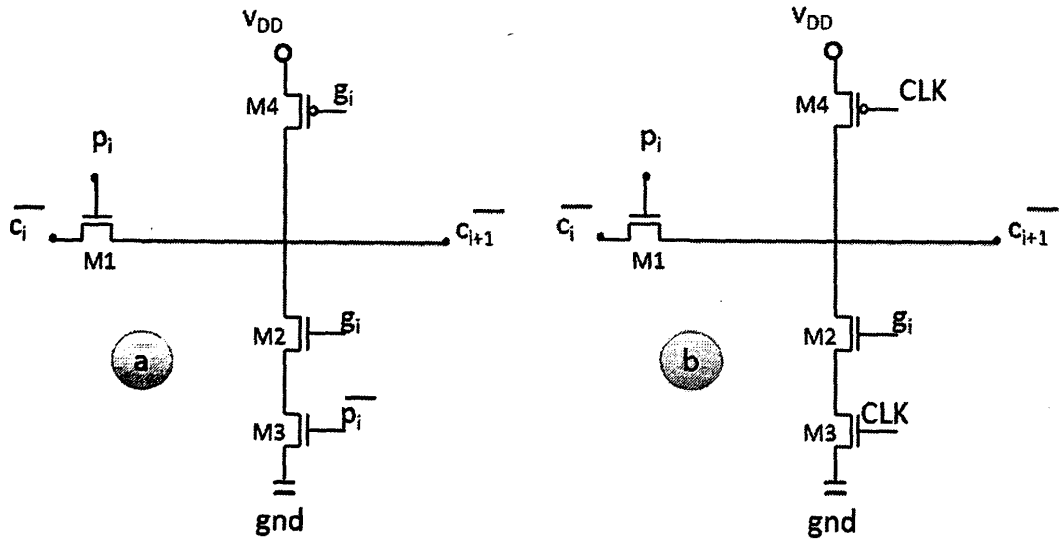


Figure 3.10 (a) Static circuit, (b) Dynamic circuit

The logic of the dynamic circuit is similar to the static except that the evaluation nMOS M3 in Fig. 3.11(b) replaces a logic transistor. During the precharge (CLK = 0), the output node is brought to the logic "1". Evaluation takes place when the clock switches to "0". A carry propagation occurs if p_i = 1, while the node discharges to "0" if g_i =1. This circuit can be use to build the Manchester carry chain shown in Fig. 3.11. Every stage undergoes precharge when CLK = "0". The carry bits are available during the evaluation time with the longest delay time for c_4.

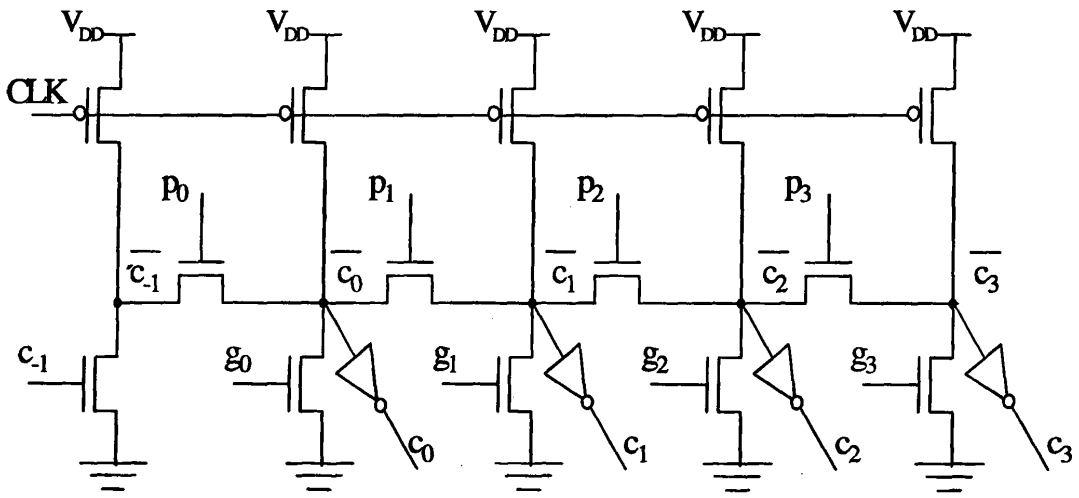


Figure 3.11 Conventional domino 4-bit MCC

In binary addition the computation of the carry signals is based on the following recursive formula:

$$c_i = g_i + z_i \cdot c_{i-1} \qquad (1)$$

$$g_i = a_i \cdot b_i$$

Generate signal

$$z_i = t_i = a_i + b_i$$

$$z_i = p_i = a_i \oplus b_i$$

Propagate signals

Where g_i and z_i are the carry generate and the carry propagate terms respectively.

In Fig. 3.12, the implementation of the generate and the two types of propagate signals (inclusive and exclusive) in Domino CMOS logic is shown.

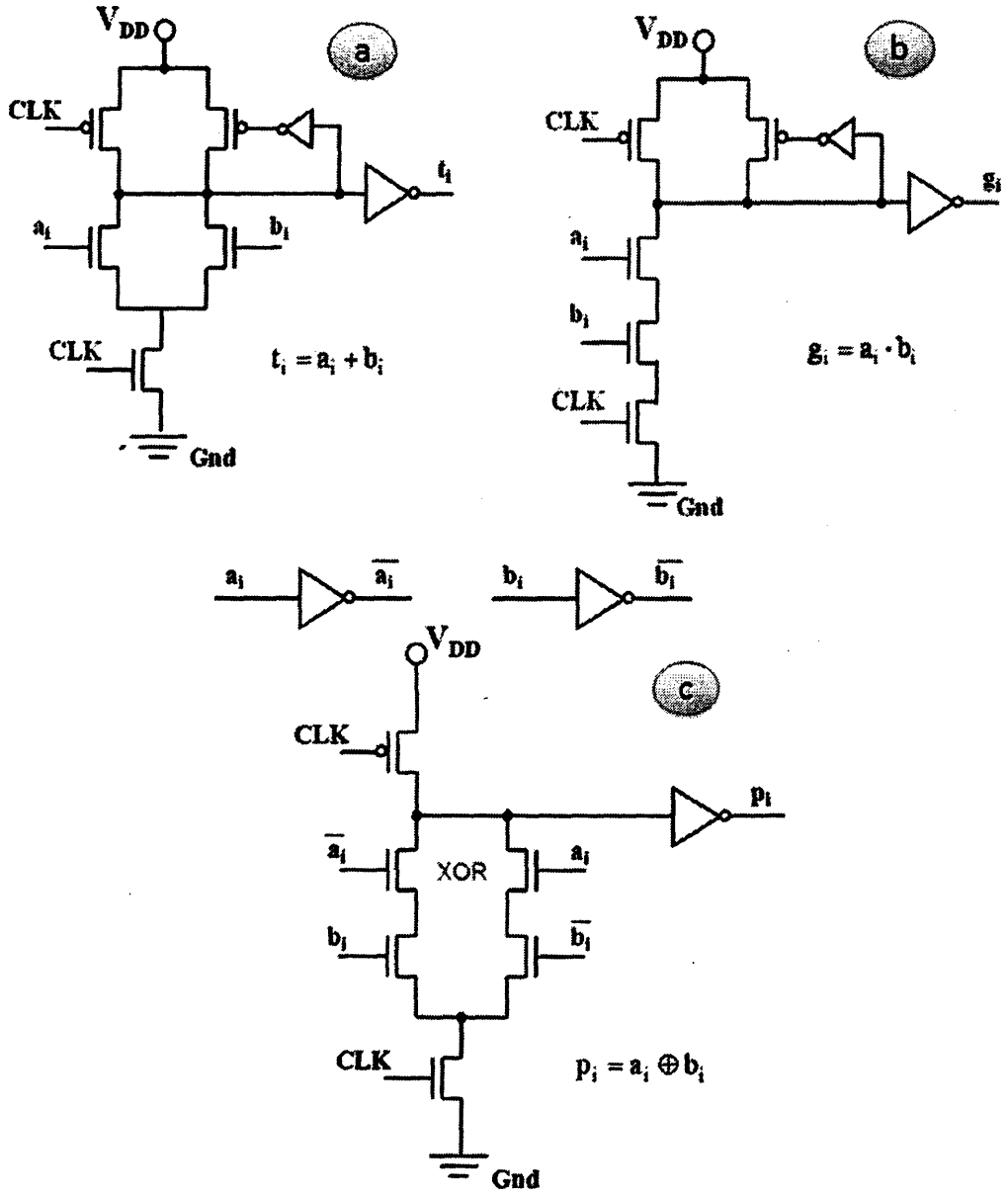


Figure 3.12 Domino implementation for the inclusive propagate (a), generate (b), and exclusive propagate (c) signals.

Expanding relations in (1) each carry bit c_i can be expressed as:

$$c_i = g_i + z_i g_{i-1} + z_i z_{i-1} g_{i-2} + \ldots + z_i z_{i-1} \ldots z_1 g_0 + z_i z_{i-1} \ldots z_0 c_{-1}$$

The sum bits of the adder are defined as $s_i = p_i \oplus c_{i-1}$, where c_{-1} is the input carry.

For the implementation of the sum signals the Domino chain is terminated and the sum bits of the Manchester Carry Chain adder are implemented using static CMOS XOR gates [17], the design of which is shown in Fig. 3.13.

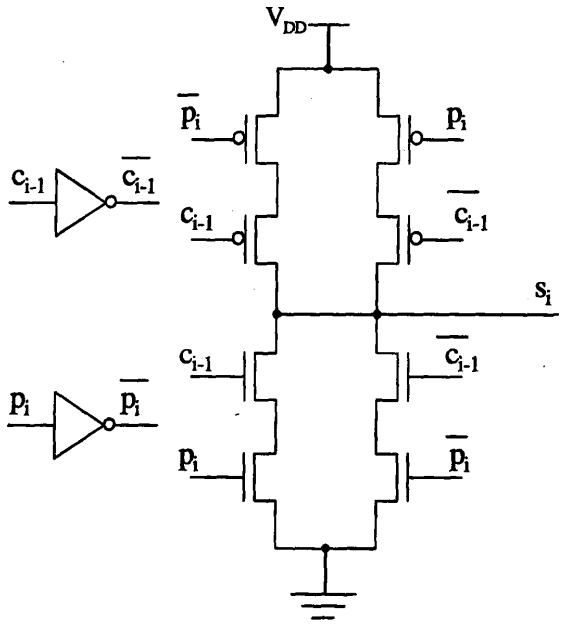


Figure 3.13 Static CMOS implementation of the XOR gate for the sum calculation.

3.5.1. Carry Bypass in MCC Adder

Several variations of the MCC adder in Domino CMOS logic have been proposed in the literature [17], [24-29]. Moreover, static CMOS MCC implementations are also available [30]-[31]. All these works, aimed to speed up the addition operation

using different techniques. In the following paragraph we will introduce the Carry Bypass technique.

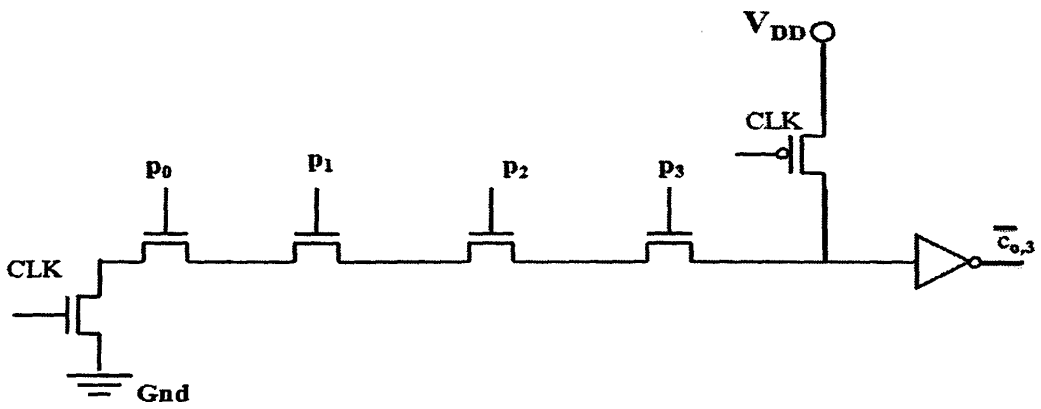


Figure 3.14 MCC implementation of the bypass adder

Fig. 3.14 demonstrates the bypass MCC adder design that can speed up the addition, where the carry propagates either through the bypass path or generated somewhere in the chain. In both cases, the delay is smaller than the normal ripple configuration.

A high speed design has been proposed in [29], where the MCC is supported by the carry-skip capability to improve performance. Each *m*-bit block has two carry skip pull-down transistors controlled by a skip signal. This skip signal (sk_j) is generated by ANDing all *m* carry propagate signals, where:

$$sk_j = P_{mj}\, P_{mj+1}\, P_{mj+2} \ldots P_{mj+m-1}$$

The carry skip pull-down transistor speed up the generation of the m^{th} carry bit of the block and restore signal strength at this node, eliminating the need for intermediate buffers between the blocks nodes.

In Fig. 3.15, an 8-bit adder is designed using this technique. We have to notice that the implementation of n-bit adder we need n/m blocks.

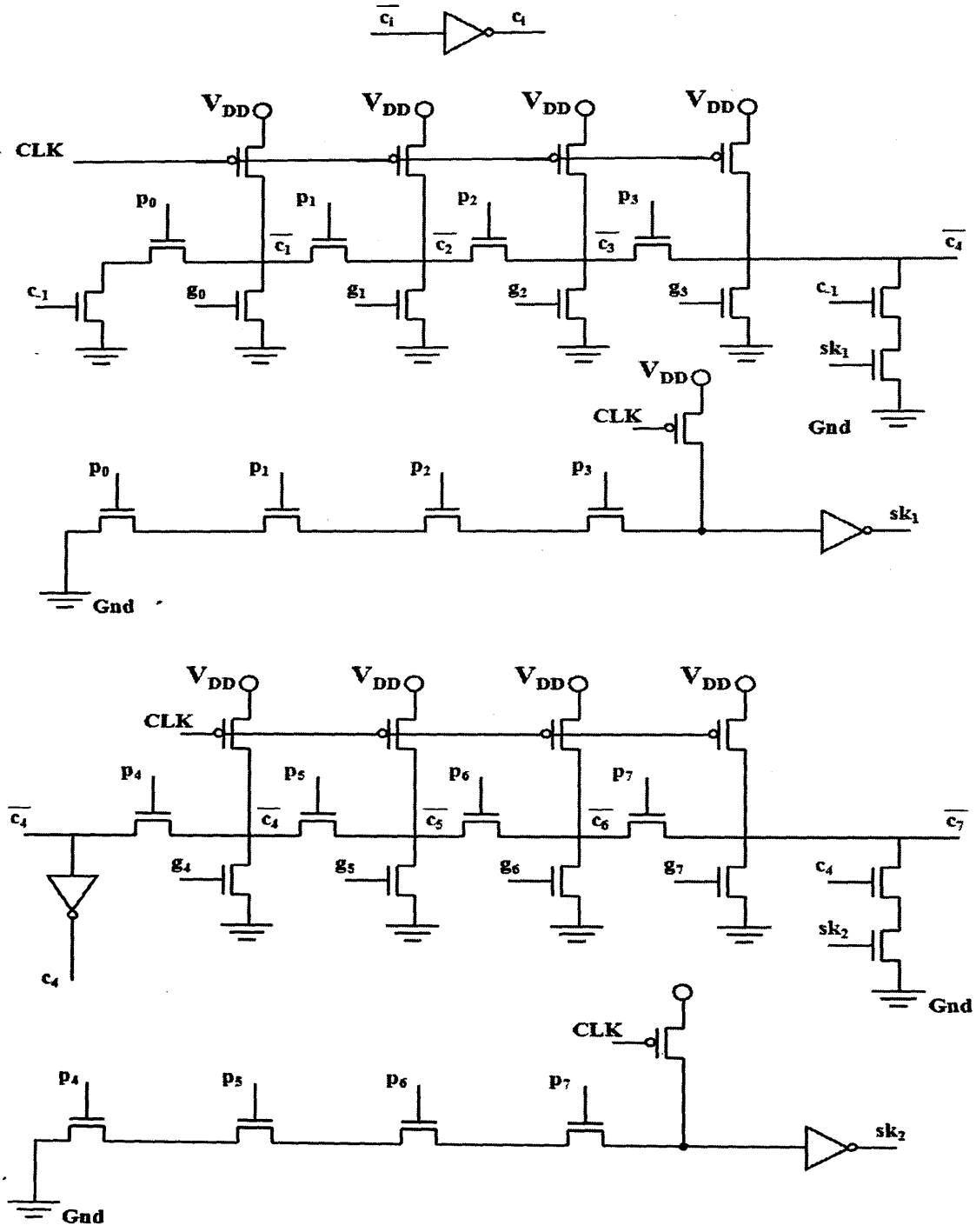


Figure 3.15 First and Second 4bit Manchester chains using SK signals

CHAPTER 4. MEMORY-LESS PIPELINE DYNAMIC CIRCUIT DESIGN TECHNIQUE

4.1. Introduction

In this chapter we present a new dynamic circuit design techniques that allow the implementation of pipeline structures without the need of memory elements; instead they exploit a three phase clocking design style. The pipeline operation along with the memory elements elimination provides very high speed circuit realizations. Their efficiency is demonstrated on Kogge-Stone adder designs.

4.2. The Pipeline Dynamic Technique

The proposed logic family is a three phase dynamic logic design style that overcomes the inherent race condition problems of the conventional dynamic logic. As in the case of conventional dynamic logic, a gate consists of two control

transistors, one nMOS that is active (in conducting state) during the evaluation phase (evaluation transistor) and one pMOS that is active in the precharge phase (precharge transistor), as it is shown in Fig. 4.1(a), and an nMOS network that its structure depends on the gate functionality (see the NAND gate of Fig. 4.1(b)).

The nMOS network is the same as the corresponding nMOS network of the full CMOS gate design for the same function. The gate version with a keeper included is illustrated in Fig. 4.1(c). Two clock signals (CLK1 and CLK2) of equal period are used to drive each one of the two control transistors and provide the three operating phases, of equal time duration (called phase time), as it is shown in Fig. 4.2. The three operating phases are the *precharge*, *evaluation* and *memory* phases.

The precharge phase of the proposed design style is exactly the same as the precharge phase in a dynamic design. The pMOS precharge transistor of the gate is activated and the output is precharged to high. The nMOS evaluation transistor is inactive and ensures that there is not any discharging path though the nMOS network. The precharge operation does not depend on the input values of the nMOS network and can be completed regardless of these values.

The evaluation phase is analogous to the evaluation phase in a dynamic design. The pMOS precharge transistor is inactive and the nMOS evaluation transistor is active. Depending on the inputs combination and the realized function by the gate, either a conducting path is formed through nMOS network (active path) and the output (*Out*) is discharged to low, or there is not any active path through nMOS network and the output remains charged to high. Thus, the input values of the nMOS network during this phase, actually determine the response value of the gate at the end of this phase. According to the proposed design style, a high value at the output of the gate at the beginning of the evaluation phase is required, while valid and stable values are assumed at the inputs of the nMOS network during the whole phase time. Note that input transitions or glitches during the evaluation phase may

discharge the output resulting to an erroneous response since it is not possible to charge the output in any other phase except the precharge phase.

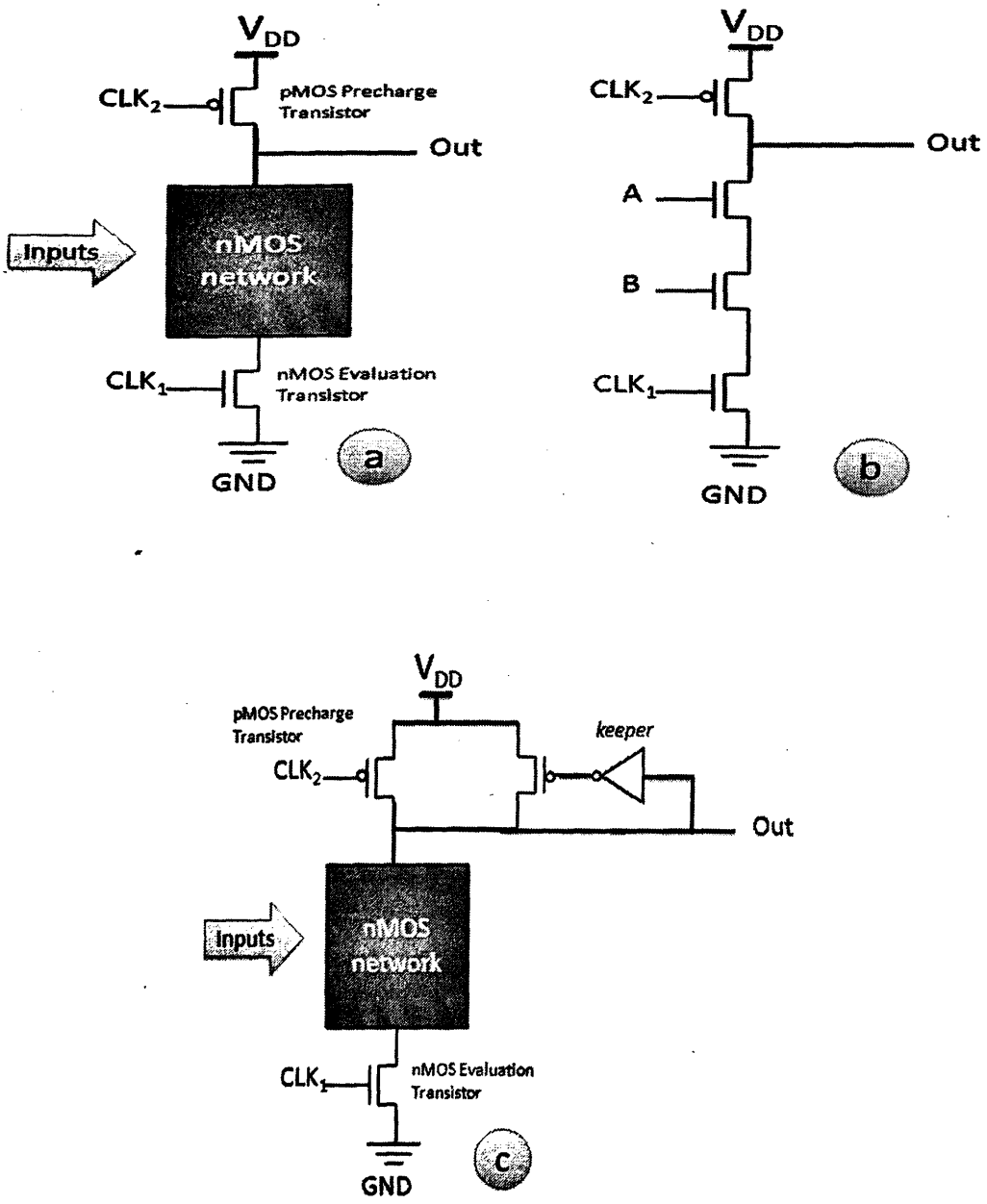


Figure 4.1 a) The proposed dynamic gate, b) The NAND gate c) The proposed dynamic gate with keeper

Finally in the memory phase both pMOS and nMOS control transistors are inactive and the output will retain the state (logic low or high). This phase does not normally exist in typical dynamic gates. During the memory phase the input values of the nMOS network should not affect the output of the gate.

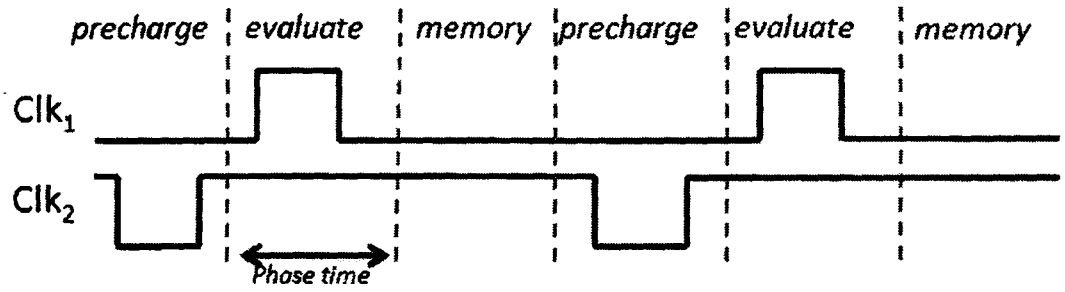


Figure 4.2 Clock signals' waveforms

For the proper operation of the proposed scheme as a pipeline, each gate level (pipeline stage) is passing continuously through the three phases precharge, evaluate and memory in that order, as shown in Fig. 4.2. The existence of the precharge phase before the evaluation phase ensures that the requirement for a high value at the output of a dynamic gate at the beginning of the evaluation phase is fulfilled. The memory phase is after the evaluation phase and ensures that for a phase time the values calculated during the evaluation phase will remain stable.

The relation of the operating phases in a specific level of the pipeline with the operating phases of the preceding and the following levels is shown in the example of Table 4.1. This refers to a four levels design and the evaluation of three sets of input data. We denote as e_j the evaluation of the data set (j) at a level while m_j denotes the memory phase holding these data; finally p denotes a precharge phase. A phase time duration corresponds to a cycle in the pipeline operation. With this arrangement we guarantee that during an evaluation phase in a level the preceding

level is at the memory phase. This ensures that during the evaluation phase, the inputs of the gate are stable since they are outputs of a level in the memory phase. In addition during the precharge or evaluation phases of a level (where its outputs may change) the following level in the pipeline is at the memory or precharge phases respectively where there is not any constraint on the inputs' status (to be stable or not).

Table 4.1 The pipeline operation of the proposed dynamic logic.

	Gate Level			
	L1	L2	L3	L4
1	p	x	x	x
2	e_1	p	x	x
3	m_1	e_1	p	x
4	p	m_1	e_1	p
5	e_2	p	m_1	e_1
6	m_2	e_2	p	m_1
7	p	m_2	e_2	p
8	e_3	p	m_2	e_2
9	m_3	e_3	p	m_2
10	x	m_3	e_3	p
11	x	x	m_3	e_3
12	x	x	x	m_3

The above arrangement also ensures the proper operation of the pipeline. The evaluated response at level L_i of the pipeline during the n cycle are retained at the

outputs of this level during the (n+1) cycle time (level L_i is at the memory phase during (n+1) cycle) and are used as inputs at level L_{i+1} during the (n+1) cycle.

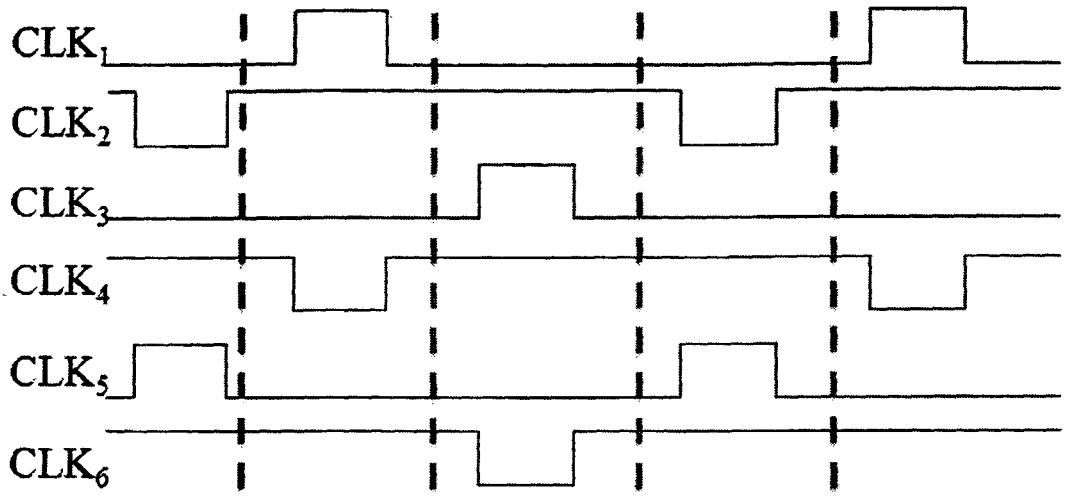


Figure 4.3 The three phases clocking scheme

In order to achieve the above pipeline operation, appropriate clock signals are required at each level. Three clock signals (CLK1, CLK2 and CLK3) and their complements (CLK2, CLK4 and CLK6) are used as shown in Fig. 4.3. The clock distribution arrangement is presented at Table 4.2 and ensures that the two clock signals used at level L_{i+1} are the clock signals used at level L_i delayed by one third of the clock period (or equivalently a phase time). The pipeline construction along with the selection of the appropriate clock signals for each gate, according to its level in the design, is demonstrated in Fig. 4.4.

Table 4.2 Clock signal selection according to the level of the gate.

Level	nMOS	pMOS
L mod 3 = 1	Clk1	Clk2
L mod 3 = 2	Clk3	Clk4
L mod 3 = 0	Clk5	Clk6

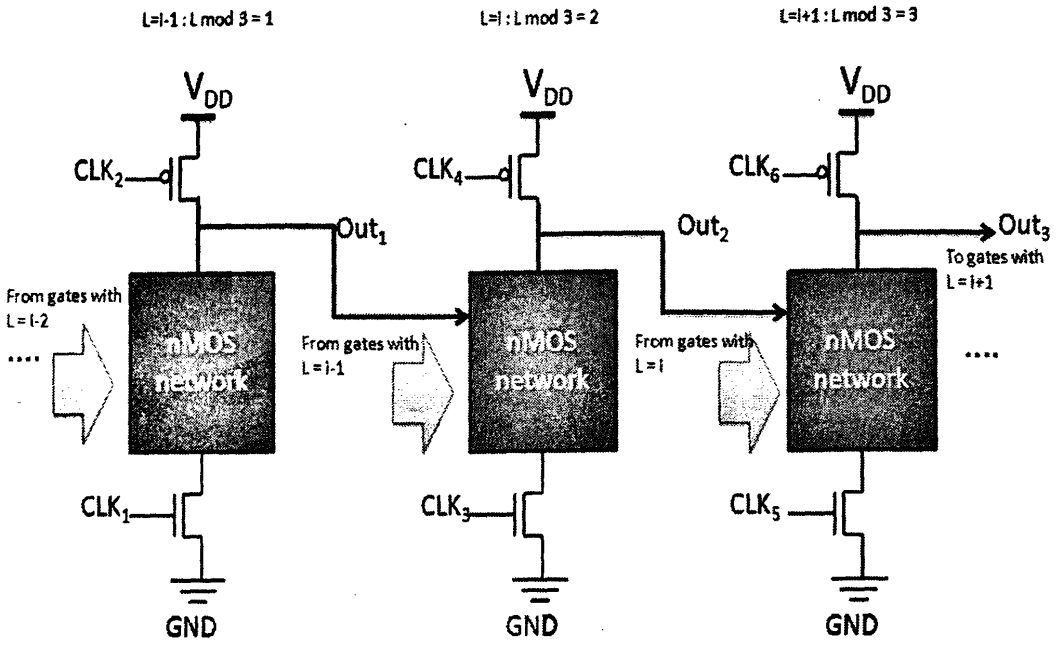


Figure 4.4 Pipeline structure.

An inherent problem in dynamic logic is charge sharing that may lead to erroneous output values during the evaluation. Common solutions to overcome this problem are the use of extra precharge transistors (to precharge the internal nodes of the nMOS network during the precharge phase) or the use of keepers as shown in Fig. 4.1(c). The proposed design technique provides an advantage with regard to the charge sharing problem. When a gate G (lets say at level L=i) is at the precharge phase its predecessor gates (at level L=i-1) are in the evaluation phase and their outputs are settling to the final value that will serve as input for the subsequent evaluation of gate G. Thus, in the nMOS network of gate G the final conducting paths are activated during its precharge phase so that the pertinent internal nodes can be precharged to $V_{DD}-V_{tn}$ (where V_{tn} is the nMOS threshold voltage). Given that the precharge time is enough the paths will be fully formed and the internal nodes will be precherged. Consequently, the charge sharing problem is alleviated

and no internal node precharging is required for these nodes as in the design of complex (high internal parasitic capacitance) Domino gates.

The proposed design technique enables the implementation of both inverting and non-inverting gates compared to the limitation of non-inverting gate implementations in the standard Domino logic. However, its main advantage is the ability to implement pipelines without the need of memory elements. For high performance applications the use of pipelines is highly desirable but the additional memory elements require more hardware and introduce extra delays in each pipeline stage.

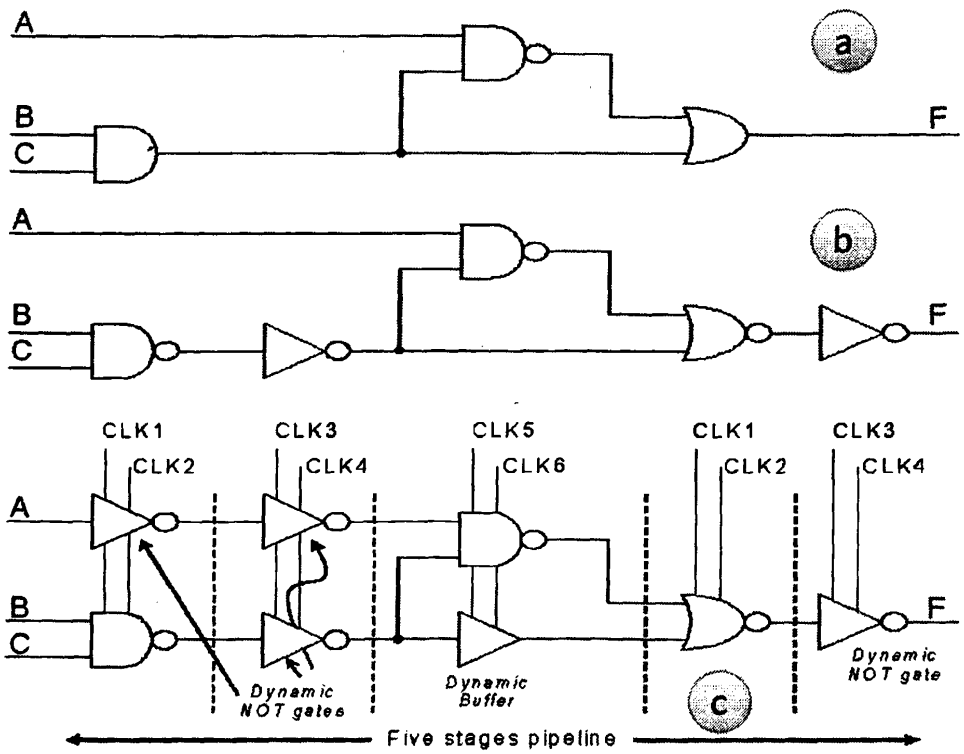


Figure 4.5 Circuit design example: a) function to be realized, b) CMOS design and c) memory-less pipeline dynamic design

A characteristic of any pipeline design is that the inputs at level L=i+1 are the outputs of level L=I, while the outputs of any level before L=i cannot be used without the addition of cascaded memory elements in-between that their number is equal to the number of intermediate levels. This increases the hardware cost of a circuit design. However, using the proposed design technique the above restriction can be easily fulfilled at a very low cost. In case that we need to connect the output of a gate at level L=i as input to a gate at level L=i+k we have to add *k* levels in-between. In case that *k* is even, the solution is to use a dynamic NOT gate for each one of the *k* intermediate levels. Since the number of the added NOT gates is even we do not alter the functionality of the circuit. In case that k is odd, the solution is to use a dynamic NOT gate for each one of the *k-1* intermediate levels plus a dynamic buffer (dynamic NOT gate followed by a static NOT gate). Once again, since the number of the added NOT gates is even we do not alter the functionality of the circuit. An example of the proposed design approach is shown at Fig. 4.5.

4.3. Enhanced Performance Pipeline Dynamic Design

In order to improve further the performance of the proposed pipeline dynamic design style a modification in the topology of the dynamic gates is introduced. The tail nMOS evaluation transistor that lies between the nMOS network and the ground is moved up between the output and the nMOS network, as it is shown in Fig 4.6. The new topology provides the ability to exploit the precharge phase of a gate as a pre-evaluation phase, where part of the evaluation operation is hidden inside the precharge phase as it is analyzed next.

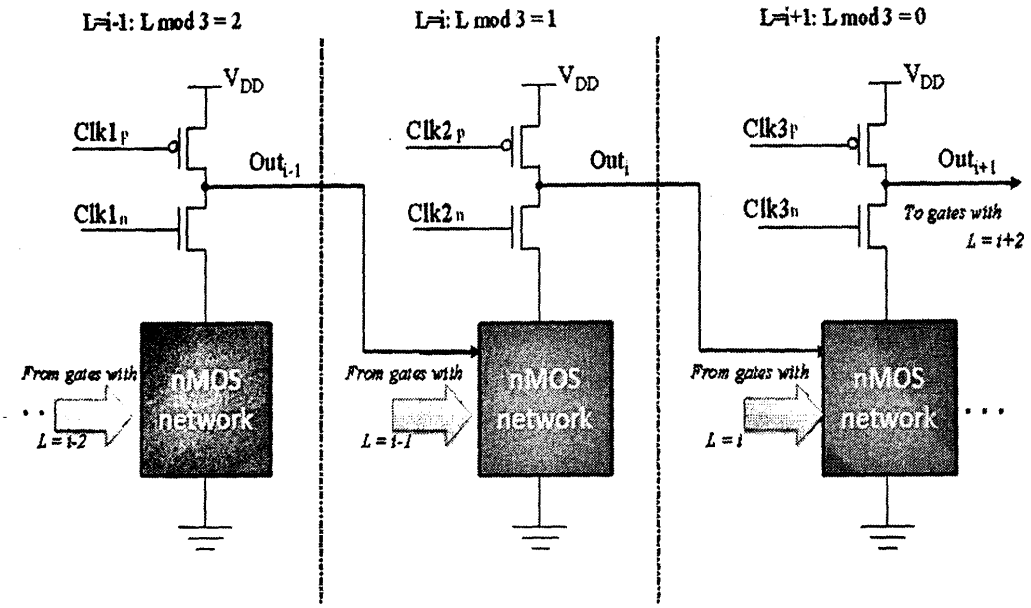


Figure 4.6 Enhanced performance pipeline dynamic design style

When a gate G (lets say at level L=i) is at the precharge phase its predecessor gates (at level L=i-1) are in the evaluation phase and their outputs are settling to the final value that will serve as input for the subsequent evaluation of gate G. Thus, in the nMOS network of gate G the final conducting paths, if any, are activated during its precharge phase so that the pertinent internal nodes can be discharged (pre-evaluation). Consequently, during the evaluation phase of gate G only its output node (Out) remains to be discharged. Obviously this operation will be completed faster than in the initial topology where all internal nodes as well as the output node of gate G are discharged during the evaluation phase with the output node last.

Although a keeper circuit can be also used in the new topology, a complex gate with a deep nMOS network, of high parasitic capacitance, may suffer by charge sharing phenomena between the output node and the nMOS network that may lead in reliability loss. Since the internal nodes of the nMOS network is not possible to

be precharged during the precharge phase of the actual gate due to the pre-evaluation of the this network, deep nMOS networks may not be feasible to be realized according to the enhanced dynamic design approach. To overcome this problem, complex gates can be split in to two or more simple gates composed of shallow nMOS networks. Alternatively, complex gates can be designed according to the initial dynamic design approach presented in section 4.2.

4.4. Kogge-Stone Adder Design and Simulation Results

16-bit Kogge-Stone adders [1], [16] have been designed in a standard 180nm CMOS technology (V_{DD}=1.8V) using the proposed dynamic design techniques for the implementation of their carry look-ahead (CLA) units. Their architecture is shown in Fig. 4.7. In addition, the corresponding CLA unit has been also designed using the standard Domino design style.

Each line inside the CLA unit of Fig. 4.7 (except the primary inputs) carries a pair of generate/propagate signals (G_j, P_j). The square symbol at the first level in Fig. 4.7 represents the calculation of the generation/propagation signals by the primary inputs. Moreover, each circle in Fig. 4.7 represents a "dot" operation between two pairs of generate/propagate signals $(G_j, P_j)\cdot(G_s, P_s)$.

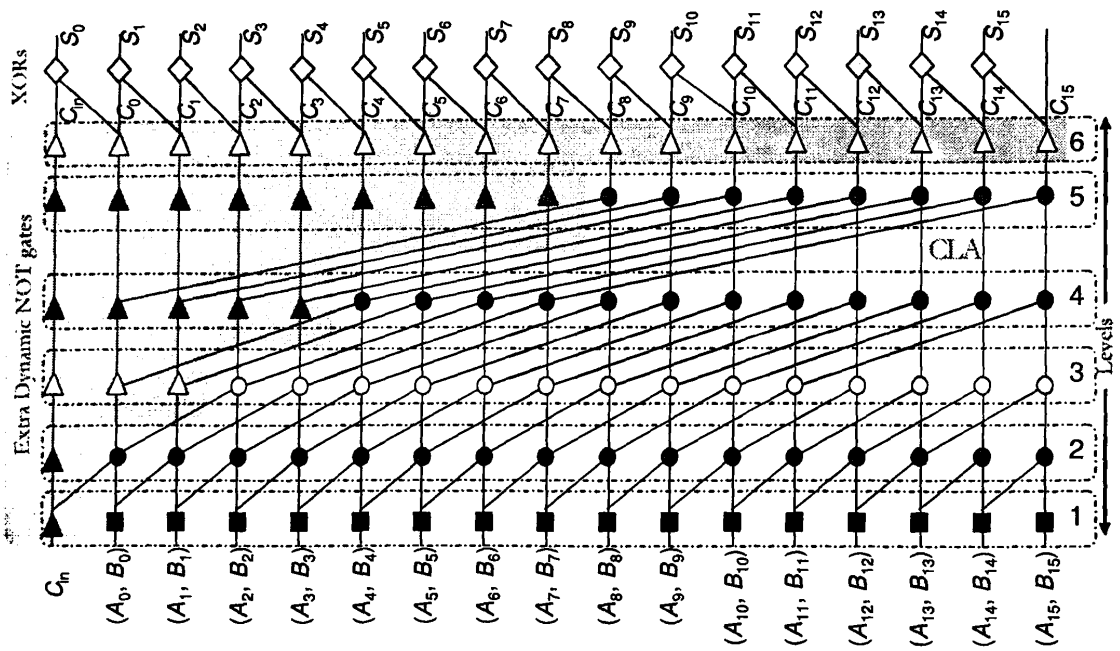


Figure 4.7 The 16-bit Kogge-Stone adder architecture

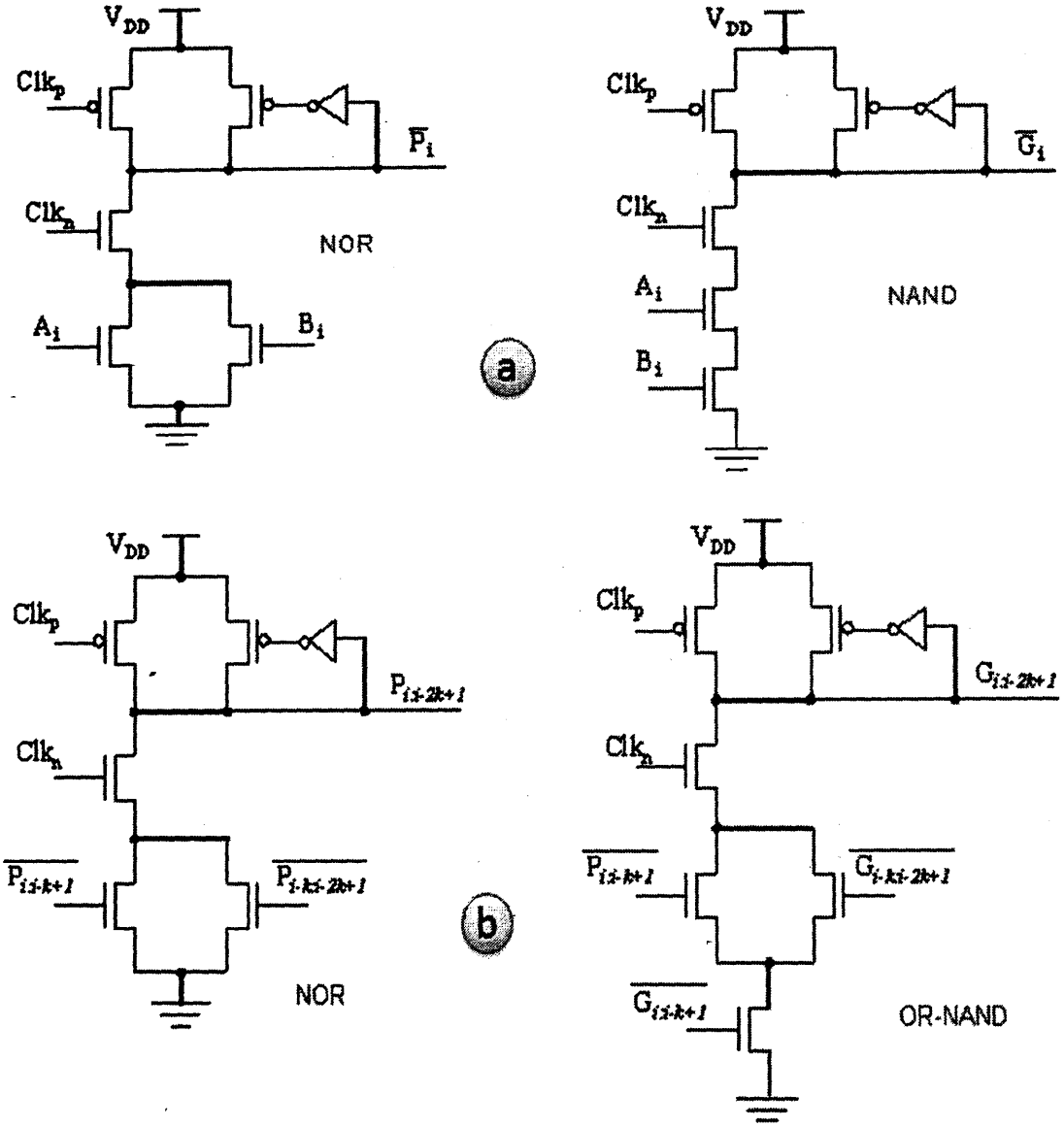
The above operations are defined as follows [1] for each level in the design:

$\overline{G_{1:0}} = A \cdot B$ and $\overline{P_{1:0}} = A + B$ for the 1st level

$G_{j:s} = \overline{\overline{G_j} \cdot (\overline{P_j} + \overline{G_s})}$ and $P_{j:s} = \overline{\overline{P_j} + \overline{P_s}}$ for the 2nd and 4th levels

$\overline{G_{j:s}} = G_j + (P_j \cdot G_s)$ and $\overline{P_{j:s}} = P_j \cdot P_s$ for the 3rd and 5th levels

In the proposed design of Fig. 4.7, each gate level is fed by the same pair of clock signals, while levels that are fed by the same clock signals are denoted with the same greyscale color. For each gate level the corresponding dynamic gates are presented in Fig. 4.8, where the enhanced design approach presented in section 4.3 has been used. It is easy to derive the initially proposed dynamic design of these gates by removing the clocked nMOS evaluation transistor and adding it as the tail transistor of the nMOS network.



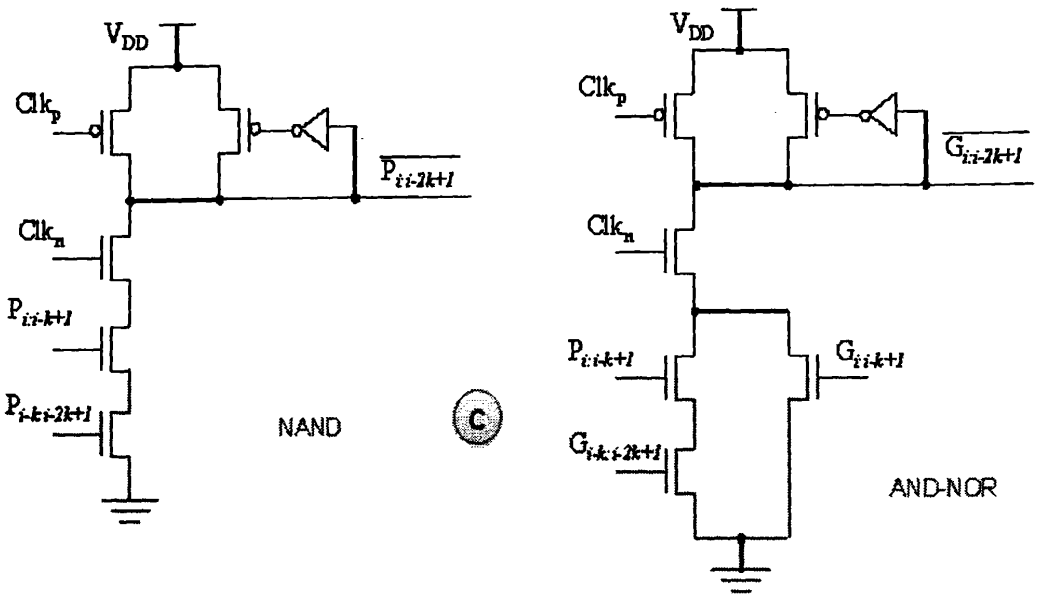


Figure 4.8 Enhanced dynamic gates

In Figure 4.8 the enhanced dynamic gates for the CLA unit design of the Kogge-Stone adder is used as follow: a) first level gates (□ symbol) for generate/propagate signals calculation, b) second and fourth levels (○ symbol) dot operation gates and c) third and fifth levels (○ symbol) dot operation gates.

According to the simulation results, the worst case propagation delay in the evaluation phase for the first one of the proposed dynamic designs is 63.36ps, while the corresponding delay of the second enhanced dynamic design is 44.80ps, which results in a delay reduction of 29.3% for second design approach. Note that considering the corresponding Domino design of the CLA unit for this Kogge-Stone adder, the worst case propagation delay in the evaluation phase is 291.69ps, which results in a delay reduction of 78.28% and 84.64% for the initial proposed and the enhanced proposed dynamic designs respectively. In addition, the worst case propagation delay in the evaluation phase of the CLA unit for the same Kogge-Stone adder using a five stages, four phases, Wave Domino design style is

164.345ps (see chapter 2.3.4.1), which results in a delay reduction of 61.45% and 72.74% with respect to the proposed designs.

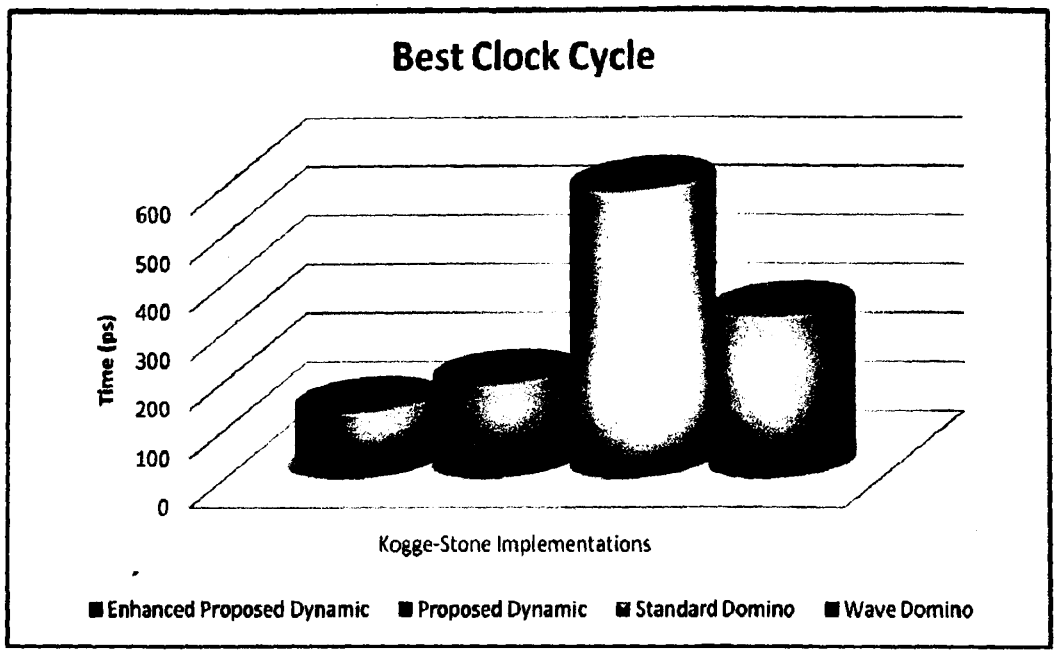


Figure 4.9 Best clock cycle of Kooge-Stone implementations

Since the above worst case evaluation times are at least equal or higher than the pertinent worst case precharge times, it is implied that clock signals with periods of at least 190.08ps (3×63.36ps) and 134.40ps (3×44.8ps) for the initial and the enhanced proposed designs are required, that results in 67.42% and 76.42% reduction with the respect of the standard Domino where is the clock period must be at least 583.38ps (2×291.69ps). The initial and the enhanced designs have as clock cycle reduction of 42.17% and 59.11% respectively with respect to the Wave Domino design where a clock signal period of at least 328.7ps (2×164.35ps) is required. The above comparison results are shown in Fig. 4.9.

The mean energy consumption per cycle is 3.98pJ and 3.50pJ for the initial proposed and the enhanced proposed dynamic designs respectively, which results in consumption decrease of 12.06% for second design approach. This energy

consumption improvement is related to the reduced energy requirements during the precharge phase, since the internal nodes of the nMOS network are not precharged in the enhanced dynamic design as it is the case in the initial dynamic design. The mean energy consumption of the standard Domino design is 2.725pJ, which results in energy consumption increase by 46.10% and 28.49% for the initial proposed, the enhanced proposed dynamic designs respectively. In the Wave Domino design the mean energy consumption is 4.97pJ per cycle. Thus, the initial proposed and the enhanced proposed dynamic designs reduce the energy consumption by 19.88% and 29.54% over the Wave Domino design respectively. The above comparisons are shown in Fig 4.10.

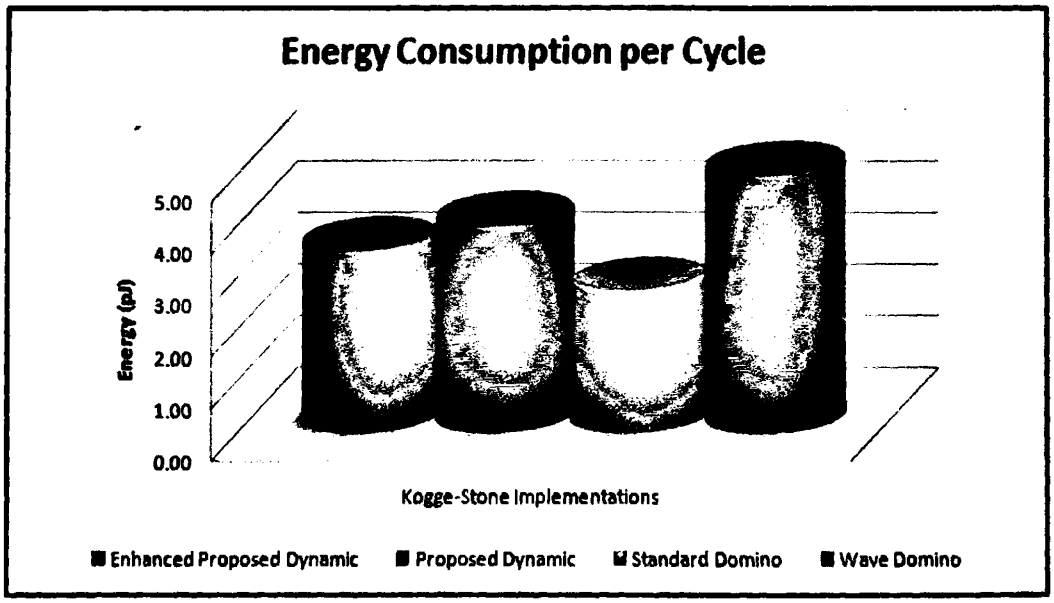
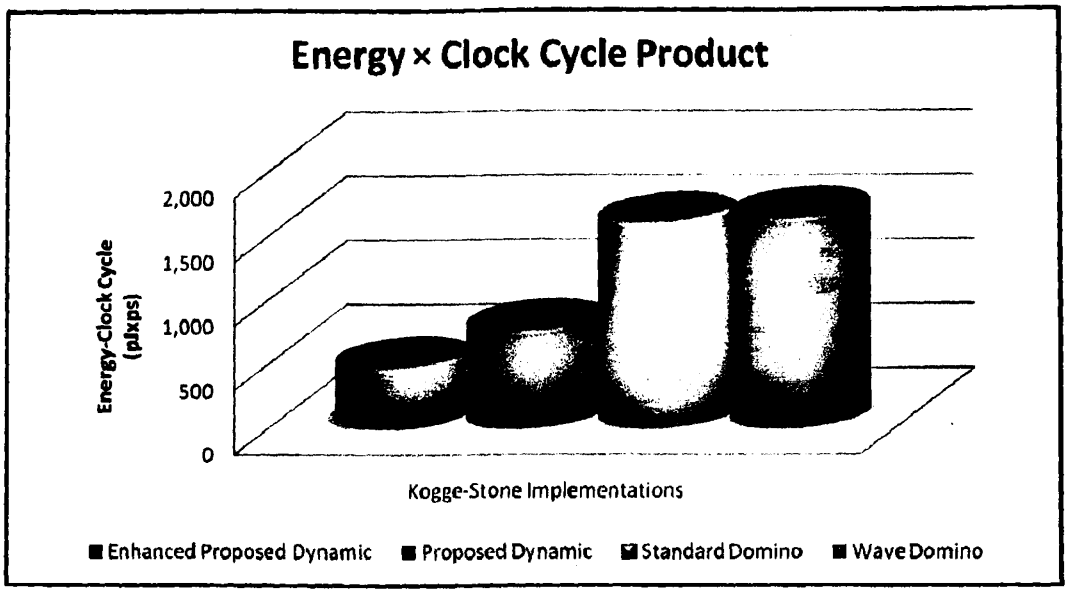


Figure 4.10 Energy consumption per cycle of Kooge-Stone implementations

The energy consumption increment of the proposed approach over the standard Domino design is related to the addition of the extra dynamic NOT gates, in order to maintain the pipeline operation. Due to the construction of the Kogge-Stone CLA unit, the number of these gates is rather high (a situation which is not the typical case in a general circuit design).



Figuré 4.11 Energy-Clock cycle product of Kooge-Stone implementations

The energy-clock cycle products of the proposed dynamic designs (initial and enhanced) are 756.73 pJxps and 470.58 pJxps for the initial proposed and the enhanced proposed dynamic designs respectively, which results in energy-clock cycle product reduction by 52.40% and 70.40% respectively over the Standard Domino design where the product is 1589.66 pJxps. In addition, the energy-clock cycle products are reduced by 53.67% and 71.19% for the initial proposed and the enhanced proposed dynamic designs respectively over the Wave Domino design, in which the energy-clock cycle product is 1633.28 pJxps. Graphical comparisons are shown in Fig 4.11.

The internal nodes precharging capability of the nMOS networks during the precharge phase of the first proposed dynamic design has been verified by the simulations. Next in Fig. 4.12, the precharging of the internal node in the OR-NAND complex gate used in the design of the CLA unit according to the proposed design technique is presented. Note, that these waveforms correspond to the worst

case scenario and that this gate has the highest nMOS network parasitic capacitance in the design.

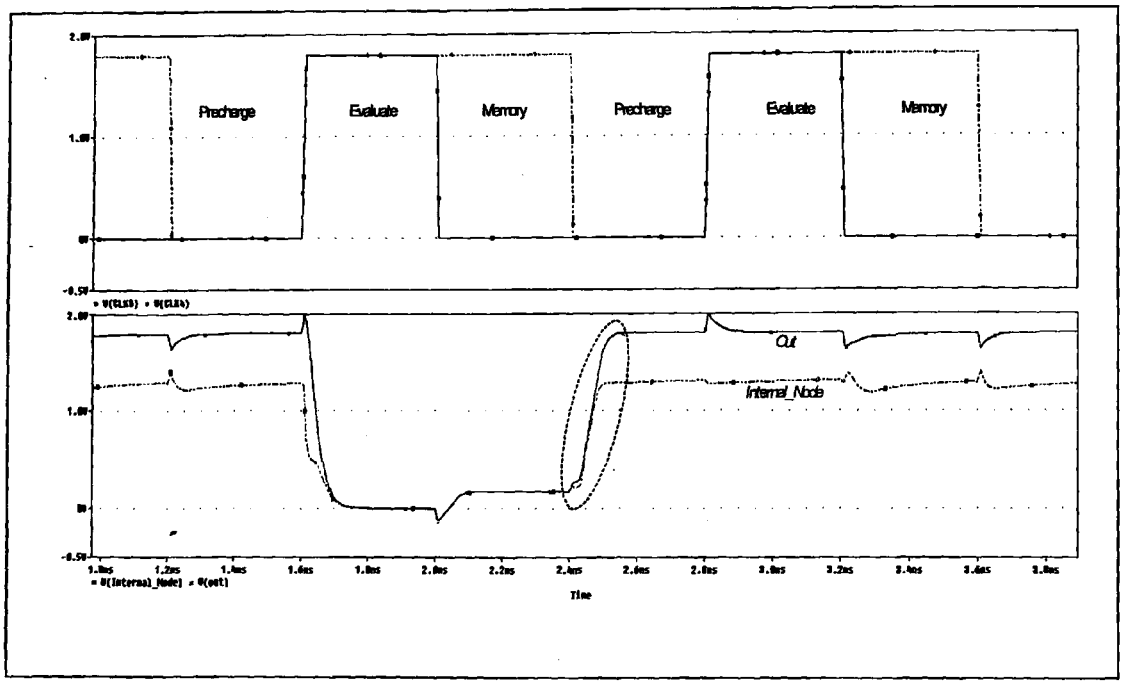


Figure 4.12 Internal node precharging during the precharge phase

Finally, the silicon area, estimated by the sum of the transistor widths in each design, is increased by 8.02% for both proposed dynamic designs with respect to the Standard Domino design and it is reduced by 9.9% with respect to the Wave Domino design, as it shown in Fig. 4.13.

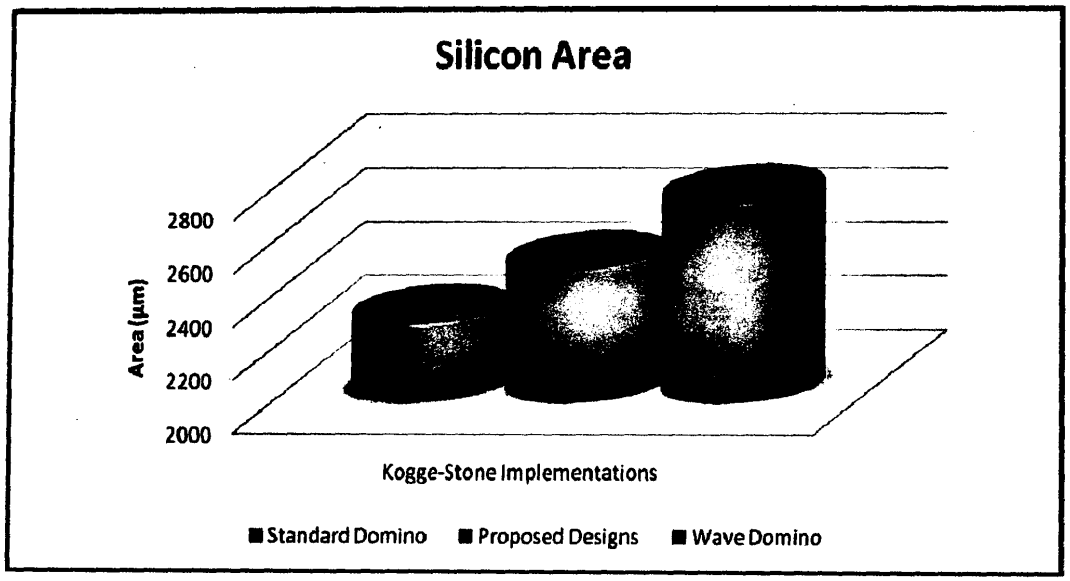


Figure 4.13 Silicon area comparisons.

CHAPTER 5. NEW HIGH SPEED MANCHESTER CARRY CHAIN ADDERS

5.1. Introdution

In this chapter, an efficient implementation of a new dynamic topology of the Manchester carry chain adder in multi-output domino CMOS logic is proposed. The carries of this adder are computed in parallel by two independent carry chains. Due to its limited carry chain length the use of the proposed adder module for the implementation of wider adders leads to significant operating speed improvement compared to the corresponding adders based on the standard Manchester carry chain adder module.

5.2. New High-Speed Double Carry Chain Adders

Manchester Carry Chain adders can efficiently be designed in CMOS logic. As mentioned previously, due to technological constraints the length of their carry chains is limited to 4 bits. However, these 4-bit adder blocks are used extensively in the literature [18], [20], [28] in the design of wider adders.

In the following we propose the design of an 8-bit adder module which is composed of two independent carry chains which have the same length (measured as the maximum number of series connected transistors) as the 4-bit Manchester Carry Chain adders. According to our simulation results, the use of the proposed adder as the basic block, instead of the 4-bit Manchester Carry Chain adder, can lead to high-speed adder implementations.

The derived here carry equations are similar to those for the Ling carries proposed in [32]-[34]. The derived carry equations allow the even carries to be computed separately of the odd ones. This separation allows the implementation of the carries by two independent 4-bit carry chains; one chain computes the even carries, while the other chain computes the odd carries. In the following the design of the proposed 8–bit Manchester Carry Chain adder is analytically presented.

As we mentioned in section 3.5, the computation of the carry signals is based on the following recursive formula:

$$c_i = g_i + z_i \cdot c_{i-1} \qquad (1)$$

$$c_i = g_i + z_i g_{i-1} + z_i z_{i-1} g_{i-2} + \ldots + z_i z_{i-1} \ldots z_1 g_0 + z_i z_{i-1} \ldots z_0 c_{-1} \qquad (2)$$

Where,

$g_i = a_i \cdot b_i$ Generate signal

$z_i = t_i = a_i + b_i$

$z_i = p_i = a_i \oplus b_i$ } Propagate signals

A. Even carry computation

For $i=0$ and $z_0=t_0$, from relation (1) we get that $c_0 = g_0 + t_0 \cdot c_{-1}$. Since the relation $g_i = g_i \cdot t_i$ holds, we get that $c_0 = t_0 \cdot (g_0 + c_{-1}) = t_0 \cdot h_0$, where $h_0 = g_0 + c_{-1}$ is the new carry.

From relation (2), for $i=2$ and $z_i = p_i$, we get that

$$c_2 = g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_{-1}$$

Since $g_i + p_i \cdot g_{i-1} = g_i + t_i \cdot g_{i-1}$ and $p_i = p_i \cdot t_i$ we have

$$c_2 = t_2(g_2 + g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_{-1}) = \quad t_2(g_2 + g_1 + p_2 p_1 t_0 (g_0 + c_{-1})) = t_2 \cdot h_2,$$

where

$h_2 = g_2 + g_1 + p_2 p_1 t_0 (g_0 + c_{-1})$ is the new carry.

In the same way the new carries for $i = 4, 6$ are computed as

$h_4 = g_4 + g_3 + p_4 p_3 t_2 (g_2 + g_1 + p_2 p_1 t_0 (g_0 + c_{-1}))$, and

$h_6 = g_6 + g_5 + p_6 p_5 t_4 (g_4 + g_3 + p_4 p_3 t_2 (g_2 + g_1 + p_2 p_1 t_0 (g_0 + c_{-1})))$

B. Odd carry computation

The new carries for the odd values of i are computed according to the methodology proposed for the even carries as follows:

$h_1 = g_1 + g_0 + p_1 p_0 c_{-1}$

$h_3 = g_3 + g_2 + p_3 p_2 t_1 (g_1 + g_0 + p_1 p_0 c_{-1})$

$h_5 = g_5 + g_4 + p_5 p_4 t_3 (g_3 + g_2 + p_3 p_2 t_1 (g_1 + g_0 + p_1 p_0 c_{-1}))$

$h_7 = g_7 + g_6 + p_7 p_6 t_4 (g_5 + g_4 + p_5 p_4 t_3 (g_3 + g_2 + p_3 p_2 t_1 (g_1 + g_0 + p_1 p_0 c_{-1})))$

Let $G_i = g_i + g_{i-1}$ and $P_i = p_i \cdot p_{i-1} \cdot t_{i-2}$ are the new generate and propagate signals respectively, where $g_{-1} = c_{-1}$, $t_{-1} = 1$. Then, the following equations are derived for the new carries for even values of i:

$$h_2 = G_2 + P_2 G_0$$

$$h_4 = G_4 + P_4G_2 + P_4P_2G_0$$

$$h_6 = G_6 + P_6G_4 + P_6P_4G_2 + P_6P_4P_2G_0$$

while for odd values of i, the equations for the new carries are rewritten as follows:

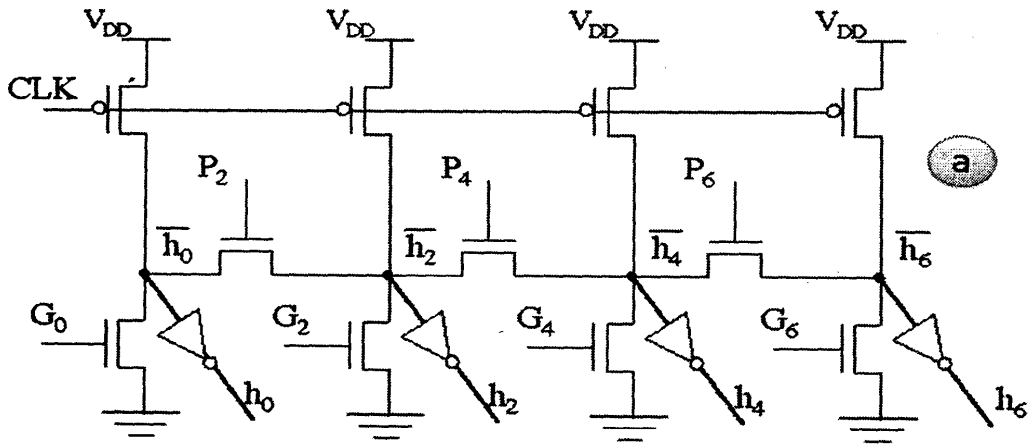
$$h_1 = G_1 + P_1c_{-1}$$

$$h_3 = G_3 + P_3G_1 + P_3P_1c_{-1}$$

$$h_5 = G_5 + P_5G_3 + P_5P_3G_1 + P_5P_3P_1c_{-1}$$

$$h_7 = G_7 + P_7G_5 + P_7P_5G_3 + P_7P_5P_3G_1 + P_7P_5P_3P_1c_{-1}$$

From the above equations it is evident that the groups of even and odd new carries can be computed in parallel by different carry chains in multi-output domino CMOS logic as shown in Fig. 5.1.



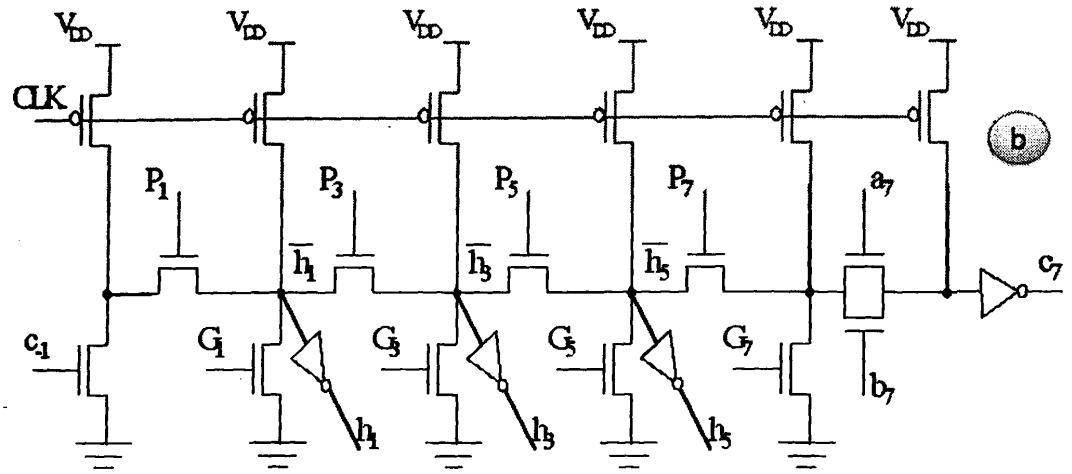


Figure 5.1 Proposed carries implementation the even carry chain (a), odd carry chain (b)

The new generate and propagate signals G_i, P_i can be easily proven that are mutually exclusive, avoiding false node discharges. Their domino CMOS implementation is shown in Fig. 5.3.

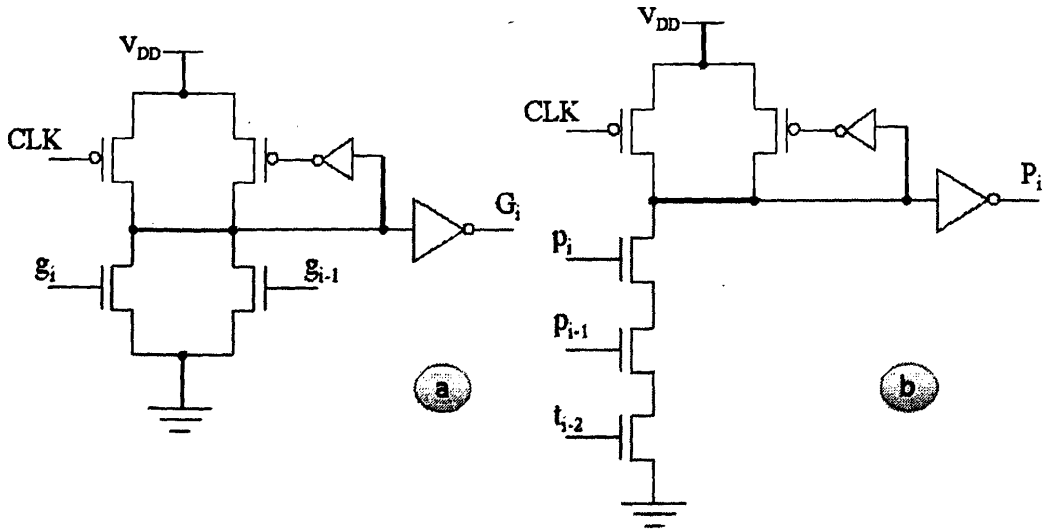


Figure 5.2 The new generate (a) and propagate (b) signals implemented in domino CMOS logic.

Between the new and the conventional carries holds that $c_{i-1}=t_{i-1}\cdot h_{i-1}$, therefore the sum bits are computed as $s_i = p_i \oplus (t_{i-1}\cdot h_{i-1})$. According to [17], [18] the computation of the sum bits can be performed as follows:

$$s_i = \overline{h_{i-1}}\cdot p_i + h_{i-1}\cdot(p_i \oplus t_{i-1}) \qquad (3)$$

for i>0, while $s_0 = p_0 \oplus c_{-1}$.

Relation (3) can be implemented using a 2→1 multiplexer that selects either p_i or $p_i \oplus t_{i-1}$ according to the value of h_{i-1} as shown in Fig. 5.3.

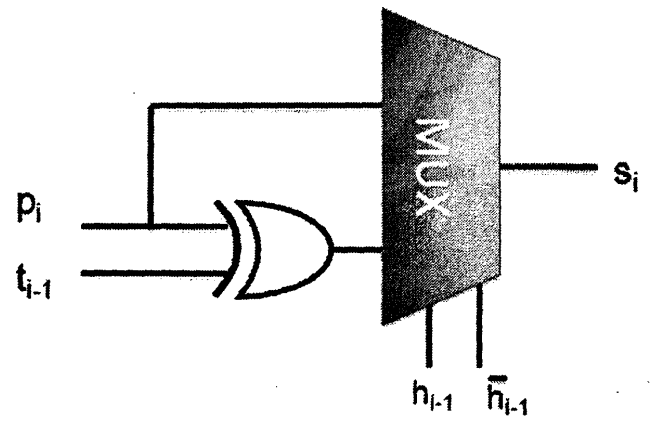


Figure 5.3 Sum bit implementation

Taking into account that an XOR gate introduces equal delay with a 2→1 multiplexer and both terms p_i and $p_i \oplus t_{i-1}$ are computed faster than h_i, then no extra delay is introduced by the use of the proposed carries for the computation of the sum bits according to (3).

For the implementation of the sum signals the domino chain is terminated and static CMOS logic is used for the $p_i \oplus t_{i-1}$ gate and the final 2→1 multiplexer. The design of the XOR gate is shown in Fig. 3.13. An efficient static CMOS implementation of the 2→1 multiplexer is shown in Fig. 5.4.

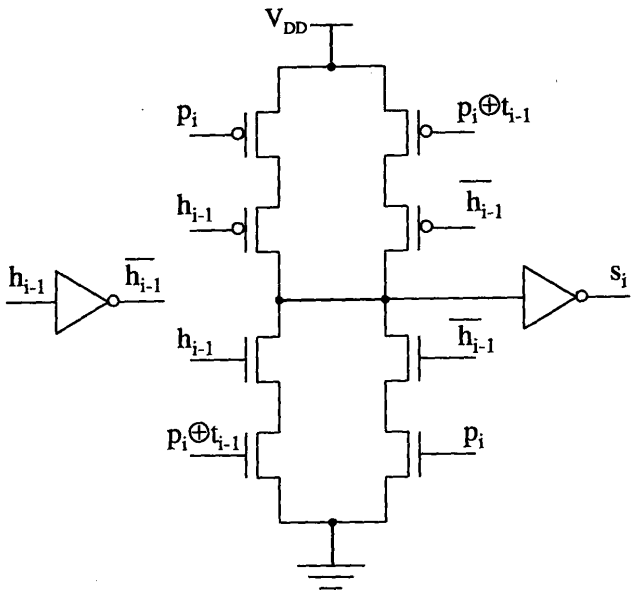


Figure 5.4 Static CMOS implementation of the 2→1 multiplexer

5.3. Manchester Carry Chain Design Issues and Comparisons

To evaluate the speed performance of the Proposed design over the Conventional one and the Proposed design over the Amin's design (see chapter 3.5.1), multi-bit adders have been designed according to the carry chain principle given in Fig. 5.5(a) and 5.5(b) respectively and simulated using SPECTRE in a standard 90nm CMOS technology (V_{DD}=1V). The conventional 8-bit Manchester Carry Chain adder is designed by cascading two 4-bit Manchester Carry Chain modules, while the 16-bit Manchester Carry Chain adder by cascading four 4-bit Manchester Carry Chain adder modules and so on. The proposed 16-bit Manchester Carry Chain adder is designed by cascading two of the proposed 8-bit Manchester Carry Chain adder modules and so on. Amin's 8-bit adder is designed by cascading two 4-bit Chains that contain the required (sk) signal generation gate, and so on.

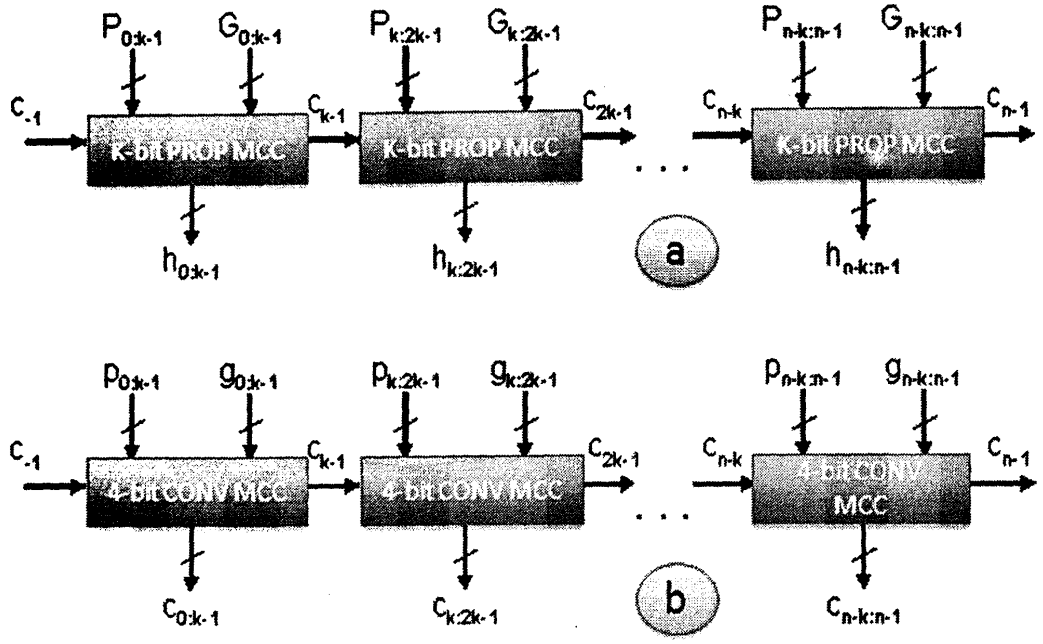


Figure 5.5 Ripple carry chains based on k-bit MCC adder modules

The performance improvement provided by the proposed design approach can be easily understood by considering the simplified timeslot diagram presented in Fig. 5.6. The various computations are grouped by the timeslots they require in the whole process. Each group is represented by a rectangle which in the x-axes expresses the time duration that is needed for the completion of the pertinent calculation. The time needed to create the propagation (p) and generation (g) signals is equal in both techniques. However, in the proposed design technique a small extra time is required for the computation of the new generate (G) and propagate (P) signals but after that the odd and the even carries are calculated simultaneously. This parallel calculation is responsible for the performance improvements achieved by the new design approach.

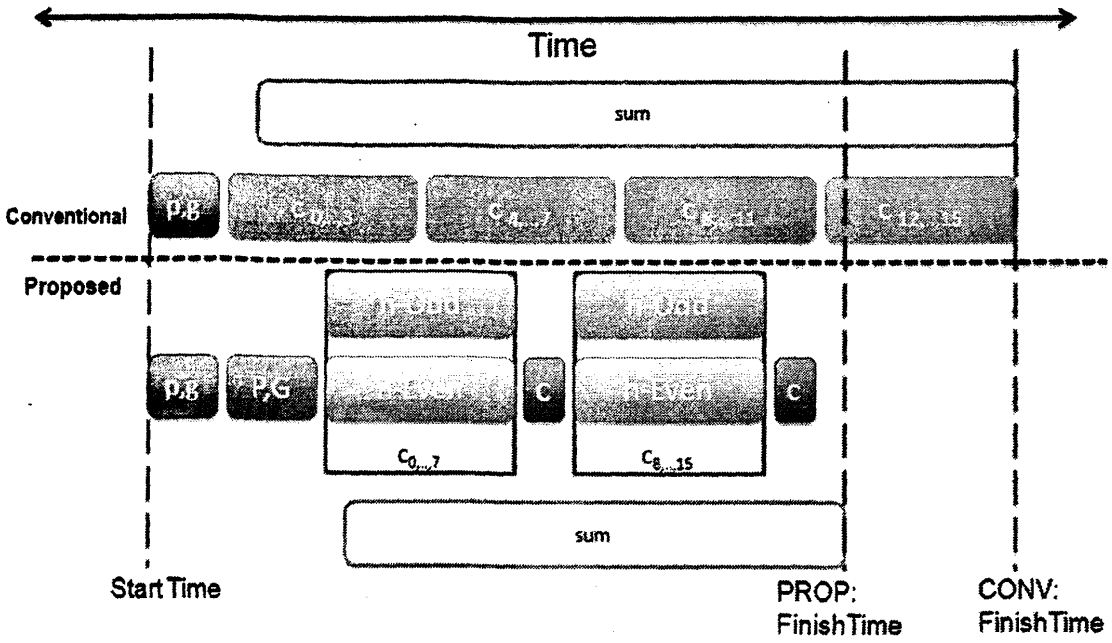


Figure 5.6 Propagation delay timeslots for the 16-bit adder PROP vs. CONV

The simulation results for the best clock cycle period achieved in the 8-bit and 16-bit implementations according to the carry propagation delays of the proposed, the conventional and the Amin's designs are presented in Fig. 5.7. The clock period for the proposed 8-bit design should be at least 431ps, which provides a performance improvement of 4.73% over the conventional design, where the clock period should be at least 452.42ps, while the proposed design is improved by 7.18% with the respect to the simulation results on the Amin's design where the clock period should be at least 464.36ps. Moreover, in the case of 16-bit adders, the clock period of the proposed design should be at least 704.8ps and it is increased by 23.08% with respect to the conventional design where the pertinent time duration should be at least 916.3ps. Moreover, the proposed design outperforms by 11.8% over the Amin's design where the clock period should be at least 799.02ps.

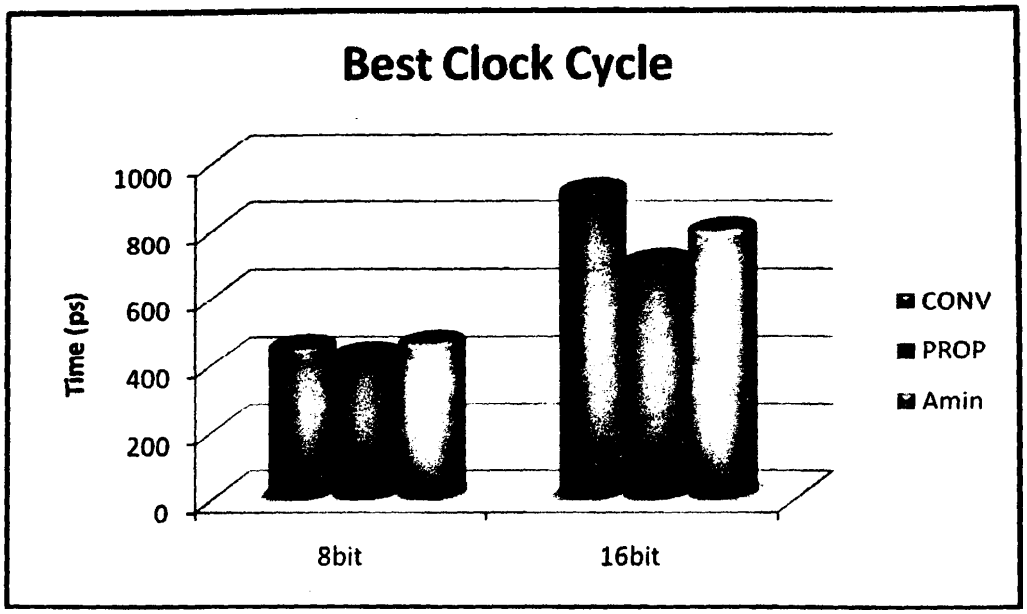


Figure 5.7 Best clock cycle period for the implementations of the 8-bit and 16-bit adders

The mean energy consumption per cycle of the proposed 8-bit design is 2.93×10^{-13}J, which results in a consumption increment by 44.75% over the conventional design, where the mean energy is 1.62×10^{-13}J per cycle. Moreover, the mean energy consumption of the proposed design is increased by 42.02% with respect to the pertinent results on the Amin's design, where the mean energy consumption is 1.7×10^{-13}J. Considering the 16-bit adders, the mean energy consumption of the proposed design is 5.62×10^{-13}J and it is increased by 42.53% over the conventional design where the energy consumption is 3.23×10^{-13}J. Moreover, the consumption of the proposed design increases by 45.2% over the energy consumption of the Amin's design which is 3.08×10^{-13}J. The above comparison results are presented in Fig. 5.8.

This energy consumption increment of the proposed design is mainly related to the additional gates used to generate the new generate and propagate signals G_i, P_i.

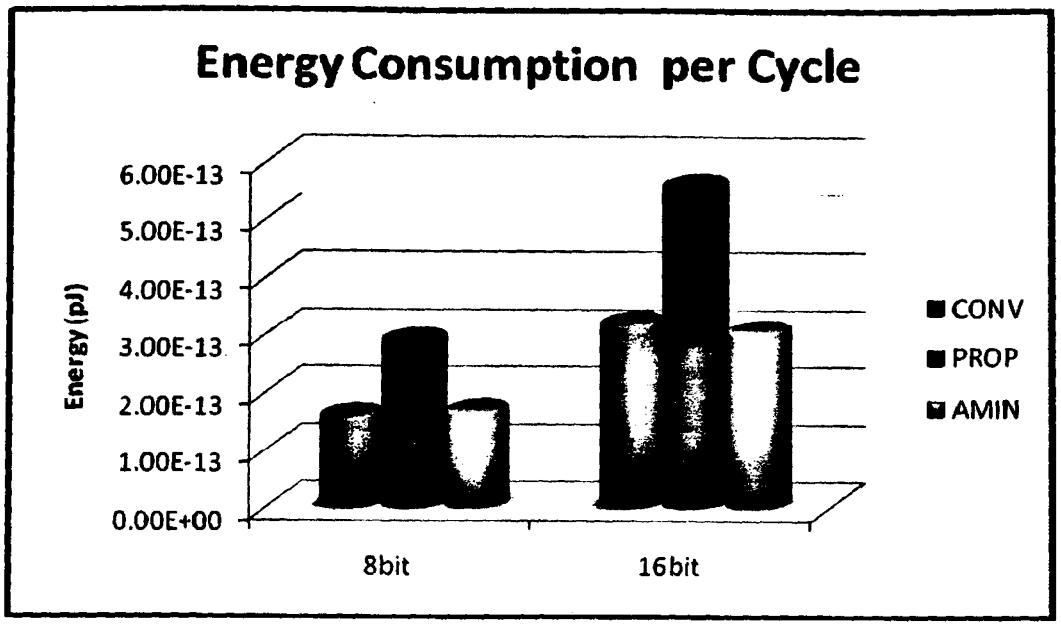


Figure5.8 Energy consumption per cycle for the implementations of the 8-bit and 16-bit adders

The energy×clock cycle product of the proposed 8-bit design is 1.26×10^{-10}J×ps, which results in an increment of 42.00% over the conventional design, where this product is 7.33×10^{-11}J×ps. Moreover, the energy×clock cycle product of the proposed design is increased by 37.53% over the Amin's design where this product is 7.89×10^{-11}J ×ps. In addition, in the case of the 16-bit adders, the energy×clock cycle product of the proposed design is 3.96×10^{-10}J×ps, which results in an increment of 25.28% over the conventional design, where this product is 2.96×10^{-10}J×ps. The energy×clock cycle product of the proposed design is increased by 37.87% over the Amin's design where this product is 2.46×10^{-10}J ×ps. The above comparisons are shown in Fig. 5.9.

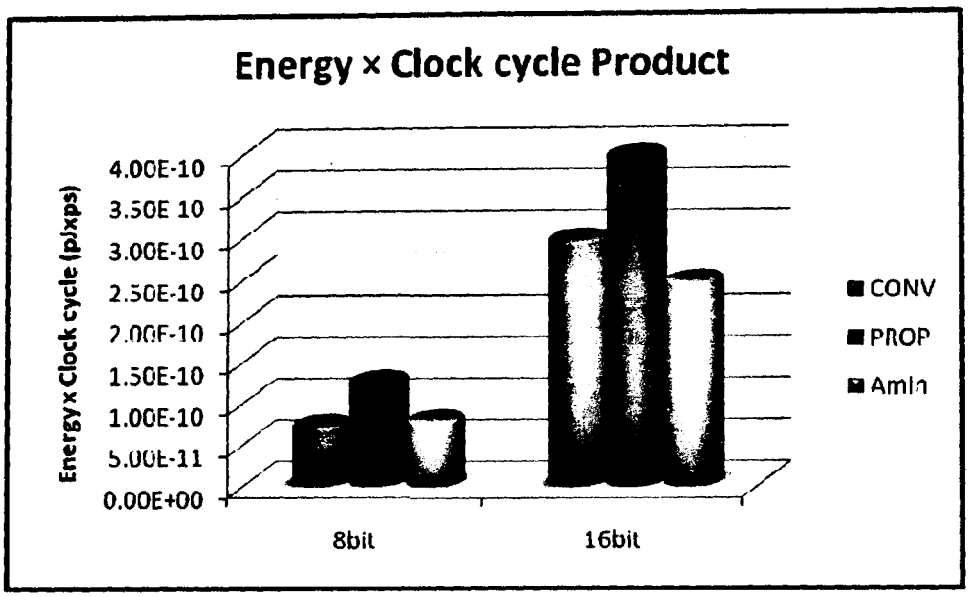


Figure 5.9 Energy-Clock Cycle product for the 8-bit and 16-bit adders

Extending the Amin's technique for higher number of bits does not provide better results over the standard Manchester design. Therefore, we exclude Amin's design in the rest of the document.

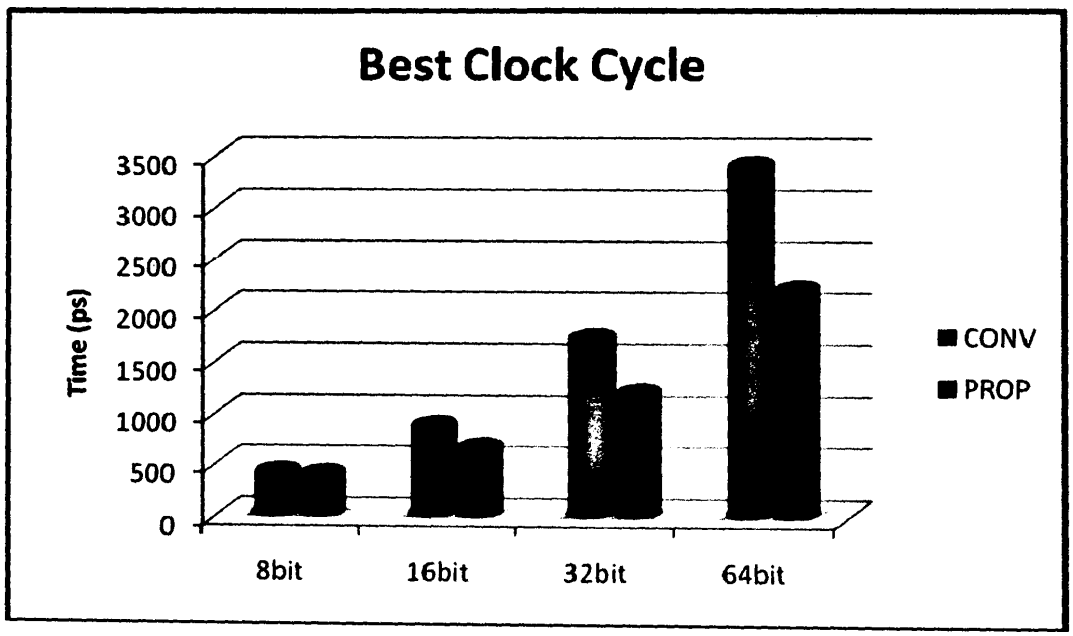


Figure 5.10 Best clock cycle period of the various adder implementations

Next, considering wider adders, the clock period of the proposed 32-bit design should be at least 1.23ns, which provides a performance improvement of 30.05% over the conventional design where the clock period should be at least 1.76ns. In addition, the clock period for the proposed 64-bit design is improved by 35.08% with the respect to the corresponding simulation results on the conventional design, where the clock period should be at least 3.44ns. The performance comparisons are shown in Fig. 5.10.

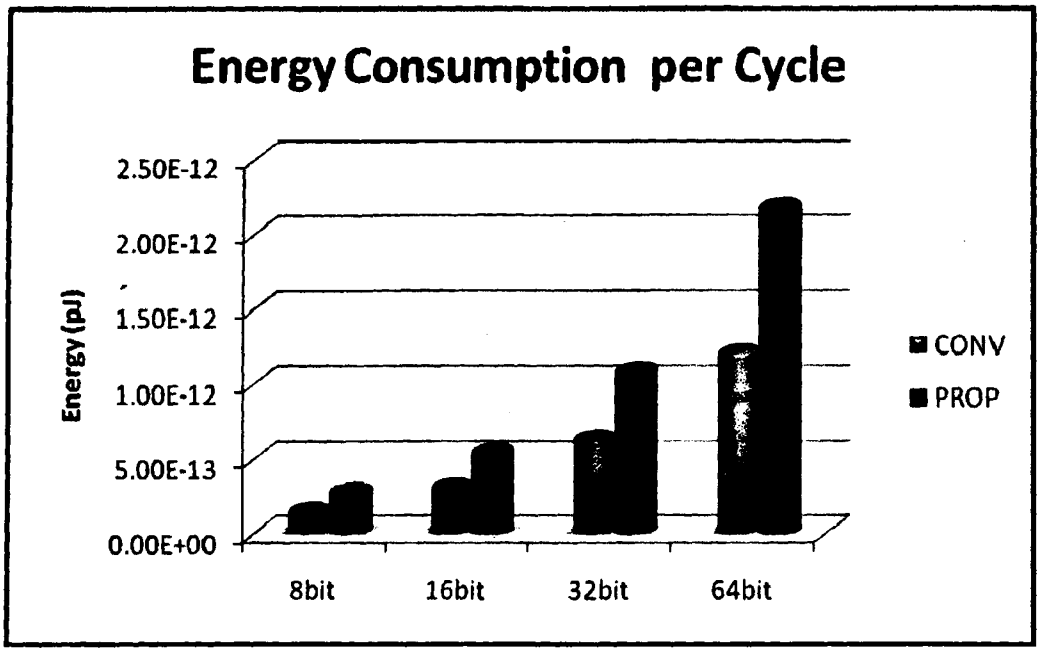


Figure 5.11 Energy consumption of the various adder implementations

The mean energy consumption per cycle of the proposed 32-bit design is 1.1×10^{-12}J, which results in a consumption increment of 44.75% over the conventional design where the energy consumption is 6.43×10^{-13}J. Moreover, the energy consumption per cycle of the proposed 64-bit design is 2.19×10^{-12}J and it is increased by 44.29% with the respect to the conventional design, where the mean energy consumption is 1.22×10^{-12}J. The comparisons are shown in Fig. 5.11.

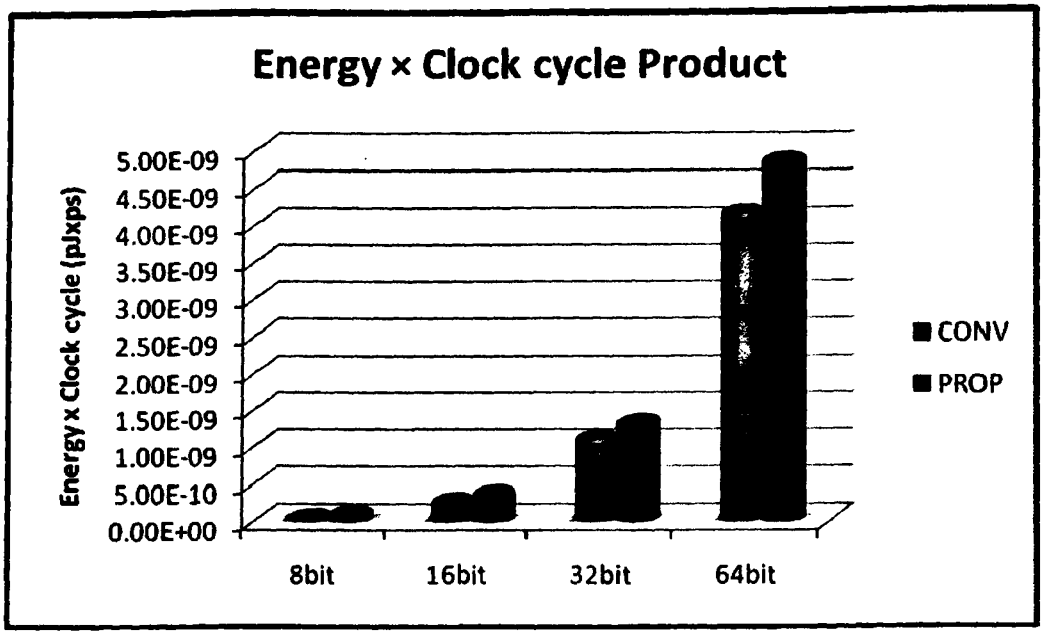


Figure 5.12 Energy × clock cycle product of the various adder implementations

The energy×clock cycle product of the proposed 32-bit adder is 1.36×10^{-9}J×ps, which results in a 16.44% increment over the conventional design, where the energy×clock cycle product is 1.13×10^{-9}J×ps. In addition, the energy×clock cycle product of the proposed 64-bit adder is 4.89×10^{-9}J×ps and it increases by 14.19% with the respect to the conventional design, where the energy×clock cycle product is 4.19×10^{-9}J×ps. The comparisons are shown in Fig. 5.12.

Tables 5.1, 5.2 and 5.3 summarize the above experimental results on performance, energy consumption and energy×clock cycle product, respectively.

Table 5.1 Simulation results on the best clock cycle period.

Clock Cycle (ps)	8-bit	16-bit	32-bit	64-bit
Conventional	452.42	916.3	1763.84	3436.4
Proposed	431	704.8	1233.84	2230.88
Amin's	464.36	799.02	-	-
Proposed vs. Conventional	4.76%	23.08%	30.05%	35.08%
Proposed vs. Amin's	7.18%	11.79%	-	-

Table 5.2 Energy consumption simulation results.

Energy (pJ)	8-bit	16-bit	32-bit	64-bit
Conventional	1.62×10^{-13}	3.23×10^{-13}	6.43×10^{-13}	1.22×10^{-12}
Proposed	2.93×10^{-13}	5.62×10^{-13}	1.1×10^{-12}	2.19×10^{-12}
Amin's	1.7×10^{-13}	3.08×10^{-13}		
Proposed vs. Conventional	-44.75%	-42.53%	-41.55%	-44.29%
Proposed vs. Amin's	-42.02%	-45.20%		

Table 5.3 Energy × clock cycle product simulation results.

Energy× Clock Cycle (ps×pJ)	8-bit	16-bit	32-bit	64-bit
Conventional	7.33×10^{-11}	2.96×10^{-10}	1.13×10^{-9}	4.19×10^{-9}
Proposed	1.26×10^{-10}	3.96×10^{-10}	1.36×10^{-9}	4.89×10^{-9}
Amin's	7.89×10^{-11}	2.46×10^{-10}		
Proposed vs. Conventional	-42.00%	-25.32%	-16.75%	-14.37%
Proposed vs. Amin's	-37.53%	-37.87%		

CHAPTER 6. CONCLUSIONS

We proposed new dynamic design techniques for the implementation of high performance arithmetic circuits like the well known Kogge-Stone adders. According to these design approaches, a three-phase clocking scheme is used that provides the ability to design high performance pipeline structures without the need to use memory elements. Furthermore, a pre-evaluation operation is introduced, which is hidden inside the precharge phase of each gate and provides significant speed improvements. Simulation results on Kogge-Stone adder implementations verified the expected gains.

A disadvantage of the proposed approaches is the need of additional clock signals. However the generation of these clock signals is a one-time cost that does not increase with circuit complexity. Moreover, in a pipeline design fashion where each stage is fed with a dedicated clock signal(s), the clock signals distribution is not a hard design task of increased cost. Especially, in structured circuits, like arithmetic ones, this cost is quite small. Finally, in order to cope with possible skew related problems among the clock signals, commonly used skew hardened dynamic design techniques proposed in the open literature can be adopted [15].

A second research activity in this thesis is related to the dynamic design of Manchester adders. The Manchester carry chain is an efficient and widely adopted design approach to construct carry look-ahead adders. We present a new Manchester design style that is based on two independent carry chains. Each chain computes, in parallel with the other, half of the carries. This way the speed performance is significantly improved with respect to earlier Manchester carry chain topologies. On the other hand, the energy consumption is getting worse and the same stands for the energy×clock cycle product. However, the latter is improved as the number of bits is increased. The proposed design technique has been applied for the implementation of 8-bit, 16-bit, 32-bit and 64-bit adders in multi-output Domino logic and the simulation results verified its performance efficiency.

REFERENCES

[1] J.M. Rabaey, A. Chandrakasan and B. Nikolic,: Digital Integrated Circuits: A Design Perspective. Prentice Hall (2003)

[2] D. Harris and M. A. Horowitz, "Skew-Tolerant Domino Circuits," IEEE Journal of Solid-State Circuits, vol. 32, no. 11, pp. 1702-1711, 1997.

[3] K. Bernstein, J. Ellis-Monaghan, and E. Nowak, "High-Speed Design Styles Leverage IBM Technology Prowess," IBM Micro News, vol. 4, no. 3, 1998.

[4] R. Heald, K. Aingaran, C. Amir, M. Ang et.al., "A Third-Generation SPARC V9 64-b Microprocessor," IEEE Journal of Solid-State Circuits, vol. 35, no. 11, pp. 1526-1538, Nov. 2000.

[5] S. D. Naffziger, G. Colon-Bonet, T. Fischer, R. Riedlinger, T. J. Sullivan and T. Grutkowski, "The Implementation of the Itanium 2 Microprocessor," IEEE Journal of Solid-State Circuits, vol. 37, no. 11, pp. 1448-1460, May 2002.

[6] C, Cornelius, S. Koppe and D. Timmermann, "Dynamic Circuit Techniques in Deep Submicron Technologies: Domino Logic Reconsidered," International Conference on IC Design and Technology (ICICDT), pp. 53-56, 2006.

[7] S. Wijerante, N. Siddaiah, S. Mathew, M. Anders, R. Krishnamurthy, J. Anderson, S. Hwang, M. Ernest, M. Nardin, "A 9GHz 65nm Intel Pentium 4 Processor Integer Execution Core," International Solid-State Circuits Conference (ISSCC), pp. 353-355, 2006.

[8] Z. Wang, G.A. Jullien, W.C. Miller, J. Wang and S.S Bizzan, "Fast Adders Using Enhanced Multiple-Output Domino Logic," IEEE Journal of Solid-State Circuits, vol. 32, no. 2, pp. 206-214, 1997.

[9] S-M. Yoo and S-M Kang, "Improved Domino Structures Effective for High Performance Design," Electronics Letters, vol. 35, no. 5, pp. 367-368, 1999.

[10] J-R. Yuan, C. Svensson and P. Larsson, "New Domino Logic Precharged by Clock and Data," Electronics Letters, vol. 29, no. 25, pp. 2188-2189, 1993.

[11] Rjoub, O. Koufopavlou and S. Nikolaidis, "Low-Power / Low-Swing Domino CMOS Logic," IEEE Int. Symp. on Circuits and Systems, pp. 13-16, 1998.

[12] Rao, Th. Haniotakis, Y. Tsiatouhas, and H. Djemil, "The Use of Pre-evaluation Phase in Dynamic CMOS Logic," in Proc. IEEE CS Annual Symposium on VLSI (ISVLSI), pp. 270–271, 2005.

[13] Amirabady, A. Afzali-Kusha, Y. Mortazavi and M. Nourani, "Clock Delayed Domino Logic with Efficient Variable Threshold Keeper," IEEE Tran. on VLSI Systems, vol. 15, no. 2, pp. 125-134, 2007.

[14] Rao, Th. Haniotakis, Y. Tsiatouhas and V. Kaky,: A New Dynamic Circuit Design Technique for High Performance TSC Checker Implementations. In: IEEE Int. On-Line Testing Symposium, pp. 52-57 (2004)

[15] D. Harris,: Skew-Tolerant Circuit Design. Morgan Kaufmann Publishers (2001)

[16] P.M. Kogge and H.S. Stone,: A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations. IEEE Transactions on Computers, 22(8), pp. 786--793 (1973)

[17] John P. Uyemura: Introduction to VLSI Circuits and Systems: John Wiley & Sons,INC (2002).

[18] Neil Weste and Kamran Eshraghian: Principles of CMOS CLSI Desgign A System Prespective: ADDISON-WESLEY PUBLISHING COMPANY (1985).

[19] K. Hwang, "Computer Arithmetic: Principles, Architecture, and Design," Wiley, New York, 1979.

[20] Parhami, "Computer Arithmetic, Algorithms and Hardware," Oxford, University Press, New York, Oxford, 2000.

[21] Koren, "Computer Arithmetic Algorithms," A. K. Peters, 2nd edition, 2002.

[22] M. D. Ercegovac and T. Lang, "Digital Arithmetic," Morgan Kaufmann Publishers, 2004.

[23] Weinberger and J. L. Smith, "A Logic for High Speed Addition," Nat. Bur. Stand. Circ., vol. 591, pp. 3–12, 1958.

[24] N. Weste, D. Harris, "CMOS VLSI Design, A Circuit and System Perspective," Addison Wesley, 2004.

[25] P. K. Chan, M. D. F. Schlag, "Analysis and Design of CMOS Manchester Adders with Variable Carry-Skip," IEEE Trans. on Computers, vol. 39, no. 8, pp. 983-992, Aug. 1990.

[26] Z. Wang, G. Jullien, W. Miller, J. Wang, S. Bizzan, "Fast Adders Using Enhanced Multiple-Output Domino Logic," IEEE J. Solid State Circuits, vol. 32, no. 2, pp. 206-214, Feb. 1997.

[27] S. Perri, P. Corsonello, F. Pezzimenti, and V. Kantabutra, "Fast and Energy-Efficient Manchester Carry-Bypass Adders," IEE Proc. Circuits Devices Syst., vol. 151, no. 6, pp. 497–502, 2004.

[28] M. Osorio, C. Sampaio, A. Reis, R. Ribas, "Multiple Output Enable-Disable CMOS Differential Logic," Proc. of the 17th Symposium on Integrated Circuits and System Design, pp. 181-185, 2004.

[29] Amin, "Area-Efficient High-Speed Carry Chain," Electronics Letters, vol. 43. no. 23, pp. 1258-1260, Nov. 2007.

[30] G. A. Ruiz, "New Static Multi-Output Carry Look-Ahead CMOS Adders," IEE Proc. Circuits, Devices and Systems, vol. 144, no. 6, pp. 350-354, Dec. 1997.

[31] G. A. Ruiz, M. Granda, "An Area-Efficient Static CMOS Carry-Select Adder Based on a Compact Carry Look-Ahead Unit," Microelectronics Journal, vol. 35, no. 12, pp. 93-106, Dec. 2004.

[32] H. Ling, "High-Speed Binary Adder," IBM Journal on Research and Development, vol. 25, pp. 156-166, May 1981.

[33] Efstathiou, H. T. Vergos, D. Nikolos, "Ling Adders in CMOS Standard Cell Technologies," 9th IEEE Inter. Conference on Electronics, Circuits and Systems, (ICECS 2002), vol. 2, pp. 485-489, Sept. 2002.

[34] G. Dimitrakopoulos, D. Nikolos, "High-Speed Parallel-Prefix VLSI Ling Adders," IEEE Trans. on Computers, vol. 54, no. 2, pp. 225-231, Feb. 2005.

[35] S. Vassiliadis, "Recursive Equations for Hardwired Binary Adders," International Journal of Electronics, vol. 67, no. 2, pp. 201-213, Aug. 1989.

[36] N.F. Goncalves and H.J. De Man, "NORA: A Race Free Dynamic CMOS Technique for Pipelined Logic Structures", IEEE Journal of Solid-State Circuits, vol. 18, no. 3, pp. 261-266, 1983.

PUBLICATIONS

Themistoklis Haniotakis, Zaher Owda and Yiorgos Tsiatouhas "Memory-less Pipeline Dynamic Circuit Design Technique" , IEEE Computer Society Annual Symposium on VLSI (ISVLSI-2010), July 2010.

SHORT BIOGRAPHY

Zaher Owda was born in 1982 and he is from Palestine. He graduated from the department of Computer Science at the University of Ioannina in 2008. The academic year 2008-2009 he was accepted in the Graduate Program of the Department of Computer Science, University of Ioannina, and by November 2009 he is a member of the Unit of Medical Technology and Intelligent Information Systems which is a highly innovative and self-contained research unit that resides at the University of Ioannina.

His research interests are in the area of analog and digital VLSI Design and high performance digital circuits.