2013

Timing and Power Optimization Using Mixed-Dynamic-Static CMOS

Hao Xue

Wright State University

TIMING AND POWER OPTIMIZAION USING MIXED-DYNAMIC-STATIC CMOS

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Engineering

By

HAO XUE

B.S., *Taiyuan University of Technology, China*, 2010

2013

WRIGHT STATE UNIVERSITY

WRIGHT STATE UNIVERSITY

GRADUATE SCHOOL

July 1, 2013

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Hao Xue ENTITLED "Timing and Power Optimization Using Mixed-Dynamic-Static CMOS" BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science in Engineering

_____
Chien-In Henry Chen, Ph.D.
Thesis Director

_____
Kefu Xue, Ph.D.
Department Chair

Committee on Final Examination

_____
Chien-In Henry Chen, Ph.D.

_____
Saiyu Ren, Ph.D.

_____
Yan Zhuang, Ph.D.

_____
R. William Ayres, Ph.D.
Interim Dean, Graduate School

Abstract

Xue, Hao. M.S.Egr, Department of Electrical Engineering, Wright State University, 2013. "TIMING AND POWER OPTIMIZAION USING MIXED-DYNAMIC-STATIC CMOS"

An effective approach to timing and power optimization for single clocking and multiple clocking dynamic CMOS designs is presented in this thesis. For the single-clocking scheme dynamic CMOS sub-blocks can be replaced by static CMOS and mixed-dynamic-static CMOS for power minimization. For the multiple-clocking scheme the delay of data ready for use plays more important role than its clock pulse in timing optimization. Power minimization can be achieved by implementing dynamic CMOS sub-blocks with static or mixed-dynamic-static CMOS. In comparison with the benchmark 16-bit carry select adder in dynamic CMOS, the critical path delay is reduced by 41.1% using the single-clock optimization approach; the power and delay are reduced by 43% and 41.1% respectively using the multiple-clock optimization approach. In comparison with the benchmark 64-bit comparator in dynamic CMOS, the critical path delay is reduced by 49% using the single-clock optimization approach; the power and delay are reduced by 43.1% and 49% respectively using the multiple-clock optimization approach.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Acknowledgement

I would like to express my thanks to Dr. Henry Chen, a senior Professor in Electrical Engineering at Wright State University (WSU). He was very helpful and kind through, not only the research process, but through my entire tenure at (WSU). I will always remember and admire him. Dr. Chen involved me in his research by giving me the project idea and helping me throughout my thesis research period. Furthermore, I would also like to thank the staff of the Department of Electrical Engineering for their cooperation and time. Finally, I express my appreciation to Drs. Saiyu Ren and Yan Zhuang for their service as members of my thesis defense committee.

1 INTRODUCTION

1.1 Background

Compare with static CMOS dynamic CMOS has less delay and smaller area, and as tradeoff, it has higher power consumption. Its high speed, which is one of the foremost characteristics in designing integrated circuits (IC's), has led it to play important role in the high performance digital IC market. But in recent years, because of the power hungry of CPU and portable devices such as cell phone, sensors, etc. that is heavy load for given battery and heat dissipation load for limited space, they are designed as static circuit, absolutely, based on the sacrifice of speed.

As seen in Fig. 1.1, the dynamic circuit has two phases of operation, precharge and evaluate, controlled by a single clock. During the precharge phase, Φ is low, PMOS M1 is on and NMOS M2 is off, then output is pulled up to the high voltage (logic 1) through the PMOS M1. During the evaluate phase, Φ is high, PMOS M1 is turned off and the NMOS M2 is turned on, so the output is pulled down to low voltage (logic 0) if any of conduction paths in the NMOS logic in the pull-down network is turned on; otherwise output stays at high voltage (logic 1).

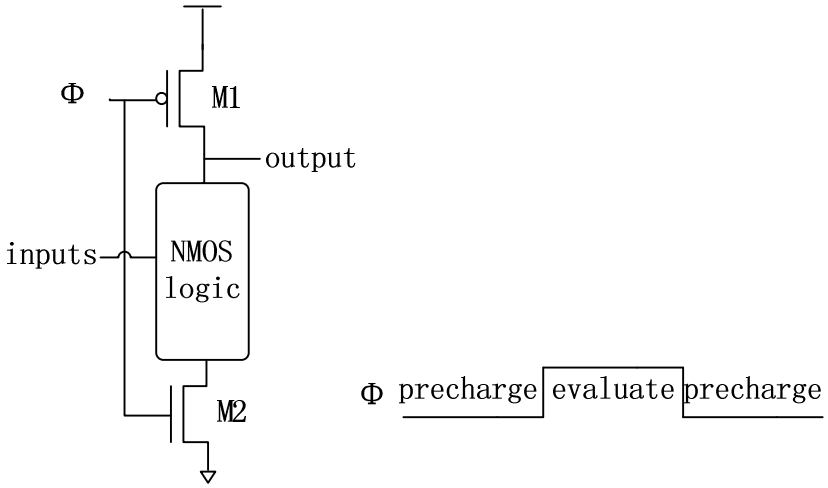


Fig. 1.1 Single-clock half-time dynamic CMOS operation

As shown in Fig. 1.1, dynamic CMOS has only one PMOS transistor, so the input capacitance load of dynamic CMOS is much less than that of static CMOS, which leads to a faster signal propagation. Output of dynamic CMOS is evaluated only by half of the operation time (Clock), which is not so efficient for static CMOS is evaluated by full of the operation time [1]. To figure out the problem, as shown in Fig. 1.2, inverted clock is used to control next stage to make it precharging in evaluate-section and evaluating in precharge-section. Then either stage 1 or stage 2 evaluates at any operating time that means the CMOS is full-time dynamic CMOS.

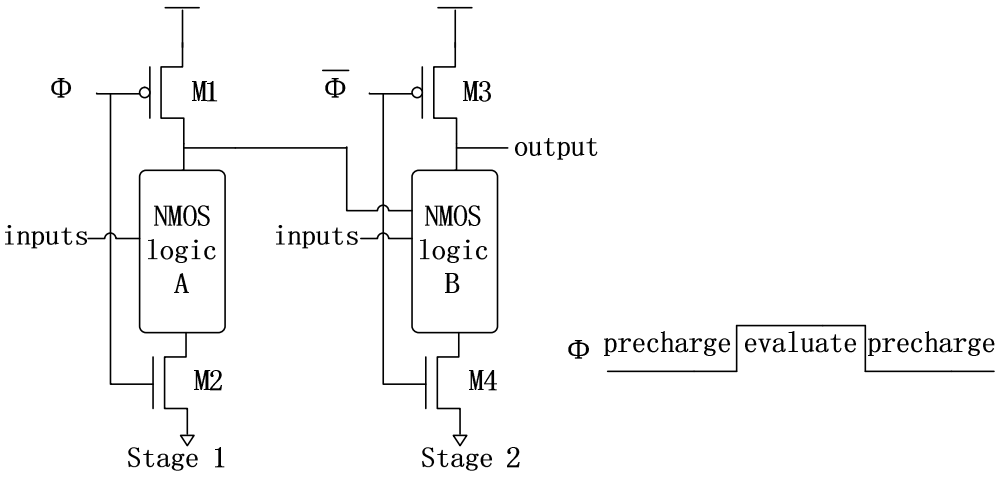


Fig. 1.2 Full-time dynamic CMOS operation

Multiple-clock dynamic CMOS, as shown in Fig. 1.3, is a desirable design to further increase the speed of full-time dynamic CMOS. Multiple same circuits, controlled by respective clock, work in parallel, so next input does not need to wait for the termination of propagation of former input that raises the frequency of obtaining output, in other words, decrease the delay of circuit.

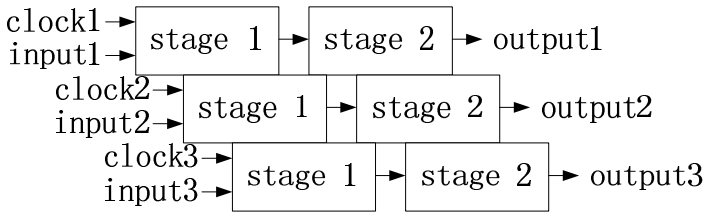


Fig. 1.3 Multiple-clock dynamic CMOS operation

1.2 Research Motivation

In multiple-clock dynamic CMOS circuits, several circuits operate in parallel, shown in Fig. 1.3, so the delay for data use (data efficiency) plays more important role than its clock pulse. The delay and clock pulse of the full-time dynamic CMOS are shown in Fig. 1.4. As long as the delay is retained, the clock pulse can be enlarged to decrease power.

In single-clock dynamic circuit, static CMOS or mixed-dynamic-static CMOS can be applied to replace traditional dynamic CMOS to decrease power consumption if power optimization is prior to timing optimization.

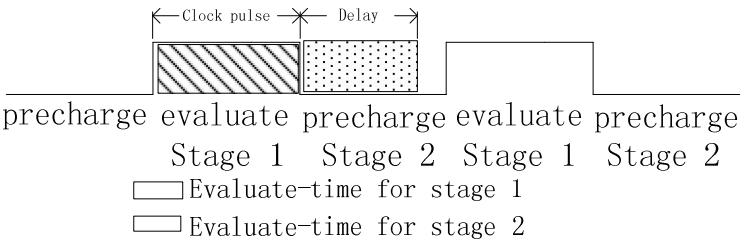


Fig. 1.4 Clock pulse and delay of full-time dynamic CMOS operation

The two stages of dynamic CMOS in Fig. 1.2 have two individual propagation

delays, so two situations of full-time dynamic CMOS operation, shown in Fig. 1.5, are generated.

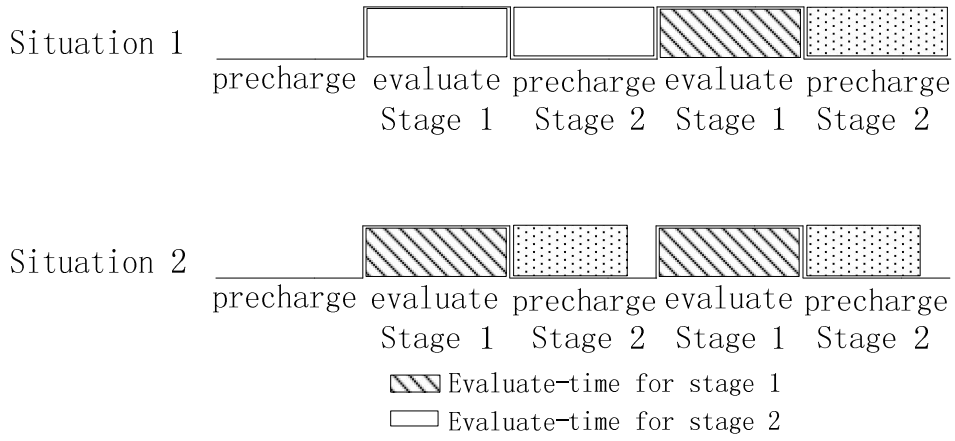


Fig. 1.5 Two situations of full-time dynamic CMOS operation

In situation 1, the delays of stage 1 and stage 2 are almost equal. For single-clock dynamic CMOS, clock pulse cannot be sacrificed, and no extra timing is available during operation, so no resource can be used to do power optimization; timing optimization is the only choice, for which full-time dynamic CMOS should be chosen. For multiple-clock dynamic CMOS, the delay for data use plays more important role than its clock pulse, which can be sacrificed to decrease power-consumption that can be implemented by replacing stage 1 with static or mixed-dynamic-static CMOS. 16-bit carry select adder (CSA) will be an example to prove the theory above in chapter 2.

As seen in Table 1.1, compare with conventional (half-time) dynamic CMOS, the delay of full-time dynamic CMOS, which is chosen for single-clock timing optimization, is decreased by 41.1%; the power and delay of mixed-dynamic-static CMOS, which is the choice for multiple-clock circuit, are reduced by 43% and 41.1%, respectively.

Table 1.1 16-bit parallel binary adder

Platform	Circuit style	Delay (ns)	Clock pulse (ns)	Power (mW)
	Half-time dynamic CMOS	1.34	1.34	4.187
Single-clock (timing optimization)	Full-time dynamic CMOS	0.789	0.79	8.216
Multiple clock	Mixed-dynamic-static CMOS	0.788	1.21	2.388

Notation: Delay: worst-case delay

Clock pulse: the minimum clock pulse under which the CMOS can operate correctly

Power: average power in worst-case operation

In situation 2, the delay of stage 1 is greater than that of stage 2. For the single-clock dynamic CMOS, if timing optimization is prior to power optimization, full-time dynamic CMOS should be chosen; if power optimization is prior to timing optimization, the free time in stage 2 can be utilized to decrease power consumption by replacing stage 2 with static or mixed-dynamic-static CMOS. For the multiple-clock dynamic CMOS, the delay for data use plays more important role than its clock pulse. Then, clock pulse can be increased to decrease power consumption that can be implemented by replacing stage 1 with static CMOS. A 64-bit binary comparator is used as an example in chapter 3.

As we can see in Table 1.2, in comparison with conventional (half-time) dynamic CMOS, the delay of full-time dynamic CMOS, which is chosen for single-clock timing optimization, is decreased by 49%; the power and delay of full-time dynamic CMOS with static CMOS forstage1 and dynamic CMOS for stage2, which is a choice for multiple-clock circuit, are reduced by 43.1% and 49%, respectively. Comparing with timing optimization single-clock CMOS, the power consumption of full-time

mixed-dynamic-static CMOS, which is chosen for power optimization of single-clock CMOS, is decreased by 3.3% without influence on clock pulse.

Table 1.2 64-bit binary comparator

Platform	Circuit style	Delay (ps)	Clock pulse (ps)	Power (mW)
	Half-time dynamic CMOS	738.5	740	13.21
Single-clock (timing optimization)	Full-time dynamic CMOS	377	450	21.83
Single-clock (power optimization)	Full-time mixed-dynamic-static CMOS	440	450	21.1
Multiple clock	Full-time static stage1 and dynamic stage2	377	690	7.51

Notation: Delay: worst-case delay

Clock pulse: the minimum clock pulse under which the CMOS can operate correctly

Power: average power in worst-case operation

1.3 Thesis organization

The thesis is organized as follows. Chapter 1 introduces background and motivation of timing and power optimization for mixed-dynamic-static CMOS. A 16-bit carry-select adder (CSA) and a 64-bit binary comparator are used as two examples for timing and power optimization. They are presented in Chapter 2 and 3 respectively. Chapter 4 summarizes design optimization and experimental results of the two example circuits.

2 TIMING AND POWER OPTIMIZATION FOR A 16-BIT CARRYSELECT ADDER

2.1 Introduction

2.1.1 Introduction of a conventional 16-bit carry select adder

In electrical products carry select adders (CSA's) are fast adders to implement summation of two binary numbers. In general, the structure of CSA consists of two main blocks, ripple carry adder(RCA) and multiplexer (mux). Two n-bit binary numbers are added by two RCAs with carry-in of 0 and 1, respectively. Then mux selects output of RCA with carry-in equals logic 0 if Cin=0; otherwise, the output of RCA with carry-in of 1 is selected.

In order to have better timing management to decrease the delay of CSA, adders with variable sizes are designed so as to have every input of mux arrives almost at the same time as the outputs of RCAs arrive for every stage. For instance, as shown in Fig. 2.1, a 16-bit CSA is comprised of four groups of adder in size of 2, 3, 4, and 5-bit.The detail of this implementation will be discussed in 2.3.2.

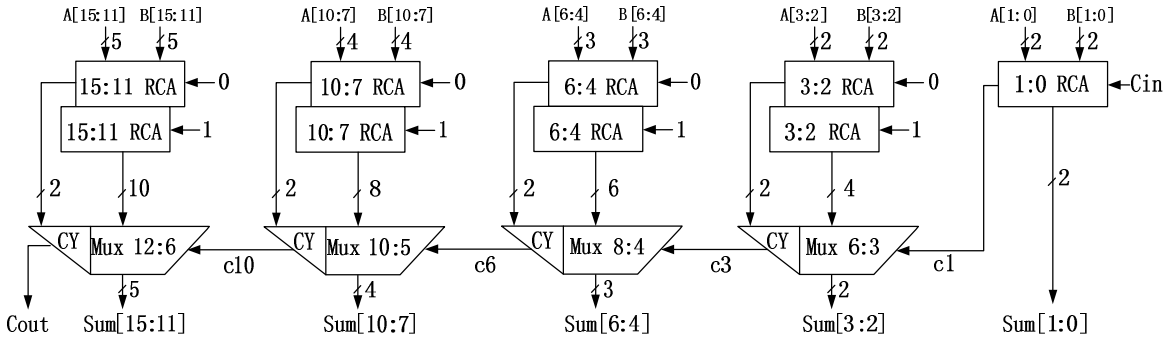


Fig. 2.1 Block diagram of 16-bit CSA

One of the available ways to decrease area and power consumption of CSA is reducing the number of gates. Binary to Excess-1 Converter (BEC) is a component that can replace RCA, and has fewer gates than RCA. BEC obtains the output of RCA with carry-in of 0, and indicates the same output as RCA with carry-in of 1. A 3-bit BEC is an example to show how to design BEC and what is the advantage of BEC compared with conventional RCA in the follows. The truth table of 3-bit BEC is shown in Table 2.1, in which B[2:0] is the 3-bit binary input (the output of 2-bit RCA with carry-in equals 0), and X[2:0] is the 3-bit binary output (the output of 2-bit RCA with carry-in equals 1).

Table 2.1 Truth table of 3-bit BEC

B[2:0]	X[2:0]
000	001
001	010
010	011
011	100
100	101
101	110
110	111
111	000

From Table 2.1, the three canonical minterm equations for each output are simplified down to

$$\begin{cases} X[0] = \overline{B[0]} \\ X[1] = B[0] \oplus B1 \\ X[2] = (B[0] \times B[1]) \oplus B[2] \end{cases} \quad (2.1)$$

According to equation (2.1), the static 3-bit BEC is depicted in Fig. 2.2 (b).It is comprised of 4 gates, less than 10 gates in conventional 2-bit adder as shown in Fig. 2.2 (a).

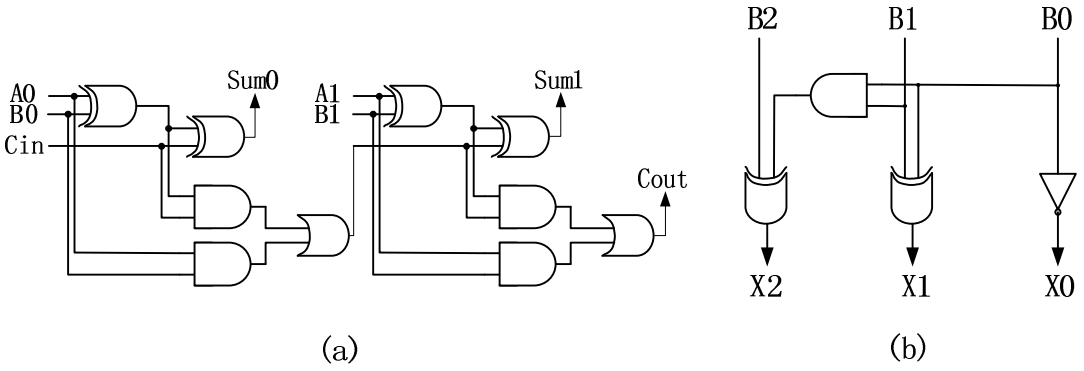


Fig. 2.2 Structure of static 2-bit adder and static 3-bit BEC. (a) 2-bit adder, (b) 3-bit BEC.

The interface of the 3-bit BEC and the 6:3 Mux is shown in Fig. 2.3. The mux selects the value of B[2:0] as output if Cin=0; otherwise the output of 3-bit BEC is selected.

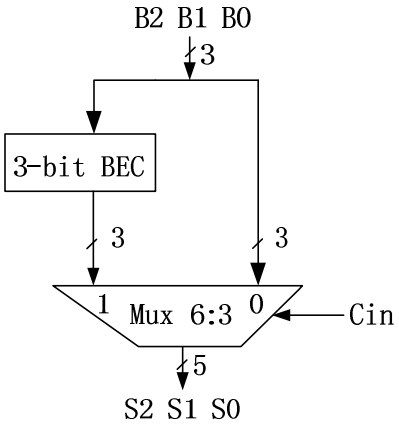


Fig. 2.3 Interface of 3-bit BEC and 6:3 mux

After the parallel RCA with Cin=1 is replaced with BEC the area and power consumption of the 16-bit CSA is reduced by 15% and 10.56%, respectively. The modified 16-bit CSA is shown in Fig. 2.4. [3]

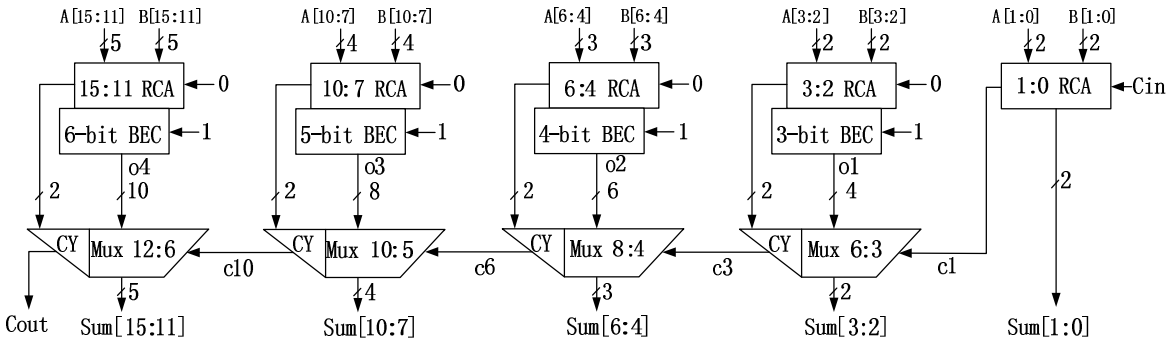


Fig. 2.4 Block diagram of the modified 16-bit CSA

2.1.2 Introduction of timing and power optimization for 16-bit CSA

In order to maintain the merit of high speed, all the blocks in Fig.2.4 are implemented by conventional (half-time) dynamic CMOS to be a reference circuit to test new designs. According to the theory proposed in chapter 1, for single-clock dynamic CMOS, delay of timing optimized 16-bit CSA is decreased by 41.1%;for multiple-clock dynamic CMOS, power and delay of optimized 16-bit CSA are decreased by 43% and 41.1%, respectively.

2.2 Design of blocks in 16-bit CSA

All the detail of designing dynamic and static blocks in Fig. 2.4, and their performance are discussed in this section.

2.2.1 Ripple Carry Adder (RCA)

For RCA, mirror adder and Manchester Carry Chain (MCC) are used for static CMOS adder and dynamic CMOS adder, respectively. The truth table of 1-bit full adder is drawn in Table 2.2, in which A, B, and Cin are three 1-bit binary inputs; Sum is the low bit of their sum and Cout is the high bit of their sum.

Table 2.2 Truth table of 1-bit full adder

Input			Output	
A	B	Cin	Cout	Sum
0	0	0	0	0
0	0	1	0	1

0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

From Table 2.2, the two equations for each output are simplified down to

$$\begin{cases} \text{Sum} = \text{A} \oplus \text{B} \oplus \text{Cin} \\ \text{Cout} = \text{A} \times \text{B} + \text{A} \times \text{C} + \text{B} \times \text{C} \end{cases} \quad (2.2)$$

According to equation (2.2), 1-bit mirror adder is drawn in Fig. 2.5 (a) [4], and that can be duplicated to implemented (n+1)-bit mirror adder, shown in Fig. 2.5 (b).

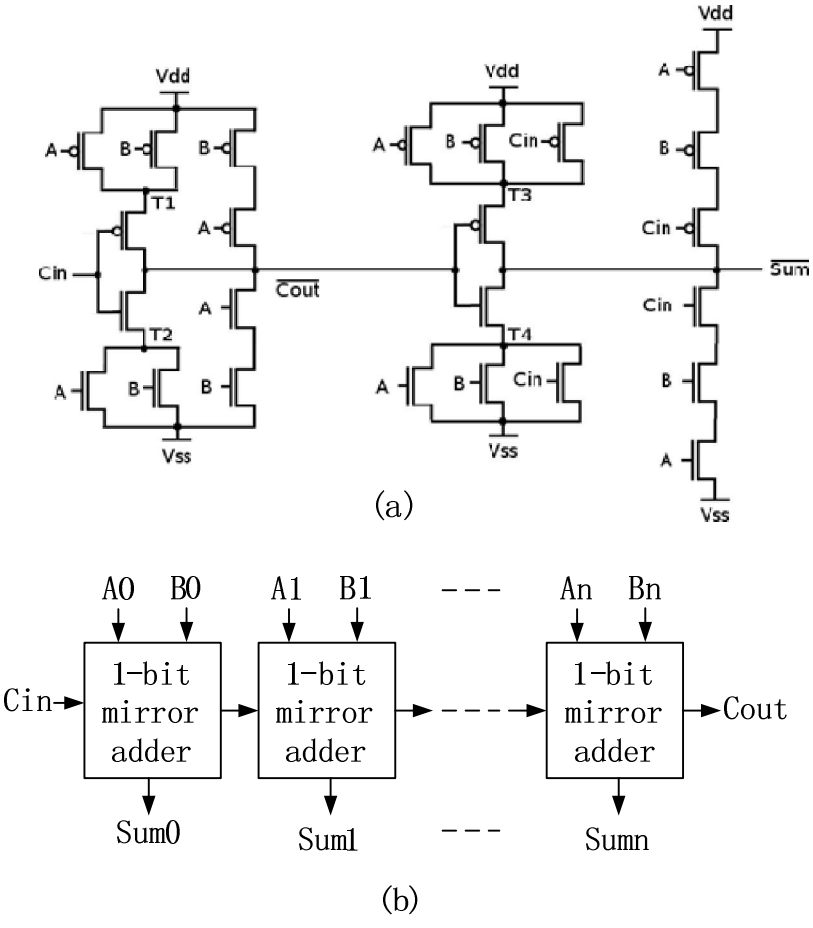


Fig. 2.5 Structure of mirror adder. (a) 1-bit mirror adder, (b) block diagram of (n+1)-bit mirror adder

Based on the equation (2.2), MCC can be drawn in Fig. 2.6, in which Cn is carry

bit of the sum of first (n-1) bit. [5]

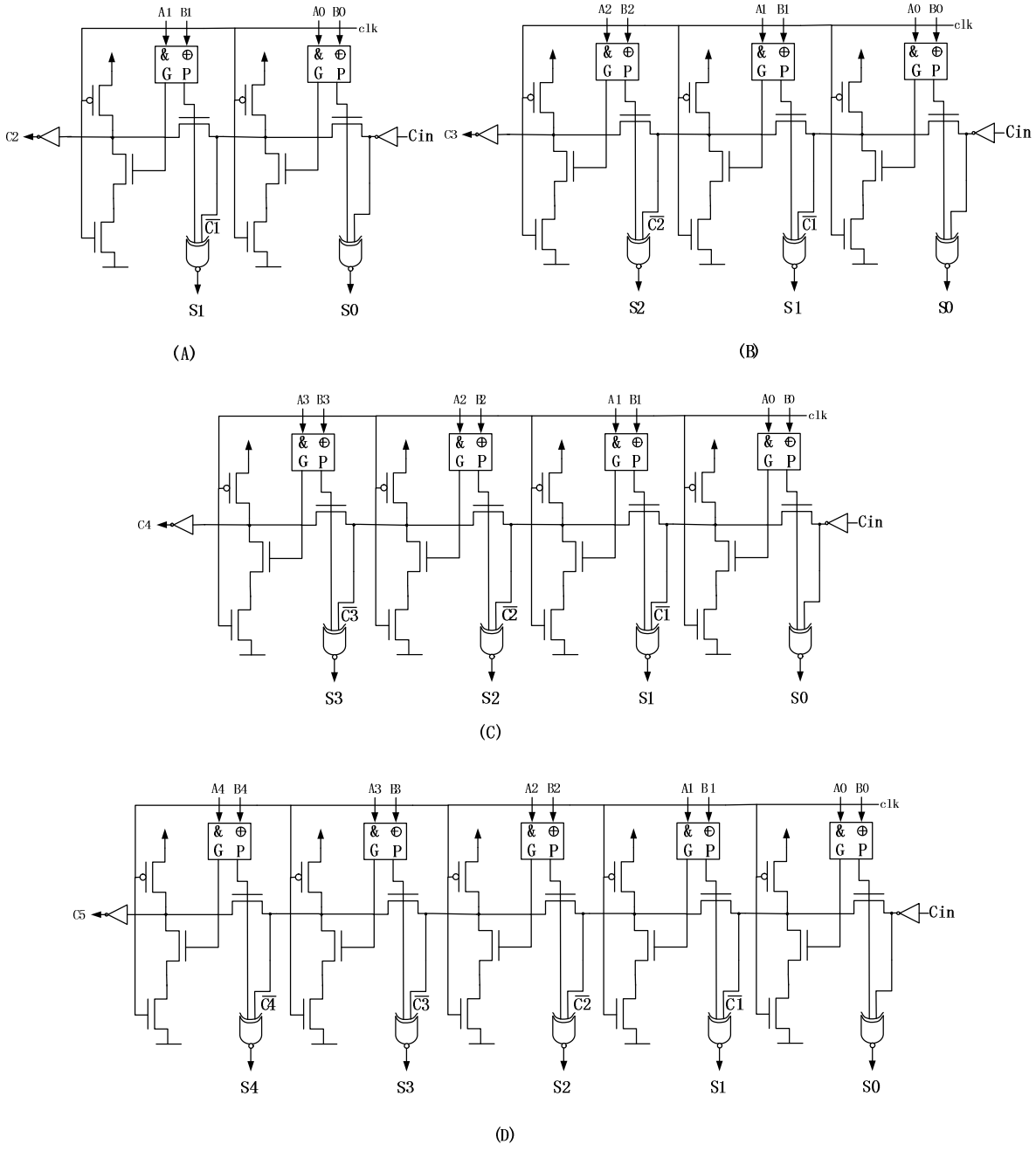


Fig. 2.6 Structure of Manchester Carry Chain (MCC). (A)2-bit MCC, (B)3-bit MCC, (C)4-bit MCC, (D)5-bit MCC.

The timing and power analysis of dynamic adder and static adder are shown in Table 2.3, in which n=1, 2, 3, 4 for 2-, 3-, 4-, 5-bit adder, respectively;$B_0 \rightarrow S_n$ means signal propagation from B_0 to S_n; “Delay” is the worst delay of circuit; “Power” is

the average power consumption of circuit operated in worst case. According to the Table 2.3, we can easily come to the conclusion that dynamic n-bit adder is faster and power-hungrier than static n-bit adder; for the same propagation in either dynamic adder or static adder, the difference of delays of n-bit adder and that of (n+1)-bit adder is pretty close.

Table 2.3 Timing and power of mirror adder and Manchester Carry Chain

			2-bit adder	3-bit adder	4-bit adder	5-bit adder
$B_0 \rightarrow S_n$	Dynamic CMOS	Delay (ps)	248	375	506	646
		Power (μW)	577.3	659.6	674.4	663.8
	Static CMOS	Delay (ps)	330	542	754	974
		Power (μW)	259.52	372.4	491	527.9
$B_0 \rightarrow C_{out}$	Dynamic CMOS	Delay (ps)	226	382.3	521	672
		Power (μW)	745.1	724.8	709.9	714.4
	Static CMOS	Delay (ps)	353	565	777	998
		Power (μW)	343.1	434.4	549.2	578
$C_{in} \rightarrow S_n$	Dynamic CMOS	Delay (ps)	209	353	522	710
		Power (μW)	473.2	516	583.1	630
	Static CMOS	Delay (ps)	319	531	744	955
		Power (μW)	257	370	489	530.7
$C_{in} \rightarrow C_{out}$	Dynamic CMOS	Delay (ps)	200	373	553	755
		Power (μW)	592.4	632.9	643.4	637.3
	Static CMOS	Delay (ps)	342	555	767	978
		Power (μW)	335.5	432.3	548	580.5

2.2.2 Binary to Excess-1 Converter (BEC)

BEC is applied to replace RCA with carry-in equals one for reducing the number of gates in CSA in order to decrease the area and power consumption. Fig. 2.7 is the structure of static BEC. Its output value is increment by 1 to its input value.

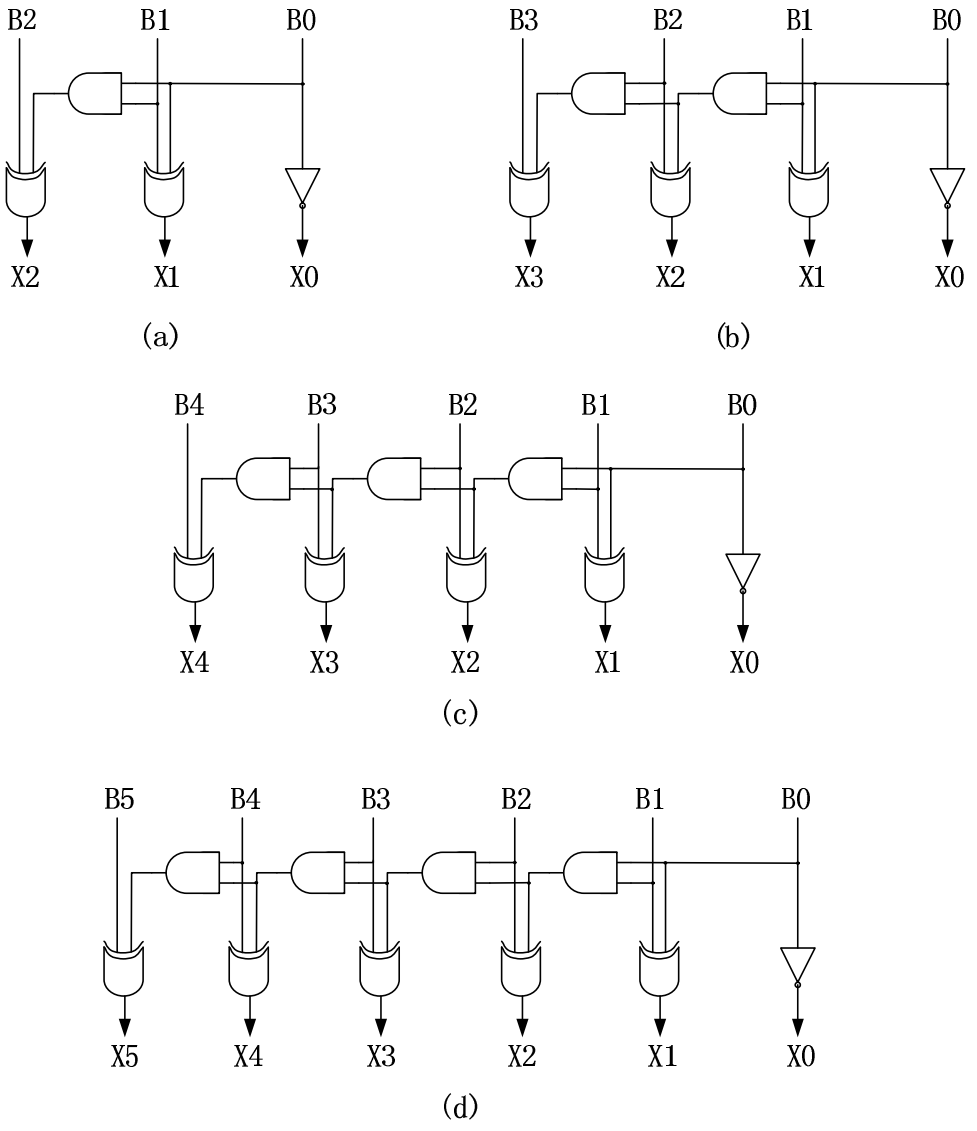
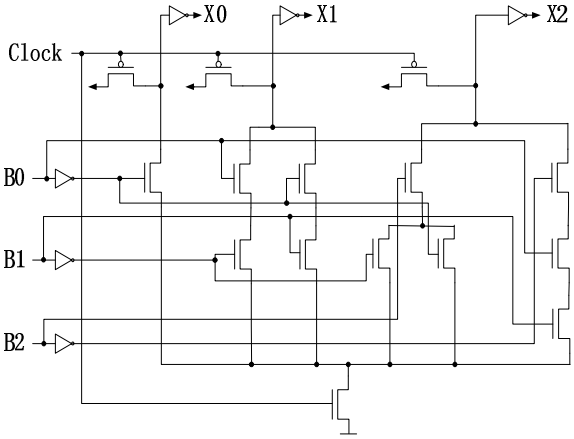
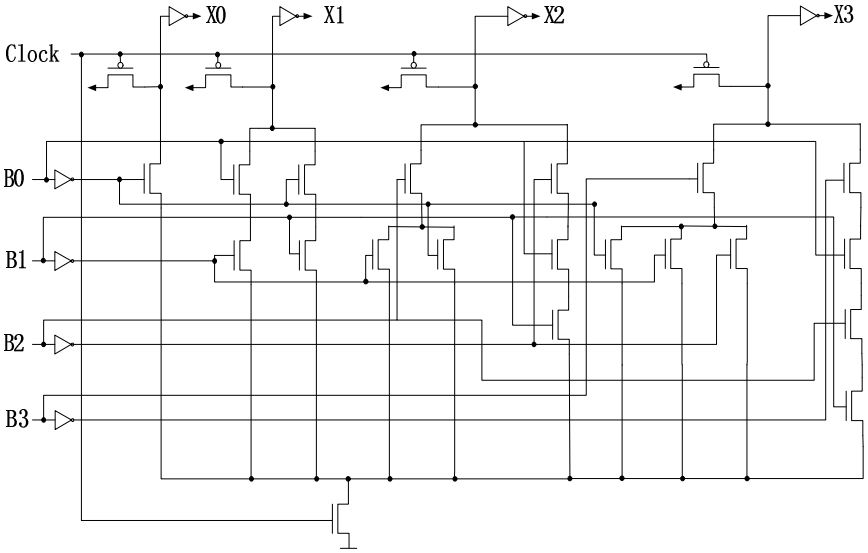


Fig. 2.7 Structure of static BEC. (a)3-bit BEC, (b)4-bit BEC, (c)5-bit BEC, (d)6-bit BEC

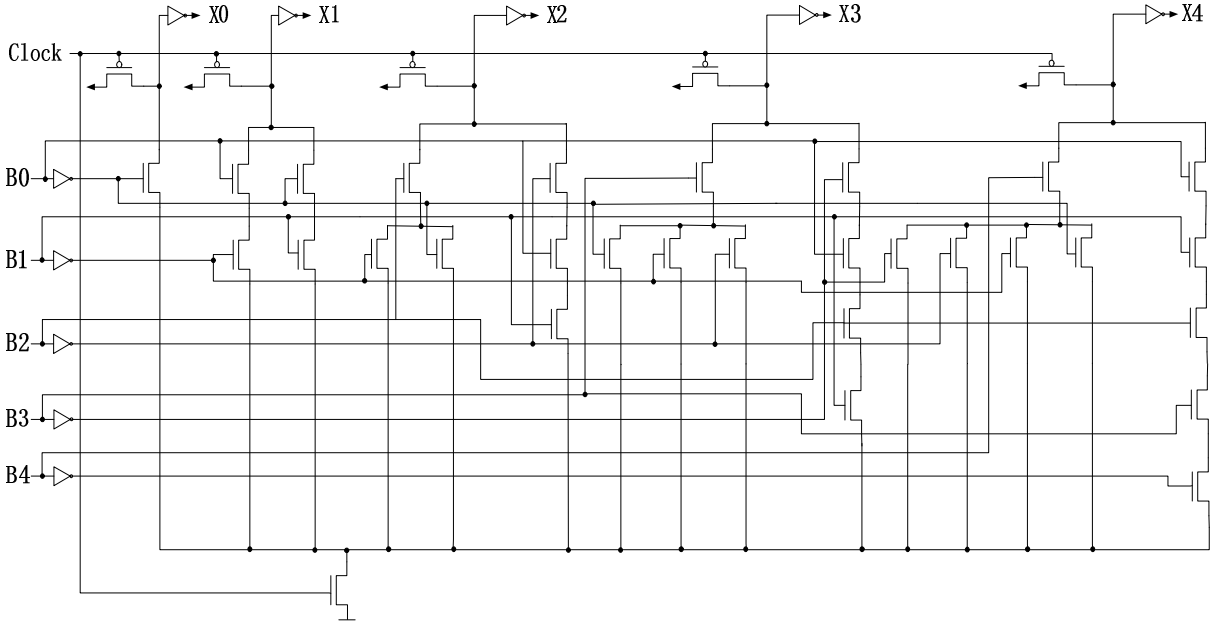
According to the function of BEC, a conventional dynamic BEC is depicted in Fig. 2.8. Base on the simulation result of all dynamic and static BEC using Cadence Spectre, the performance of timing and power of all BEC are presented in Table 2.4,in which delay and power are measured for signal propagation form B0 to Xn in n-bit BEC. In comparison with the static BEC, the dynamic BEC is roughly 30-50% faster but consume around 30% extra power. For both dynamic and static BEC delay and power consumption of n-bit BEC are all increased when n is increased.


X0
X1
X2
Clock
B0
B1
B2

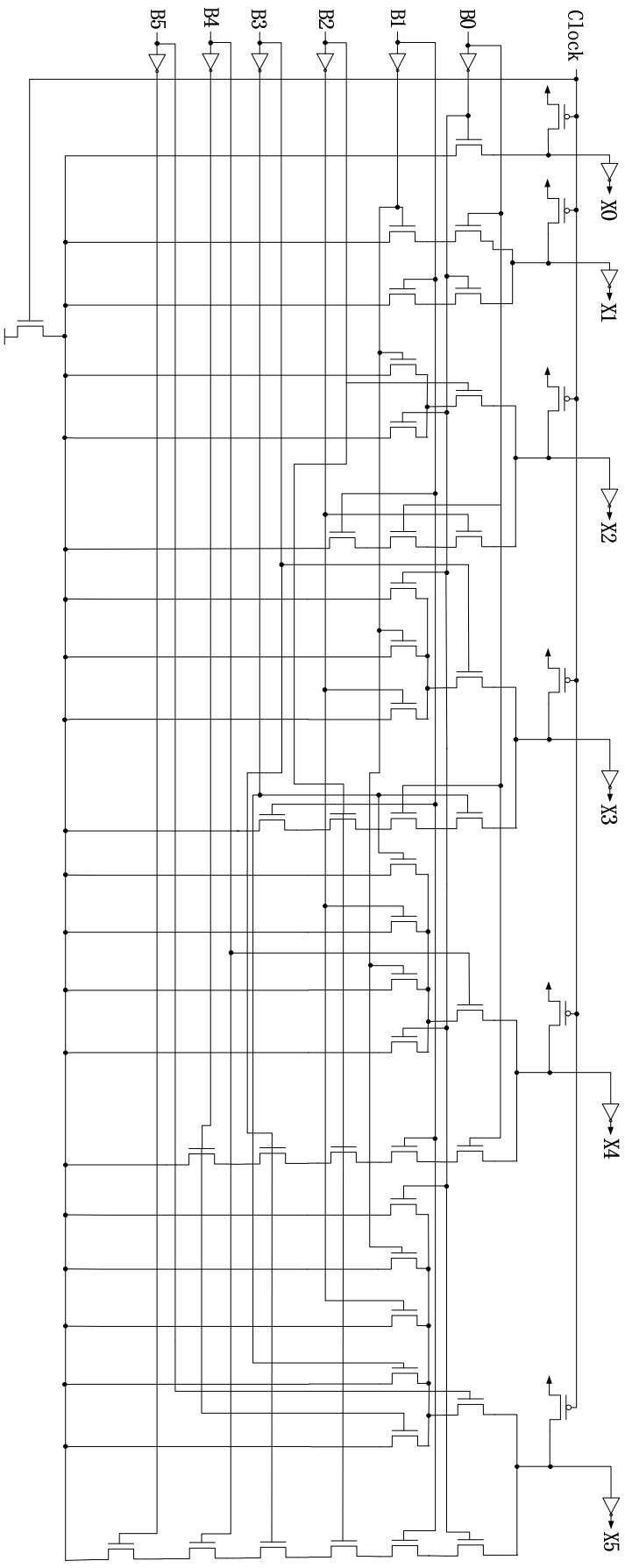
(a)


X0
X1
X2
X3
Clock
B0
B1
B2
B3

(b)


X0
X1
X2
X3
X4
Clock
B0
B1
B2
B3
B4

(c)



(d)

Fig. 2.8 Structure of dynamic BEC. (a)3-bit BEC, (b)4-bit BEC, (c)5-bit BEC, (d)6-bit BEC

Table 2.4 Timing and power of static and dynamic BEC

		3-bit BEC	4-bit BEC	5-bit BEC	6-bit BEC
Static CMOS	Delay (ps)	288	385	484	592
	Power (μW)	355	474.1	512.7	572.9
Dynamic CMOS	Delay (ps)	200	239	265	278
	Power (μW)	457.3	586.3	667.8	732.2

2.2.3 Multiplexer (mux)

The 16-bit CSA utilizes mux controlled by carry-in to select value from two vector-inputs. All the mux, 6:3, 8:4, 10:5, and 12:6 mux, we need for 16-bit CSA are composed by several 2:1 mux, which is controlled by select-signal to choice one of the two inputs as output. Table 2.5 is the truth table of 2:1 mux.

Table 2.5 Truth table of 2:1 mux

Input			Output
S (Select-signal)	Input 0	Input 1	
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

From Table 2.5, the equation for the output of 2:1 mux is simplified down to

$$\text{Output} = \text{S} \times \text{Input } 1 + \bar{\text{S}} \times \text{Input } 0 \qquad (2.3)$$

According to equation (2.3), the static 2:1 mux is depicted in Fig. 2.9 (a), in which when S equals 0, NMOS M1 is on and NMOS M2 is off, thereafter the value of “input0” is connected to the output; otherwise, NMOS M1 is off and NMOS M2 is on, then the value of “input1” is transferred to output. Buffer is used before the output of static 2:1 mux to drive logic 1 to sufficient voltage due to NMOS is bad at conducting high voltage (logic 1). For dynamic 2:1 mux, whose structure is depicted in Fig. 2.9

(b), in precharge phase (clock is 0) the PMOS M3 is on and the NMOS M8 is off, then the signal ‘a’ is pulled up to high voltage and output is pulled down to low voltage. Afterwards, in evaluate phase (clock is 1) the PMOS M3 is off and the NMOS M8 is on. When S=0, the NMOS M4 is on and the NMOS M5 is off, and the output is pulled up to ‘1’ if input0 equals ‘1’; otherwise the output stays at‘0’.The output is assigned by the value of “input 0”. When S=1, the NMOS M4 is off and the NMOS M5 is on, then output is pulled up to 1 if “input1” is‘1’; otherwise output stays at logic‘0’. The output is assigned by the value of “input 1”.

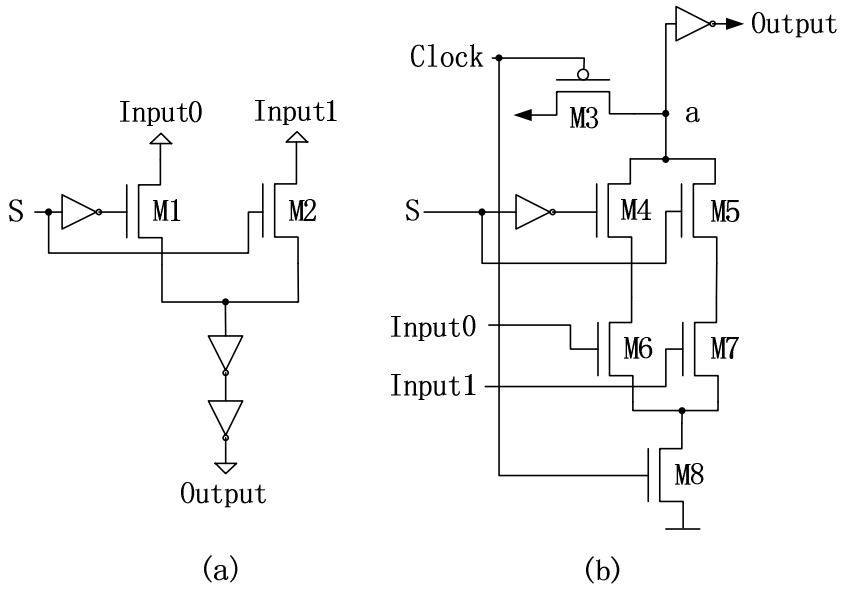


Fig. 2.9 Structure of 2:1 mux. (a)static 2:1 mux, (b)dynamic 2:1 mux.

A 2n:n mux, shown in Fig. 2.10, is a combination-circuit of n 2:1 mux controlled by the same select-signal. So the delay of 2n:n mux should equal to that of 2:1 mux. But as we can see from Table 2.6, in which the timing and power consumption are measured when input x_0 keeps 0, input x_1 keeps 1 (x=1, 2, …, n), and S changes from 0 to 1, the delay of 2n:n mux increases as n increases. The reason is the arriving time of S is extended with the increase of fan-out of S when n rises. To prove the theory, I duplicate single S in 12:6 static mux to make sure the fanout of every S in

static 6:3 mux and that in static 12:6 mux are exactly equal. Then their delays (250ns) are perfectly equal. For both dynamic and static mux the power of 2n:n mux is approximately n times of that of 2:1 mux, which is 282μW for static 2:1 mux and 141μW for dynamic 2:1 mux.

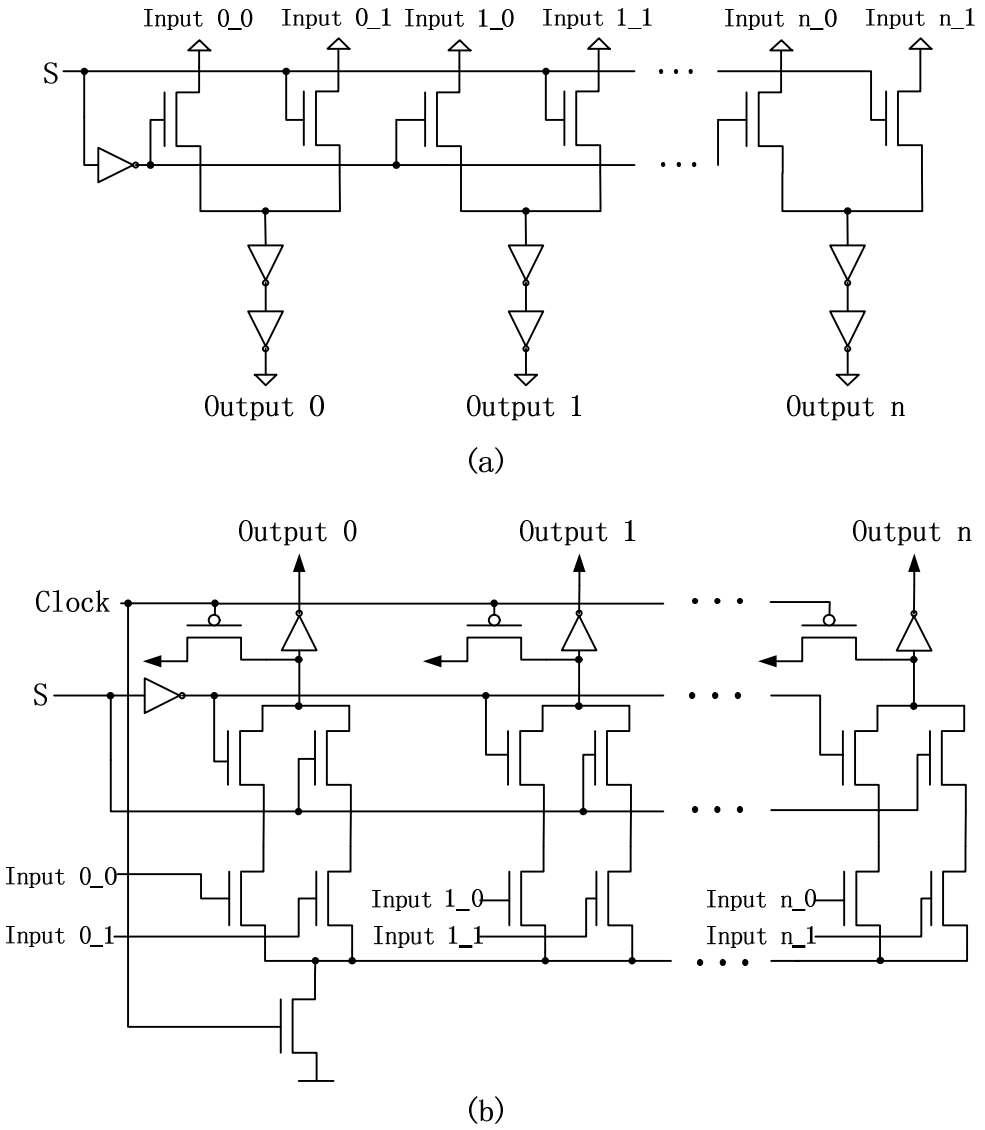


Fig. 2.10 Structure of 2n:n mux. (a)static 2n:n mux, (b)dynamic 2n:n mux.

Table 2.6 Timing and power of dynamic and static 6:3, 8:4, 10:5, and 12:6 mux

		6:3 mux	8:4 mux	10:5 mux	12:6mux
Static CMOS	Delay (ps)	250	254	259	265
	Power (μW)	859.1	1.132	1.395	1.677
Dynamic CMOS	Delay (ps)	138	145	150	153
	Power (μW)	423.4	581	695	844.6

2.3 Timing and Power optimization for mixed-dynamic-static16-bit CSA

As shown in Fig. 2.4 the 16-bit CSA has three operation stages. They are RCA, BEC, and mux. We will discuss in this section about choosing static CMOS and dynamic CMOS for the three stages for timing and power optimization.

2.3.1 Partition in 16-bit CSA

As mentioned in chapter one, full-time dynamic CMOS is the fastest conventional design for defined function. So as the requirement of structure of full-time dynamic CMOS, the three stages in 16-bit CSA should be separated to two groups which will evaluate in evaluate-section and precharge-section, respectively.

As seen in Fig.2.4, RCA and BEC are parallel connected. The structure of connection of 5-bit RCA and 6-bit BEC, shown in Fig. 2.11, will be an example to explain the connection between RCA and BEC and how it affects the final choice of partition of stages. In Fig. 2.11, each level of 6-bit BEC works immediately after the same stage of 5-bit CSA, in other words, the circuit works vertically parallel. To prove the assumption, the circuit is tested with two continuous vector, A[4:0]=0000, B[4:0]=0111 and A[4:0]=0001, B[4:0]=0111, in which case, signal propagates from A0 to X5, and both S4 and X5 change from 0 to 1. The signal-arriving time of S4 and X5 are 557ps and 624ps, separately. The difference between the two times is only 67ps that is much less than the operation time of the whole 6-bit BEC (592ps), so RCA and BEC are really parallel operating. Based on the relationship between RCA and BEC, it is better to group them together to operate under the same clock pulse to avoid wasting time.

Therefore, the method to design fastest dynamic 16-bit CSA, shown in Fig. 2.4, is that RCA-stage and BEC-stage evaluate together in one clock pulse, and mux-stage evaluates in the following clock pulse.

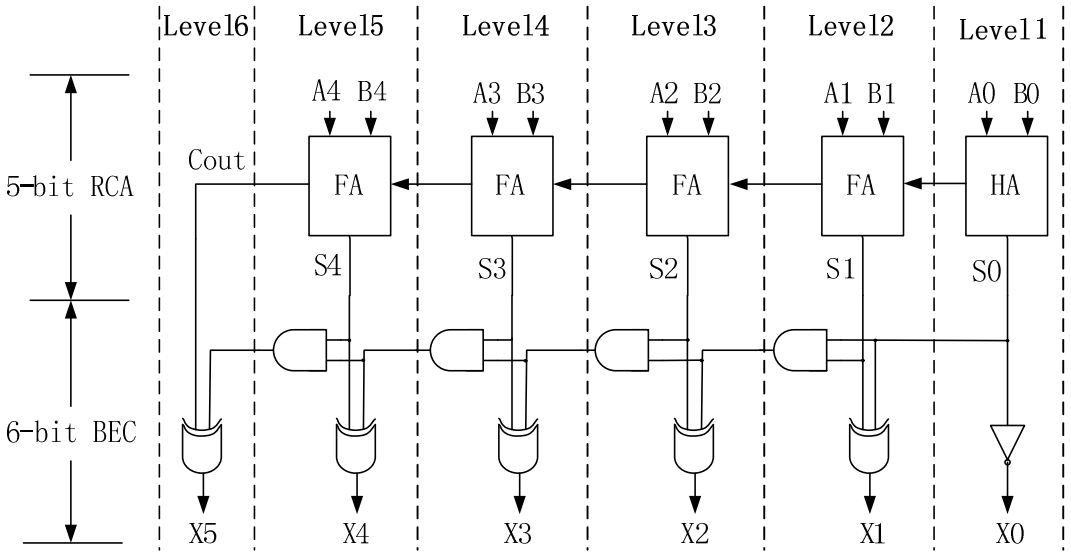


Fig. 2.11 Connection of 5-bit RCA and 6-bit BEC

Because of the operation feature of dynamic CMOS, output of RCA and BEC are all 0 in precharge-section that is unavailable for next stage, mux-stage. So CMOS switch, shown in Fig. 2.12, should be used before mux-stage in order to hold the output value of RCA and BEC for mux during the precharge-section. The value of input propagates to output directly if clock=1, NMOS M1 and PMOS M2 are on; otherwise M1 and M2 are off, and output keeps the former value of input until clock=1 again. So CMOS switch controlled by the same clock with stage 1 and stage 2 can be inserted after stage 2 to hold value for stage 3 during precharge-section.

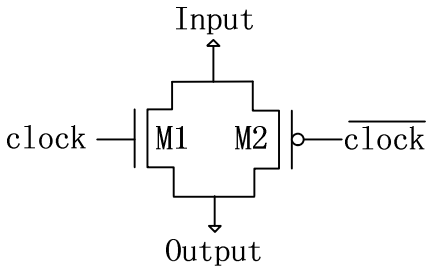


Fig. 2.12 Structure of CMOS switch

2.3.2 Modification of full-time dynamic 16-bit CSA

The original intention of utilizing RCA with different sizes in 16-bit CSA is to

adjust all the input signals of each mux arrive at the same time at far as possible. To prove the theory, all the blocks in CSA are assumed to be dynamic CMOS, and the data in Table 2.2, Table 2.3, and Table 2.5 is used to estimate the arriving time of input signals of mux, shown in Fig. 2.13. As RCA and BEC operate parallel, 70ps, which is approximately the delay of BEC after RCA work out, will be used for operation time of BEC in CSA.

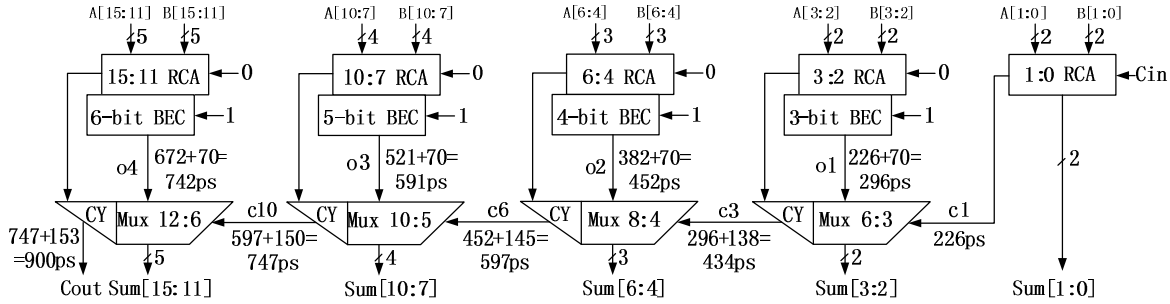


Fig. 2.13 Timing analysis of input signals of mux in 16-bit CSA

The arriving times of vertical and horizontal input signals of mux in Fig. 2.13 are recorded in Table 2.7, from which we can see the arriving times of all inputs of each mux are almost equal. So the theory mentioned above is verified.

Table 2.7 Arriving times of input signals of mux in half-time dynamic 16-bit CSA

Component	Input signal	Arriving time (ps)	Difference between arriving times (ps)
mux 6:3	c1	226	70
	o1	296	
mux 8:4	c3	434	18
	o2	452	
mux 10:5	c6	597	6
	o3	591	
mux 12:6	c10	747	5
	o4	742	

However, if mux-stage is separated to operate in another clock-pulse, c1 and o1 will arrive much earlier than o4 in Fig. 2.13 and have to wait for o4 that is waste of time what we do not want to see. In order to operate efficiently, c1, o1, o2, and o4 should be obtained at the same time as far as possible, then the next stage can start to

evaluate in next clock pulse immediately. So it is better to choose all RCA with the same size, 4-bit RCA. According to the data I measured, the timing of propagation of signal from Cin to Sum15 (635ps) and that from B12 to o1 (660ps) in Fig. 2.14, the structure of modified 16-bit CSA, are almost equal, so the circuit belongs to situation 1 in chapter 1.

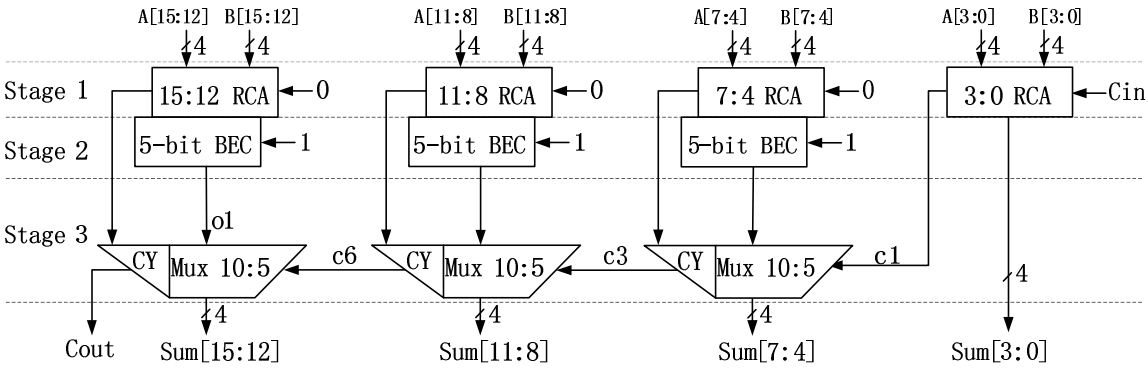


Fig. 2.14 Block diagram of 16-bit CSA consists of RCA with same size

The power and timing analysis of conventional 16-bit CSA and modified 16-bit CSA are shown in Table 2.8, in which half-time dynamic CMOS is faster and power-hungrier than static CMOS, but slower and more power-economical than full-time dynamic CMOS; for full-time dynamic CMOS, the fastest design, the worst delay of modified 16-bit CSA (789ps) is 26.9% less than that of conventional 16-bit CSA (1080ps); for static circuit, minimum input pulse is even less than the delay of static CMOS, and the reason is that value of output is not changed immediately after new input arrives, but can be kept until next value propagates to output.

Table 2.8 Power and timing analysis of conventional 16-bit CSA and modified 16-bit CSA

Signal propagation	Circuit	Circuit type	Delay (ps)	Power (mW)	Clock/input pulse (ps)
$B_{11} \rightarrow S_{15}$	Conventional	Static CMOS	1,110	2.101	880

	16-bit CSA	Half-time dynamic CMOS	810	6.428	810
		Full-time dynamic CMOS	390	10.39	650
$B_{12} \rightarrow S_{15}$	Modified 16-bit CSA	Static CMOS	1,020	1.901	850
		Half-time dynamic CMOS	682	4.187	690
		Full-time dynamic CMOS	235	9.878	550
$B_{11} \rightarrow Cout$	Conventional 16-bit CSA	Static CMOS	1,080	2.751	650
		Half-time dynamic CMOS	829	6.483	830
		Full-time dynamic CMOS	546	10.16	690
$B_{12} \rightarrow Cout$	Modified 16-bit CSA	Static CMOS	979	2.437	650
		Half-time dynamic CMOS	697	6.822	700
		Full-time dynamic CMOS	234	9.866	580
$Cin \rightarrow S_{15}$	Conventional 16-bit CSA	Static CMOS	1,820	2.39	740
		Half-time dynamic CMOS	1,220	4.501	1,240
		Full-time dynamic CMOS	1,080	6.935	1,070
	Modified 16-bit CSA	Static CMOS	1,960	2.156	600
		Half-time dynamic CMOS	1,340	3.982	1,340
		Full-time dynamic CMOS	789	7.889	790
$Cin \rightarrow Cout$	Conventional 16-bit CSA	Static CMOS	1,820	2.499	740
		Half-time dynamic CMOS	1,220	4.711	1,240
		Full-time dynamic CMOS	1,080	7.222	1,070
	Modified 16-bit CSA	Static CMOS	1,960	2.265	600
		Half-time dynamic CMOS	1,340	4.187	1,340
		Full-time dynamic CMOS	789	8.216	790

Notation: Delay: the timing of corresponding signal propagation

Power: the average power consumption of corresponding operation

Clock/input pulse: for static COMS, it is minimum input pulse under what CMOS can operate correctly; for dynamic COMS, it is minimum clock pulse

under what CMOS can operate correctly

2.3.3 Timing and Power optimization for 16-bit CSA

Timing and power optimization for 16-bit CSA of single-clock and multiple-clock will be discussed in this section.

The analysis of delay, power, and minimum clock pulse for modified 16-bit CSA is shown in Table 2.9, in which DDS means the first and second stages in CSA are dynamic CMOS and the third one in CSA is static CMOS; SSD means the first and second stages in CSA are static CMOS and the third one in CSA is dynamic CMOS. For multiple-clock dynamic CMOS, several same circuits operate parallel as pipeline, shown in Fig 1.3, so the delay for data use plays more important role than its clock pulse. Afterwards the best platform for full-time dynamic SSD CMOS is multiple-clock CMOS, because the original idea of designing SSD is keep delay, and sacrifice clock pulse to achieve low power; half-time dynamic CMOS and full-time dynamic DDS CMOS should be operated in single-clock CMOS, because they do not trade clock pulse to any benefit. In Table 2.9, timing of signal propagation of $\mathrm{Cin} \rightarrow \mathrm{S}_{15}$ and $\mathrm{Cin} \rightarrow \mathrm{Cout}$, utmost delay, are pretty close, but power consumption of signal propagation of $\mathrm{Cin} \rightarrow \mathrm{Cout}$ is greater than that of $\mathrm{Cin} \rightarrow \mathrm{S}_{15}$. So I will define $\mathrm{Cin} \rightarrow \mathrm{Cout}$ as worst case to analysis the performance of different CMOS.

Table 2.9 Power and timing analysis of modified 16-bit CSA

Signal propagation	Circuit type	Delay (ps)	Clock pulse (ps)	Power (mW)
$\mathrm{B}_{12} \rightarrow \mathrm{S}_{15}$	Half-time dynamic CMOS	682	690	4.187
	Full-time dynamic CMOS	235	550	9.878
	Full-time dynamic CMOS (DDS)	233	630	8.934
	Full-time dynamic CMOS (SSD)	285	1,320	2.082
$\mathrm{B}_{12} \rightarrow \mathrm{Cout}$	Half-time dynamic CMOS	697	700	6.822
	Full-time dynamic CMOS	234	580	9.866
	Full-time dynamic CMOS	233	660	9.017

	(DDS)			
	Full-time dynamic CMOS (SSD)	288	1,080	2.55
$Cin \rightarrow S_{15}$	Half-time dynamic CMOS	1,340	1,340	3.982
	Full-time dynamic CMOS	789	790	7.889
	Full-time dynamic CMOS (DDS)	835	830	7.557
	Full-time dynamic CMOS (SSD)	784	1,210	2.333
$Cin \rightarrow Cout$	Half-time dynamic CMOS	1,340	1,340	4.187
	Full-time dynamic CMOS	789	790	8.216
	Full-time dynamic CMOS (DDS)	831	830	7.988
	Full-time dynamic CMOS (SSD)	788	1,210	2.388

Notation: Delay: the timing of corresponding signal propagation

Power: the average power consumption of corresponding operation

Clock pulse: minimum input pulse under what CMOS can operate correctly

For single-clock dynamic CMOS, clock pulse cannot be sacrificed, and no extra timing can be utilized during precharge- and evaluate-section, so no resource can be used to do power optimization; timing optimization is the only choice, and full-time dynamic CMOS should be chosen. As we can see Table 2.9, the power consumption of full-time dynamic CMOS (8.216mW) is almost two times of that of half-time dynamic CMOS (4.187mW), and 2.9% greater than that of full-time dynamic DDS CMOS, but the delay of full-time dynamic CMOS (789ps) is 41.1% and 5.1% less than that of dynamic CMOS (1,340ps) and that of full-time dynamic DDS CMOS (831ps), respectively.

For multiple-clock circuit, the delay for data use plays more important role than its clock pulse, so clock pulse can be sacrificed to decrease power-consumption with keeping delay that can be implemented by replacing stage 1 and stage 2 with static or mixed-dynamic-static CMOS. So even minimum clock pulse of full-time dynamic SSD CMOS is about 50% greater that of other full-time dynamic COMS, the best

choice for multiple-clock CMOS is still it. As shown in Table 2.8, the delay of full-time dynamic SSD CMOS (788ps) is 0.6%, 6.1%, and 41.2% less than that of full-time dynamic CMOS (789ps), full-time dynamic DDS CMOS (831ps), and half-time dynamic CMOS (1,340ps), respectively; the power consumption of full-time dynamic SSD CMOS is roughly 70% and 43% less than that of full-time dynamic CMOS and half-time dynamic CMOS, respectively.

3 TIMING AND POWER OPTIMIZATION FOR A 64-BIT BINARY COMPARATOR

3.1 Introduction

In this chapter, we will discuss the method to optimize timing and power for a 64-bit binary comparator in order to propose the optimization theory for CMOS in situation 2 mentioned in chapter 1.

3.1.1 Introduction of 64-bit binary comparator

Binary comparator is basic digital arithmetic component that operates to compare two binary numbers. A 64-bit binary comparator has two 64-bit binary input (A_{63} to A_0 & B_{63} to B_0) and three binary output, which indicates if A>B, A<B, or A=B.

In recent years, low power and high speed become the foremost parameter for designing electrical devices due to explosive demand of portable equipment that has limited battery, but needs quicker response, such as cell phone, laptop, and GPS etc. [6]

The existing design principles of 64-bit binary comparator and their performance are compared in [7], which includes:

A. Priority-Encoding-Based Comparator [8], [9]

Priority-encoding-based comparators utilize priority encoders to speed up the comparison of two binary numbers.

B. BCL-Based Comparator [10]

The two n-bit binary number inputs (A & B)of BCL-based comparator are encoded to two n-bit number (Ae & Be), in which each bit of Ae (or Be) is 0 if the same bit of A (or B) is greater than that of B (or A); otherwise it is 1. Then the 1

in Be and Ae closest to the MSB is detected and the comparison result can be determined afterwards.

C. Tree-Structure-Based Comparator

[11] and [12] introduce a method to design comparator that called tree-based comparator, in which dynamic Manchester adder is used to speed up the comparison in the longest stage in comparator.

Table 3.1 shows the performance comparison of 64-bit binary comparators mentioned above [7].

Table 3.1 Performance comparison of 64-bit comparators

Publication	Frustaci*et al.* [12]			Lam and Tsui [9]			Kim and Yoo [10]			Huang and Wang [8]		
Process (nm)	180	90	65	180	90	65	180	90	65	180	90	65
Delay (ps)	633	352	211	453	180	124	1005	386	268	752	311	212
Worst Power (μW)	1133	283	216	3102	844	608	2194	401	339	1364	307	234
Number of transistors	1365			3386			964			1640		

A fast 64-bit binary comparator is proposed and used to demonstrate our approach to timing and power optimization. The delay and power of the 64-bit comparator are 738.5ps and 13.21mW respectively, which is implemented in 250nm CMOS process. And, the number of transistors in this comparator is 1314.

3.1.2 Introduction of timing and power optimization for 64-bit binary comparator

In order to maintain the merit of high speed, all the blocks in 64-bit binary comparator are implemented by conventional (half-time) dynamic CMOS and are used as our reference circuits for future comparison. Using single clocking dynamic CMOS delay of the 64-bit binary comparator after timing optimization is reduced by 49%. In comparison with the timing optimized circuit, power after optimization is

reduced by 3.3% without changing clock pulse. Using multiple clocking dynamic CMOS power and delay after optimization is decreased by 43.1% and 49%, respectively.

3.2 Design of the 64-bit binary comparator

3.2.1 Module design of the 64-bit binary comparator

The module design of the 64-bit binary comparator is shown in Fig. 3.1. The module design implementation indicates the comparison of two 64-bit binary numbers (A_{63} to A_0&B_{63} to B_0). The three binary outputs indicate if one number is greater than, equal to, or less than another one.

In Fig. 3.1, the 32-bit binary comparator as shown in Fig. 3.2 is used to compare two 32-bit binary numbers and the output results (A>B, A=B, or A<B) are fed to the inputs of the 6-input binary comparator that processes two 32-bit binary comparator' outputs. The higher order 32-bit binary comparator result, "A>B" or "A<B", dominates the 6-input binary comparator result, "A>B" or "A<B". If the higher order 32-bit binary comparator result is "A=B" then the lower order 32-bit binary comparator result dominates the 6-input binary comparator result.

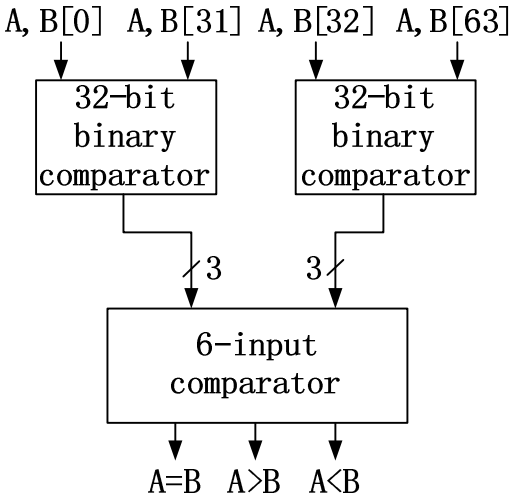


Fig. 3.1 Block diagram of 64-bit binary comparator

In Fig. 3.2, the 8-bit binary comparator as shown in Fig. 3.3 is used to compare

two 8-bit binary numbers and the output results (A>B, A=B, or A<B) are fed to the inputs of the 12-input binary comparator that processes for 8-bit binary comparator' outputs. Following the operation of 6-input comparator in Fig. 3.1 the higher order 8-bit binary comparator result, "A>B" or "A<B", dominates the 12-input binary comparator result, "A>B" or "A<B".

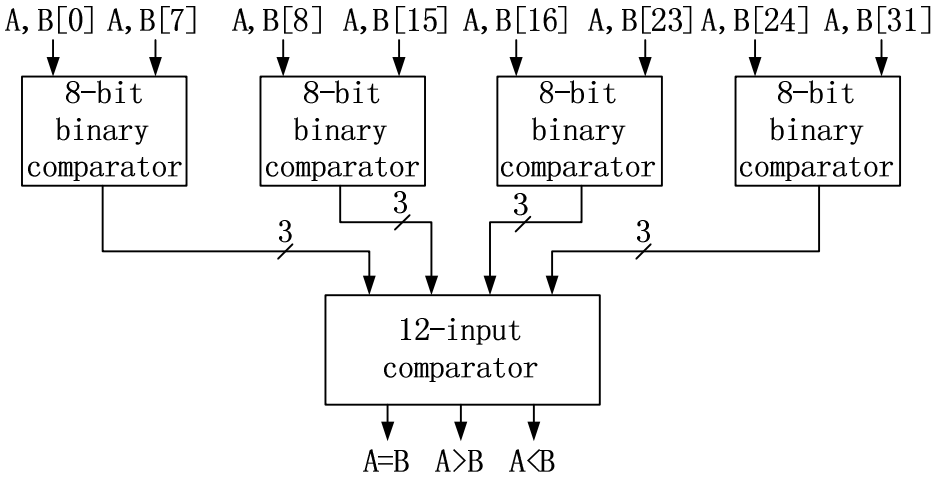


Fig. 3.2 Block diagram of 32-bit binary comparator

In Fig. 3.3, the 2-bit binary comparator compares two 2-bit binary inputs and the output results (A>B, A=B, or A<B) are fed to the inputs of the 12-input binary comparator that processes four 2-bit binary comparators' outputs. Following the operation of 12-input comparator in Fig. 3.2 the higher order 2-bit binary comparator result, "A>B" or "A<B", dominates the 12-input binary comparator result, "A>B" or "A<B".

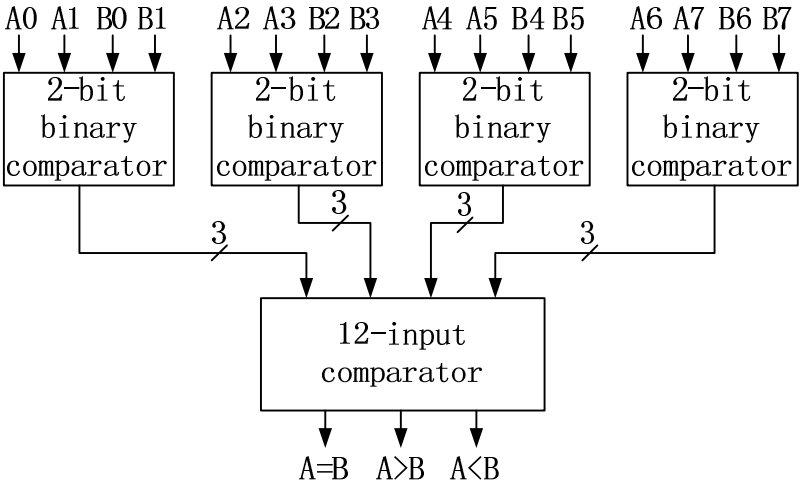


Fig. 3.3 Block diagram of 8-bit binary comparator

3.2.2 Design of blocks in the 64-bit binary comparator

The transistor level design and the analysis of timing and power of all blocks (dynamic & static 2-bit binary comparator, dynamic & static 12-input binary comparator, and dynamic & static 6-input binary comparator) we need for optimizing 64-bit binary are discussed in this section.

3.2.2.1 2-bit binary comparator

The truth table of the 2-bit binary comparator is shown in Table 3.2. It determines if one 2-bit binary number is greater than, equal to, or less than another one.

Table 3.2 Truth table of 2-bit binary comparator

Input				Output		
A1	A0	B1	B0	A>B	A=B	A<B
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1

1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

Notation:

A1 The high bit of 2-bit binary number A

A0 The low bit of 2-bit binary number A

B1 The high bit of 2-bit binary number B

B0 The low bit of 2-bit binary number B

A>B The value is logic 1 if A>B; otherwise it is logic 0

A=B The value is logic 1 if A=B; otherwise it is logic 0

A<B The value is logic 1 if A<B; otherwise it is logic 0

From Table 3.2, the three canonical minterm equations for each output are simplified down to

$$\begin{cases} \text{"A>B"} = \mathrm{A1} \cdot \overline{\mathrm{B1}} + \mathrm{A0} \cdot \overline{\mathrm{B0}} \cdot \left(\overline{\mathrm{B1}} + \mathrm{A1}\right) \\ \text{"A=B"} = \overline{(\mathrm{A1} \oplus \mathrm{B1}) + (\mathrm{A0} \oplus \mathrm{B0})} \\ \text{"A<B"} = \overline{\mathrm{A0}} \cdot \mathrm{B0} \cdot \left(\overline{\mathrm{A1}} + \mathrm{B1}\right) + \overline{\mathrm{A1}} \cdot \mathrm{B1} \end{cases} \quad (3.1)$$

Afterwards, based on Eq. (3.1) the transistor schematic of the 2-bit binary comparator is shown in Fig. 3.4.

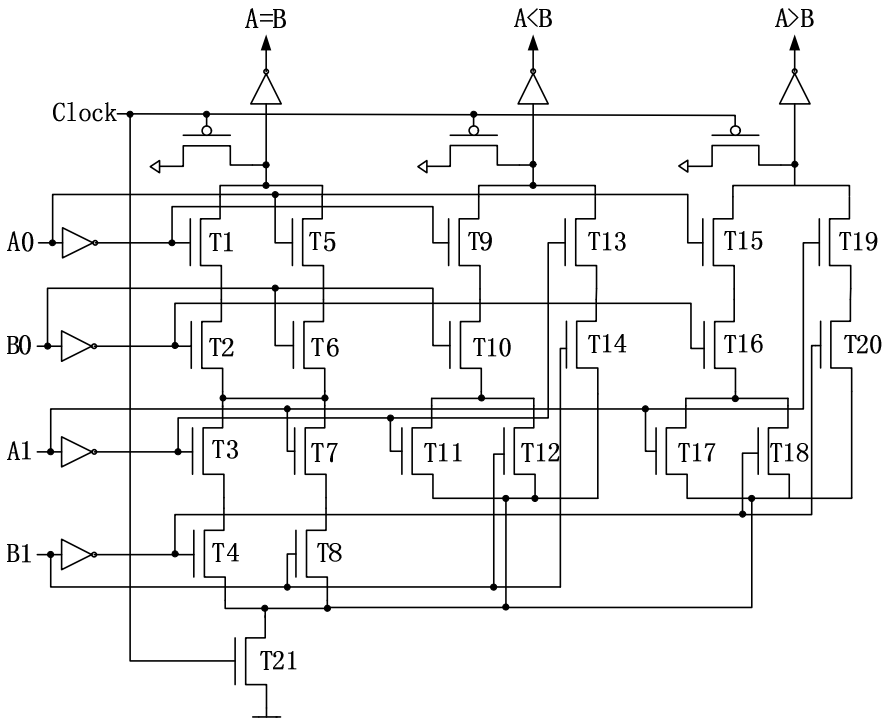


Fig. 3.4 Structure of dynamic 2-bit binary comparator

According to the transistor size optimization algorithm for dynamic CMOS logic [18], the process of optimizing transistor size for 2-bit binary comparator is discussed in the following.

A. Identify all timing paths and assign weights to each transistor

The ten timing paths of 2-bit binary comparator are shown in Table 3.3. In order to put more effort in increasing the size of transistor that appear in most timing paths and have more effect on decreasing delay of circuit, the algorithm considers the number of timing paths a transistor participates in and defines the number as *repeat* for each transistor. Because the discharging time of a transistor in a series path increases with the distance from output, the algorithm denotes *weight* (from 0.05 to 0.5) to individual transistor according to its distance from output, and *weight* 0.5 is assigned to transistors closest to the output. The *repeat* and *weight* of all transistors in Fig. 3.4 are shown in Table 3.4.

Table 3.3 Timing path in 2-bit binary comparator

Path No.	Transistors	Path No.	Transistors
1	T_1, T_2, T_3, T_4	6	T_9, T_{10}, T_{12}
2	T_1, T_2, T_7, T_8	7	T_{13}, T_{14}
3	T_5, T_6, T_3, T_4	8	T_{15}, T_{16}, T_{17}
4	T_5, T_6, T_7, T_8	9	T_{15}, T_{16}, T_{18}
5	T_9, T_{10}, T_{11}	10	T_{19}, T_{20}

Table 3.4 Repeat and weight profiles for 2-bit binary comparator

Repeats	Near GND			Near VDD
2	T_4, T_8	T_3, T_7	T_2, T_6, T_{10}, T_{16}	T_1, T_5, T_9, T_{15}
1		$T_{11}, T_{12}, T_{17}, T_{18}$	T_{14}, T_{20}	T_{13}, T_{19}
Weight	0.5	0.4	0.3	0.2

B. Choose top 20% critical paths, increase size of transistors in the chosen path

According to the simulation result using Cadence Spectre, the delays of path 1 to path 10 are 158ps, 158ps, 158ps, 158ps,172ps, 172ps, 122ps, 172ps, 172ps, and 122ps, respectively. The transistors in top 20% of critical path (path 5, path 6, path 8, and path 9) are grouped to *set-x*, and increase their size by equation (3.2):

$$\text{New Size} = \text{Old Size} \times \left(1 + \frac{\text{repeats}}{1 + \text{repeats}} \times \text{weight}\right) \qquad (3.2)$$

C. Identify the first order connections (set-y) to set-x, choose transistors for set-z from set-y that is not in critical paths

Because the channel connected capacitive load increases delay of the critical path, the algorithm reduces the channel connected capacitive load by decreasing the size of transistor in the interacting path. All the transistors directly connected to set-x transistors are grouped to set-y, and transistors in set-y but not in set-x are grouped to set-z. The size of transistor in set-z is reduced by equation (3.3) and equation (3.4) if it is in set-x of previous iteration; otherwise it is decreased by equation (3.5).

$$\text{Temp New} = \text{Old Size} \times \left(1 - \frac{\text{repeats}}{1 + \text{repeats}} \times \text{weight}\right) \qquad (3.3)$$

$$\text{New Size} = \frac{\text{Old Size} + \text{Temp New}}{2} \qquad (3.4)$$

$$\text{New Size} = \text{Old Size} \times \left(1 - \frac{\text{repeats}}{1 + \text{repeats}} \times \text{weight}\right) \qquad (3.5)$$

Repeat (B) and (C) until the worst delay of circuit cannot be further decreased. Then all transistors in Fig. 3.4 have their sizes: T_1 (360nm), T_2 (480nm), T_3 (600nm), T_4 (720nm), T_5 (360nm), T_6 (480nm), T_7 (600nm), T_8 (720nm), T_9 (360nm), T_{10} (480nm), T_{11} (600nm), T_{12} (600nm), T_{13} (360nm), T_{14} (360nm), T_{15} (360nm), T_{16} (480nm), T_{17} (600nm), T_{18} (600nm), T_{19} (360nm), T_{20} (360nm), T_{21} (960nm), based on 250nm CMOS technology. After transistor size optimization, the delay, minimum clock pulse, and power consumption of the dynamic 2-bit binary comparator are 152ps, 200ps, and 500.5 μ W, respectively.

Fig. 3.5 is the transistor level structure of static 2-bit binary comparator on the basis of the equation (3.1). According to the simulation result using Cadence Spectre, the delay, minimum input pulse, and power consumption of static 2-bit binary comparator are 275ps, 300ps, and 277.6 μ W, respectively.

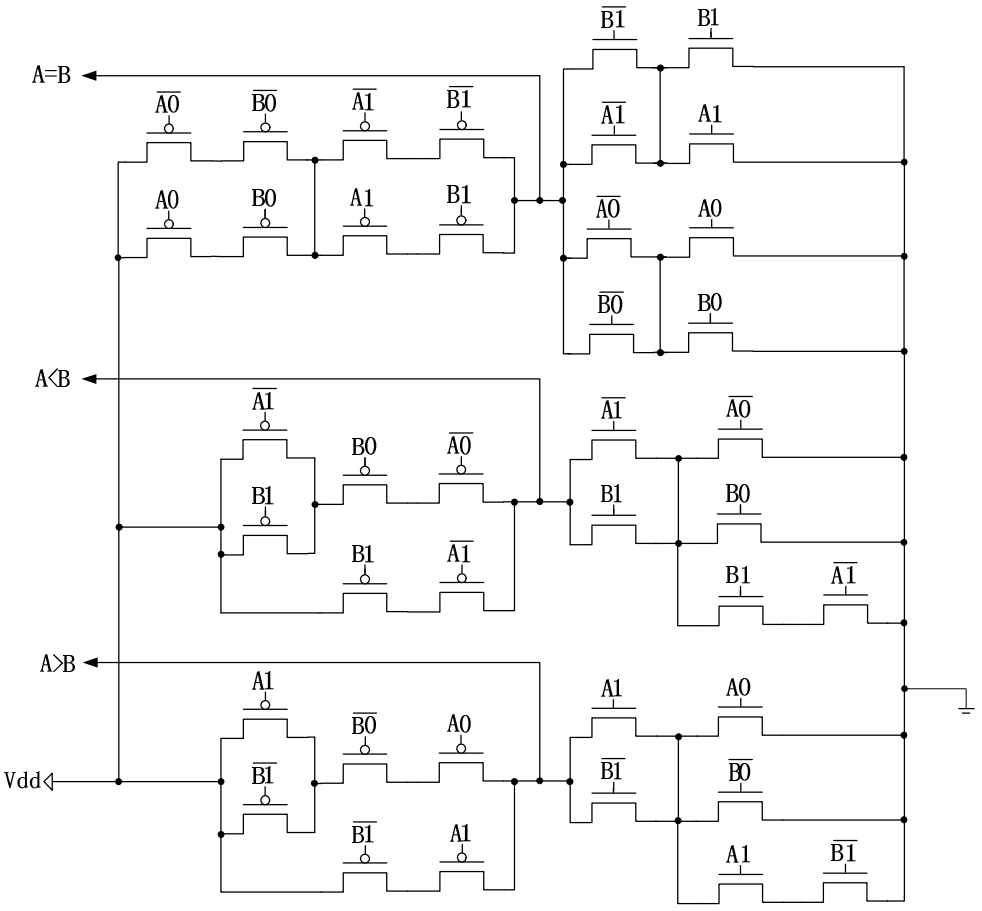


Fig. 3.5 Structure of static 2-bit binary comparator

3.2.2.2 12-input binary comparator

The 12-input binary comparator is used to compare the outputs of four binary comparators. The truth table of the 12-input binary comparator is drawn in Table 3.5, in which "Ag_n", "Al_n", and "eq_n" are inputs of 12-input binary comparator (outputs of nth binary comparator), in which n=4 is for the highest bit and n=1 is for the lowest bit; and "Ag", "Al", and "eq" are outputs of 12-input binary comparator that indicate A is greater than, less than, and equal to B, respectively; x means "don't care", either logic 1 or 0.

Table 3.5 Truth table of 12-input binary comparator

Input												Output		
Ag_4	Al_4	eq_4	Ag_3	Al_3	eq_3	Ag_2	Al_2	eq_2	Ag_1	Al_1	eq_1	Ag	Al	eq
1	0	0	x	x	x	x	x	x	x	x	x	1	0	0
0	1	0	x	x	x	x	x	x	x	x	x	0	1	0

0	0	1	1	0	0	x	x	x	x	x	x	1	0	0
0	0	1	0	1	0	x	x	x	x	x	x	0	1	0
0	0	1	0	0	1	1	0	0	x	x	x	1	0	0
0	0	1	0	0	1	0	1	0	x	x	x	0	1	0
0	0	1	0	0	1	0	0	1	1	0	0	1	0	0
0	0	1	0	0	1	0	0	1	0	1	0	0	1	0
0	0	1	0	0	1	0	0	1	0	0	1	0	0	1

From the Table 3.5, the three canonical minterm equations for each output are simplified down to

$$\begin{cases} \mathrm{eq} = \mathrm{eq}_4 \times \mathrm{eq}_3 \times \mathrm{eq}_2 \times \mathrm{eq}_1 \\ \mathrm{Ag} = \mathrm{Ag}_4 + \mathrm{eq}_4 \times \mathrm{Ag}_3 + \mathrm{eq}_4 \times \mathrm{eq}_3 \times \mathrm{Ag}_2 + \mathrm{eq}_4 \times \mathrm{eq}_3 \times \mathrm{eq}_2 \times \mathrm{Ag}_1 \\ \mathrm{Al} = \mathrm{Al}_4 + \mathrm{eq}_4 \times \mathrm{Al}_3 + \mathrm{eq}_4 \times \mathrm{eq}_3 \times \mathrm{Al}_2 + \mathrm{eq}_4 \times \mathrm{eq}_3 \times \mathrm{eq}_2 \times \mathrm{Al}_1 \end{cases} \quad (3.6)$$

Then the transistor level structure of dynamic 12-input binary comparator, as shown in Fig. 3.6, can be depicted based on Eq. (3.6).

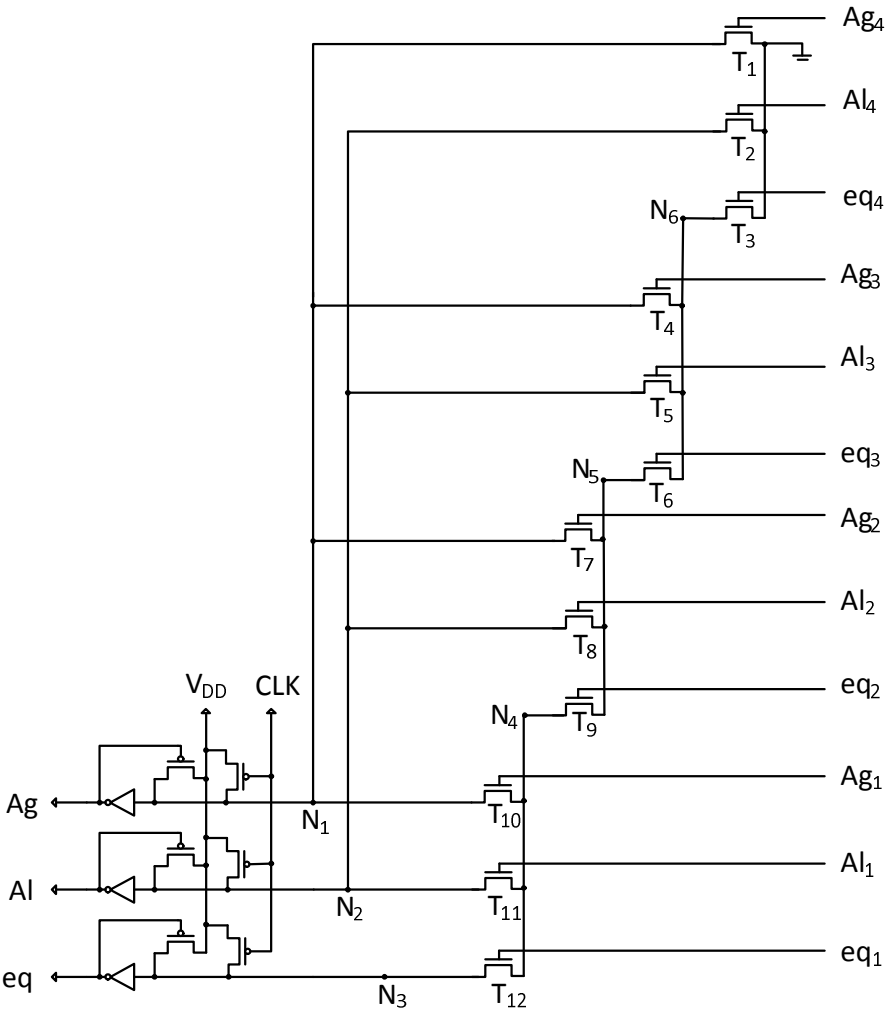


Fig. 3.6 Structure of dynamic 12-input binary comparator

After optimization of transistor size according to the transistor size optimization algorithm mentioned in 3.2.2.1, all transistors and their sizes in Fig. 3.6 are T_1 (360nm), T_2 (360nm), T_3 (3180nm), T_4 (360nm), T_5 (360nm), T_6 (1920nm), T_7 (540nm), T_8 (540nm), T_9 (540nm), T_{10} (540nm), T_{11} (540nm), T_{12} (360nm) based on 250nm technology process.

As the problem in all dynamic CMOS, the worst delay of dynamic 12-input binary comparator is not only the timing of signal propagation through the longest path (T_{12}, T_9, T_6, T_3), but the timing of signal propagation of the longest path after all transistors in longest path are on except the transistor that is farthest away from output

(T_{12}, T_9, T_6 are turned on, T_3 is turned off→ T_{12}, T_9, T_6, T_3 are turned on). N_4, N_5, and N_6 in Fig. 3.6 are pulled to high voltage when T_{12}, T_9, T_6 are turned on and T_3 is turned off, so if T_{12}, T_9, T_6, T_3 are all turned on in next statement, then not only N_3 but N_3, N_4, N_5, and N_6 are all needed to be pulled down. It is much slower than only pulling down N_3. To solve the problem, a pull-down transistor controlled by $\overline{Clock}$ as shown in Fig. 3.7 is connected to N_4, N_5, and N_6. Then the NMOS transistors are turned on and N_4, N_5, and N_6 are pulled down during the precharge-phase. Afterwards, no matter what is the former statement only one node (N_1, N_2, or N_3) needs to be discharged when any pull-down path is on that is much time-economical. The worst delays of 12-input binary comparator with pull-down transistor (143ps) is 24.7% less than that without pull-down transistor (190ps).

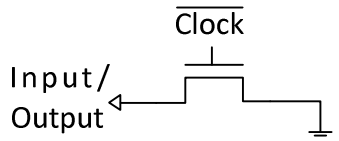


Fig. 3.7 Pull-down transistor

After modification, the clock pulse, and power consumption of dynamic 12-input binary comparator are180ps and 903.9 μ W, respectively.

Fig. 3.8 is the transistor level structure of static 12-input binary comparator according to equation (3.6). Based on the simulation result using Cadence Spectre, the delay, minimum clock pulse, and power consumption of static 12-input binary comparator are 235ps, 210ps, and 390 μ W, respectively.

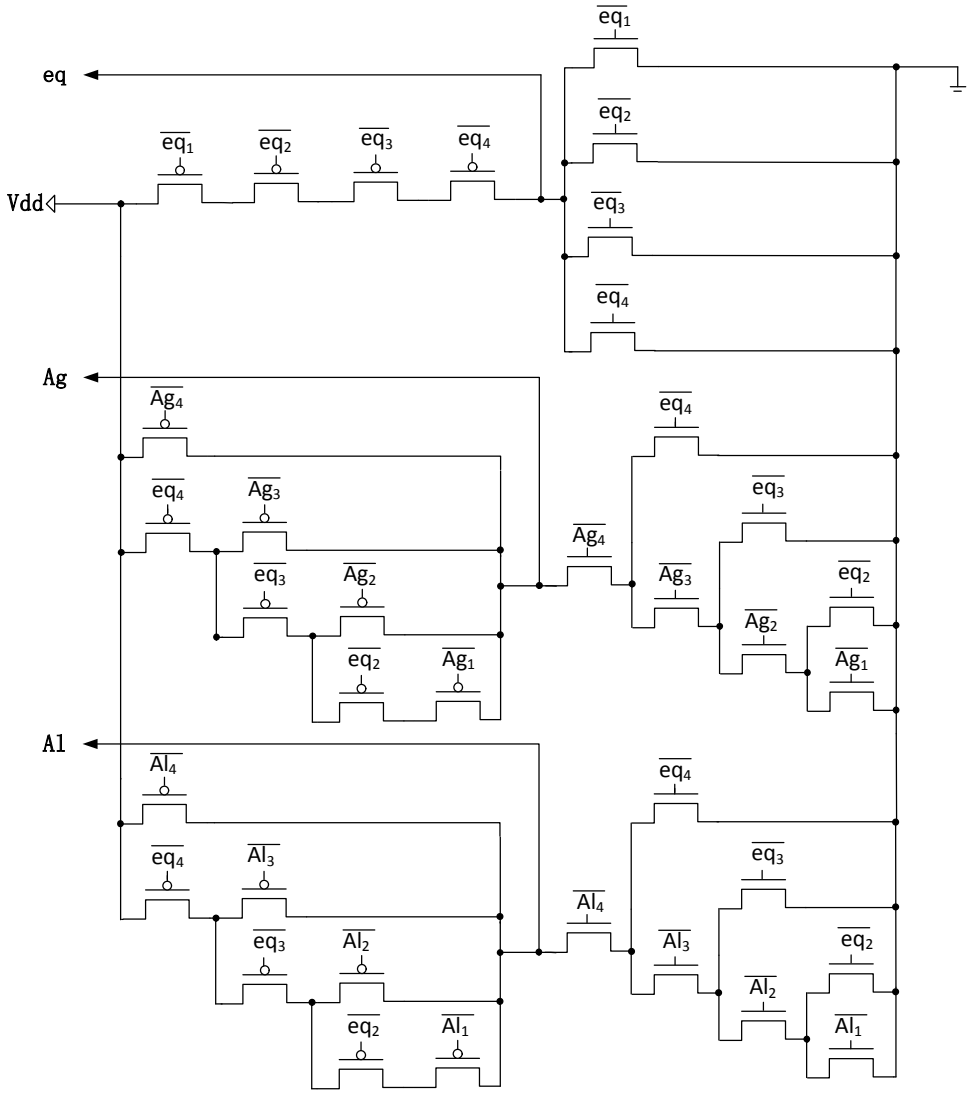


Fig. 3.8 Structure of static 12-input binary comparator

3.2.2.3 6-input binary comparator

The 6-input binary comparator is used to compare the outputs of two binary comparator which have hierarchy. The truth table of 6-input binary comparator is shown in Table 3.6, in which "Ag_n", "Al_n", and "eq_n" are inputs of 6-input binary comparator (outputs of nth binary comparator), in which n=2 is for the high bit and n=1 is for the low bit; "Ag", "Al", and "eq" are outputs of 6-input binary comparator that indicate A is greater than, less than, and equal to B, respectively; x means either logic 1 or logic 0.

Table 3.6 Truth table of 6-input binary comparator

Input						Output		
Ag_2	Al_2	eq_2	Ag_1	Al_1	eq_1	Ag	Al	eq
1	0	0	x	x	x	1	0	0
0	1	0	x	x	x	0	1	0
0	0	1	1	0	0	1	0	0
0	0	1	0	1	0	0	1	0
0	0	1	0	0	1	0	0	1

From Table 3.6, the three canonical minterm equations for each output are simplified down to

$$\begin{cases} \mathrm{eq} = \mathrm{eq}_2 \times \mathrm{eq}_1 \\ \mathrm{Ag} = \mathrm{Ag}_2 + \mathrm{eq}_2 \times \mathrm{Ag}_1 \\ \mathrm{Al} = \mathrm{Al}_2 + \mathrm{eq}_2 \times \mathrm{Al}_1 \end{cases} \quad (3.7)$$

Then based on Eq. (3.7) the transistor schematic of the dynamic 6-input binary comparator is shown in Fig. 3.9.

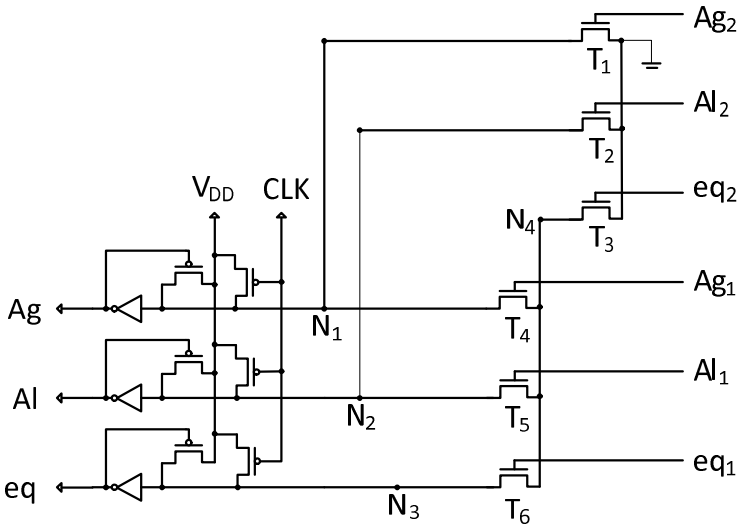


Fig. 3.9 Structure of dynamic 6-input binary comparator

After optimization of transistor size according to the transistor size optimization algorithm mentioned in 3.2.2.1, all transistors and their sizes in Fig. 3.9 are T_1 (1380nm), T_2 (840nm), T_3 (660nm), T_4 (360nm), T_5 (1200nm), T_6 (540nm) based on 250nm technology process. Because of the probable of discharging extra

nodes during evaluate-section that proposed in 3.2.2.2, a pull-down transistor, as shown in Fig. 3.7, needs to be connected to N_4 in Fig. 3.9. According to the simulation result using Cadence Spectre, the delay, minimum clock pulse, and power consumption of dynamic 6-input binary comparator are 85ps, 100ps, and 882.6 μW, respectively.

The transistor level structure of static 6-input binary comparator can be sketched as shown in Fig. 3.10 according to equation (3.7). The delay, minimum clock pulse, and power consumption of static 12-input binary comparator are 125ps, 90ps, and 427.7 μW, respectively.

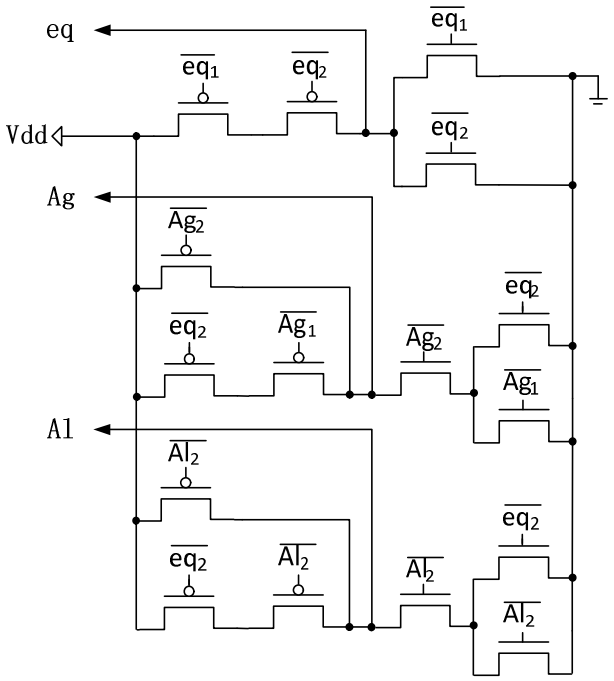


Fig. 3.10 Structure of static 6-input binary comparator

3.3 Timing and Power optimization for mixed-dynamic-static 64-bit binary comparator

According to Fig. 3.1, Fig. 3.2, and Fig. 3.3, 64-bit binary comparator has four operation-stages, 2-bit binary comparator, 12-input binary comparator, 12-input binary comparator, and 6-input binary comparator, shown in Fig. 3.11. How to choose

either static CMOS or dynamic CMOS for the four stages to optimize timing and power is the research we will discuss in this section.

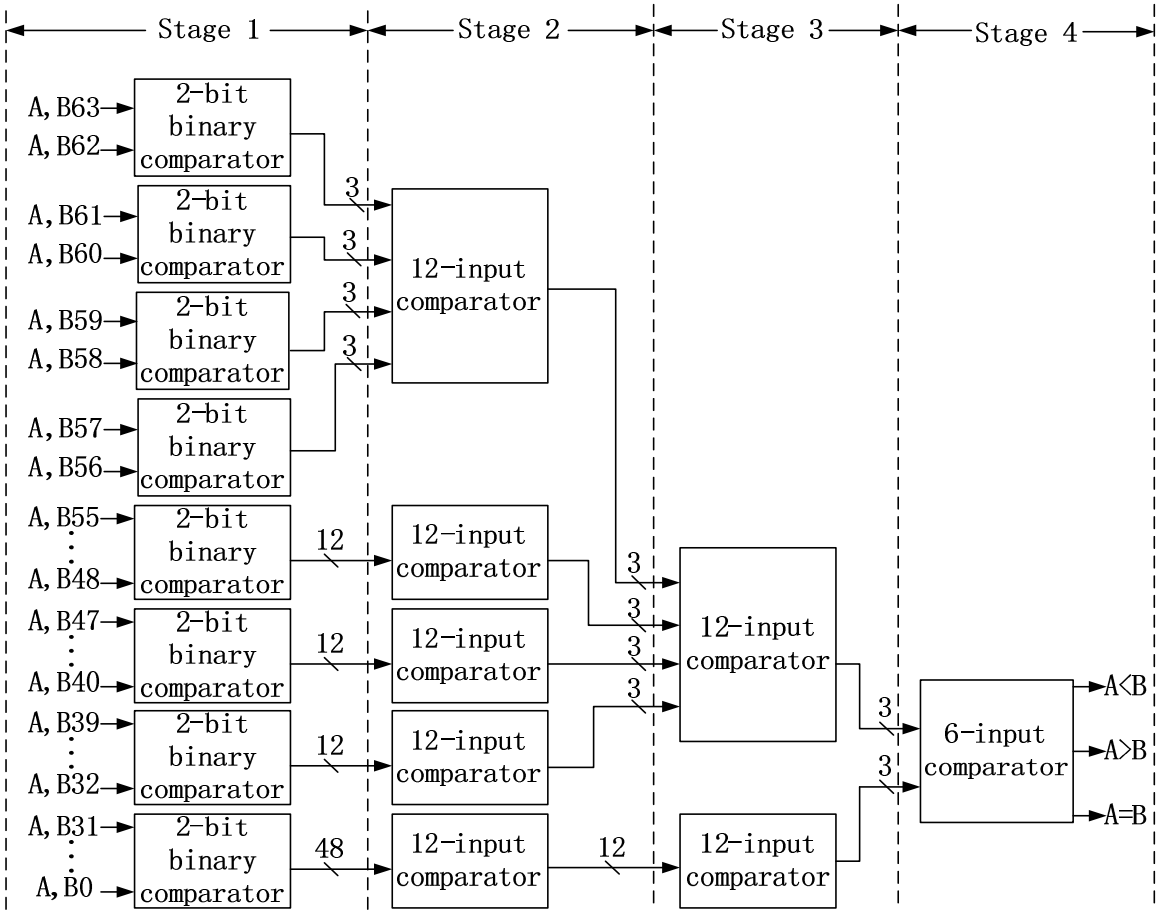


Fig. 3.11 Block diagram of 64-bit binary comparator

3.3.1 Partitioning in 64-bit binary comparator

As mentioned in chapter one, the fastest conventional design for 64-bit binary comparator is full-time dynamic CMOS that requires to separate the four stages in 64-bit binary comparator to two groups, which will be evaluated in evaluate-section and precharge-section, respectively.

According to the simulation result of half-time (conventional) dynamic 64-bit binary comparator, the timing of signal propagation from the input of stage 1 to the output of stage 2 (417ps) and that from the output of stage 2 to the output of stage 4 (321ps) are similar, so the best choice is that group stage 1 and stage 2 together to

operate under one clock pulse and group stage 3 and stage 4 together to operate under another clock pulse. Absolutely, CMOS switch, shown in Fig. 2.12, controlled by the same clock with stage 2 should be inserted between stage 2 and stage 3 in order to hold the value of output of stage 2 for stage 3 during the evaluate time of stage 3.

3.3.2 Timing and Power optimization for 64-bit binary comparator

Timing and power optimization for 64-bit binary comparator of single-clock and multiple-clock will be proposed in this section. The delay of fist stage-group (417ps), stage 1 and 2, is greater than that of second stage-group (321ps), stage 3 and 4. So the 64-bit binary comparator is in situation 2 as classification in chapter 1.

The analysis of delay, power consumption, and minimum clock pulse for dynamic, static, and mixed-dynamic-static 64-bit binary comparator is shown in Table 3.7, in which DDDS means the first, second, and third stages in Fig. 3.11 are dynamic CMOS and the forth one in it is static CMOS; SSDD means the first and second stages in Fig. 3.11 are static CMOS and the third and fourth one in it are dynamic CMOS. For multiple-clock dynamic CMOS, several same circuits operate parallel as pipeline, shown in Fig 1.3, so the delay for data use plays more important role than its clock pulse. Then the best platform for full-time dynamic SSDD 64-bit binary comparator is multiple-clock CMOS, because it can sacrifice clock pulse, which is not important, to achieve low power; other full-time dynamic CMOS and half-time dynamic CMOS should be operated in single-clock CMOS, because they can keep the smallest clock pulse. In Table 3.7, “Delay” is the worst delay of 64-bit binary comparator, signal propagation from input “A0” to output “A=B”; “Clock pulse” is the minimum clock pulse under which the circuit can operate correctly; “Power” is the average power consumption of operation under worst case.

Table 3.7 Power and timing analysis of 64-bit binary comparator

Circuit type	Delay (ps)	Clock pulse (ps)	Power (mW)
Half-time dynamic CMOS	738.5	740	13.21
Full-time dynamic CMOS	377	450	21.83
Full-time dynamic CMOS (DDDS)	440	450	21.1
Full-time dynamic CMOS (SSDD)	377	690	7.51

In single-clock dynamic CMOS, clock pulse cannot be sacrificed, but the first stage-group consumes more delay than the second one, so the second stage-group can be implemented by mixed-dynamic-static CMOS to decrease power consumption and still maintains the same clock pulse. The timing of stage 3 (206ps) occupies roughly 64% of that of the second stage-group (321ps), but the delay of stage-group 1 (417ps) is just 29.9% greater than that of stage-group 2 (321ps), so only stage 4, which is smaller than stage 3, can be replaced by static CMOS in order to keep the same minimum clock pulse. Then for single-clock dynamic CMOS, if power optimization is prior to timing optimization, full-time dynamic DDDS CMOS should be chosen; if timing optimization is prior to timing optimization, full-time dynamic CMOS is the optimal choice. For multiple-clock dynamic CMOS, the delay for data use plays more important role than its clock pulse, so clock pulse can be sacrificed to decrease power consumption that can be implemented by replacing stage-group 1 with static CMOS.

As seen in Table 3.6, the delay of full-time dynamic CMOS (377ps) is almost half of that of half-time dynamic CMOS (738.5ps), and 14.3% less than that of full-time dynamic DDDS CMOS; full-time dynamic DDDS CMOS and full-time dynamic CMOS have the same clock pulse, but the power consumption of the former one (21.1mW) is decreased by 3.3% compared that of the latter one (21.83mW).

For multiple-clock circuit, clock pulse can be increased to decrease power consumption that can be implemented by replacing stage-group 1 in Fig. 3.11 with

static CMOS. So even minimum clock pulse of full-time dynamic SSDD CMOS (690ps) is 53.3% greater that of other full-time dynamic COMS, the best choice for multiple-clock CMOS is still it. As shown in Table 3.6, the delay of full-time dynamic SSDD CMOS (377ps) is the same with that of full-time dynamic CMOS, and 14.3% and 50% less than that of full-time dynamic DDDS CMOS (440ps) and full-time dynamic CMOS (738.5ps), respectively; the power consumption of full-time dynamic SSDD CMOS is about 43.1%, 65.6%, and 64.4% less than that of full-time dynamic CMOS, half-time dynamic CMOS, and full-time dynamic DDDS CMOS, respectively.

4 CONCLUSION AND FUTURE WORK

4.1 Conclusion

A general study of optimizing power and timing for dynamic CMOS has been presented. The fastest design of dynamic CMOS is full-time dynamic CMOS, in which circuit needs to be divided to two group that evaluate in evaluate-section and precharge-section, respectively. There are two situations for circuit after partition, the propagation delays of two groups are equal and not equal. 16-bit CSA is proposed as an example for the situation the delays are equal, and 64-bit binary comparator is showed for the situation the delays are not equal.

For 16-bit CSA, if it is applied in single-clock circuit, timing optimization should choose full-time dynamic CMOS, whose worst delay is decreased by 41.1% compared with the conventional (half-time) dynamic CMOS; if it is applied in multiple-clock circuit, the CMOS evaluated in former clock pulse should be replaced by static CMOS, then the power consumption and delay are reduced by 43% and 41.1%, respectively, compared with the conventional (half-time) dynamic CMOS.

For 64-bit binary comparator, if it is applied in single-clock circuit, timing optimization should choose full-time dynamic CMOS, whose delay is decreased by 49% compared with the conventional (half-time) dynamic CMOS, and in power optimization, the CMOS evaluated in later clock pulse should be implemented by mixed-dynamic-static CMOS, in which the circuit can operate under the same minimum clock pulse, but the power consumption is decreased by 3.3% compared with the power-optimized circuit. If it is applied in multiple-clock circuit, the CMOS

evaluated in former clock pulse should be designed by static CMOS. In comparison with the conventional (half-time) dynamic CMOS, the power consumption and delay are reduced by 43.1% and 49%, respectively.

4.2 Future work

The theory proposed in the thesis is just how to choose dynamic CMOS or static CMOS for every part of a circuitry to optimize power or timing. Another important method to decrease power and timing is the design tactics of single CMOS. For example, the power of dynamic CMOS can be decreased by reducing the number of transistor, decreasing the probability of pulling down logic 1 to 0, and etc. The timing of dynamic CMOS can be decreased by decreasing the difference of high voltage and low voltage, decreasing the load of output, and etc. The ideas above will be the future work of this research to further optimize power and timing for dynamic CMOS.

5 REFERENCE

[1] R. Jacob Baker, Harry W. Li and David E. Boyce, “CMOS Circuit Design, Layout, and Simulation”, pp. 282

[2] Anjuli, Satyajit Anan, “A High-Speed 64-Bit Binary Comparator”, IOSR Journal of Electronics and Communication Engineering (IOSR-JECE) e-ISSN: 2278-2834, p-ISSN: 2278-8735. Volume 4, Issue 5 (Jan. – Feb. 2013), pp 38-50

[3] B. Ramkumar and H. M. Kittur, “Low-Power and Area-Efficient Carry Select Adder,” IEEE Trans. On VLSI Systems, Vol. 20, No. 2, pp. 371-375, Feb. 2012

[4] Vladimir V. Shubin, “New High-Speed CMOS Full Adder Cell of Mirror Design Style,” XI International Conference and Seminar EDM’2010, section II, June 30-July 4, 2010

[5] Hashemian, Reza, “Design of a 54-bit Adder Using a modified Manchester Carry Chain, ”VLSI, 1994. Design Automation of High Performance VLSI Systems. GLSV '94, Proceedings., Fourth Great Lakes Symposium on4-5 Mar 1994

[6] Anjuli, SatyajitAnand, “High-Speed 64-Bit CMOS Binary Comparator”, Innovative Systems Design and Engineering, ISSN 2222-1727 (Paper) ISSN 2222-2871 (Online), Vol.4, No.2, 2013

[7] Pierce Chuang, David Li, and ManojSachdev, Fellow, IEEE, “A Low-Power High-Performance Single-Cycle Tree- Based 64-Bit Binary Comparator”, IEEE Transactions on Circuits and Systems-II: Express Briefs, Vol.59, No. 2, February 2012

[8] C.-H. Huang and J.-S. Wang, “High-performance and power-efficient CMOS comparators,” IEEE J. Solid-State Circuits, vol. 38, no. 2, pp. 254-262, Feb. 2003

[9] H.-M. Lam and C.-Y.Tsui, “A MUX-based high-performance single-cycle CMOS

comparator," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 54, no. 7, pp. 591-595, Jul. 2007

[10] J.-Y. Kim and H.-J.Yoo, "Bitwise competition logic for compact digital comparator," in Proc. IEEE Asian ASSCC, 2007, pp. 59-62

[11] S. Perri and P. Corsonello, "Fast low-cost implementation of single-clock-cycle binary comparator," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 55, no.12, pp. 1239-1243, Dec. 2008

[12] F. Frustaci, S. Perri, M. Lanuzza, and P. Corsonello, "A new low-power high-speed single-clock-clock-cycle binary comparator," in Proc. IEEE Int. Sym. Circuits Syst., 2010, pp. 317-320"

[18] Kumar Yelamarthi, Henry Chen, "Process Variation-Aware Timing Optimization for Dynamic and Mixed- Static-Dynamic CMOS Logic", IEEE Transactions on semiconductor manufacturing, Vol. 22, No. 1, February 2009