Signed Comparator By Adder or Magnitude Comparator

Name: Noor Aldeen Tirhi

Student Number: 1190081

Introduction

In the world of digital electronics we use a lot of combinational circuits on a daily basis, one of most frequently used is the comparator, there are multiple types of this circuit, one type is a magnitude comparator, this circuit takes in bits of two operands, we can call them A and B, and interprets them as positive integers, this circuits outputs 3 bits marked as, A>B, A<B and A=B, only one of these bits will be set to ‘1’ at a time, so for example a 4 bit magnitude comparator with A = “1101” and B =”0101” will have the output A>B= ‘1’, A<B= ’0’, A=B = ‘0’ , another type of comparator is the signed comparator, specifically 2’s complement signed comparator, this comparator takes in the same input as the magnitude comparator, only difference is that the signed comparator interprets the operands in 2’s complement’s form, so they can be negative or positive, so for example A=’1101” and B=”0101” will have the output A>B = ‘0’, A<B = ‘1’, A=B = ‘0’.

Design Theory

We will need to Design an 8-bit Signed Comparator using an Adder circuit and an 8-bit Signed Comparator using a magnitude comparator.

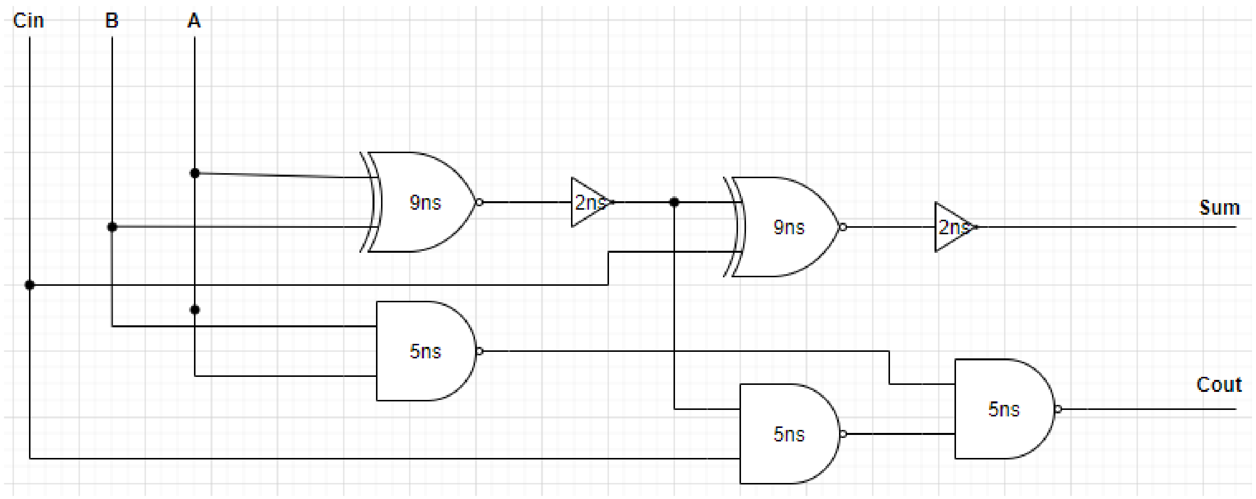
Signed Comparator by Adder

The Design Approach taken is that the adder will compare the 7 least significant bits of both operands, and the most significant bits of both operands are used to manipulate the output of the adder into the appropriate output of the signed comparator.

Before designing the comparator circuit, the adder circuit must be designed, this circuit will be designed of 7 identical full adders.

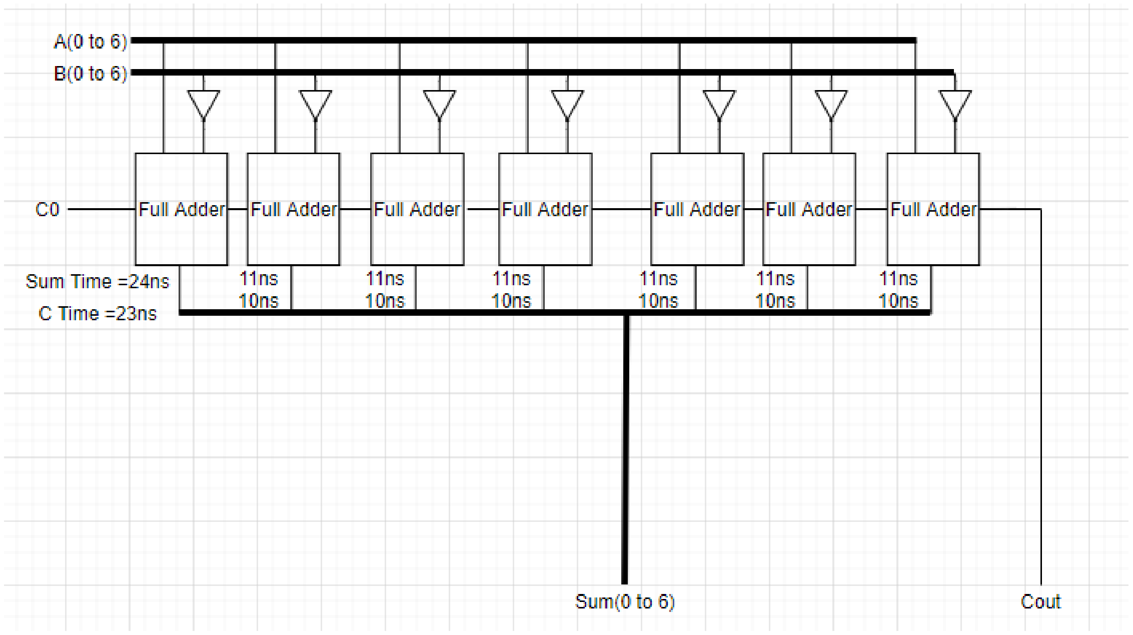
In this report the circuits to be designed are an 8-bit 2’s complement signed comparator by using an adder, and an 8-bit 2’s complement signed comparator by using a magnitude comparator.

Full Adder:



The full adder is designed with Xnors and inverters for calculating the Sum, instead of Xors, this shaves off 1ns of propagation time, also the Cout is calculated using only Nand gates instead And and Or gates, this works out due to Demorgan’s law, this shaves off 4 ns for the Cout, in total the circuit produces the Sum in 22ns and the Cout in 21ns.

Many instances of this circuit will be used to design the 7-bit full adder



The input bits of the B operand will all be inverted, and C0(Cin) will be set to '1', this will act as a ripple subtractor circuit, it produces the output bus Sum and the output bit Cout.

This circuit needs a total of 84 ns to produce the correct Sum output, and 83 ns to produce the correct Cout bit, the time is shorter than expected due to some of the work being done in parallel, this concerns work that doesn't need the Carry bit of the previous full adder.

After extensive testing it was concluded that Cout bit indicates which operand has the larger magnitude, if $\mathrm{Cout} = \text{'1'}$ that means $\mathrm{A}(6 \text{ down to } 0) \geq \mathrm{B}(6 \text{ down to } 0)$, $\mathrm{Cout} = \text{'0'}$ means $\mathrm{A}(6 \text{ down to } 0) < \mathrm{B}(6 \text{ down to } 0)$, another indicator that the we get from this circuit is the sum, specifically wither the sum is equal to 0 or not, this can be produced by a 7 bit Nor gate that will produce to the 'Zero' bit.

$$\mathrm{Zero} = (\mathrm{Sum}(0) + \mathrm{Sum}(1) + \mathrm{Sum}(2) + \mathrm{Sum}(3) + \mathrm{Sum}(4) + \mathrm{Sum}(5) + \mathrm{Sum}(6))`$$

Finally we attend to the A(7) and B(7), these bits decide the sign of their respective operands, and so we need to do a truth table to figure out the digital circuit that produces the sought after results.

A(7)	B(7)	Cout	Zero	A>B	A<B	Eq
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	1	0	0
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	1	0	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	1	0
1	0	0	1	0	1	0
1	0	1	0	0	1	0
1	0	1	1	0	1	0
1	1	0	0	0	1	0
1	1	0	1	0	0	1
1	1	1	0	1	0	0
1	1	1	1	0	0	1

🟧 "A=B" = '1' Rows	🟩 "A>B" = '1' Rows	🟦 "A<B" = '1' Rows

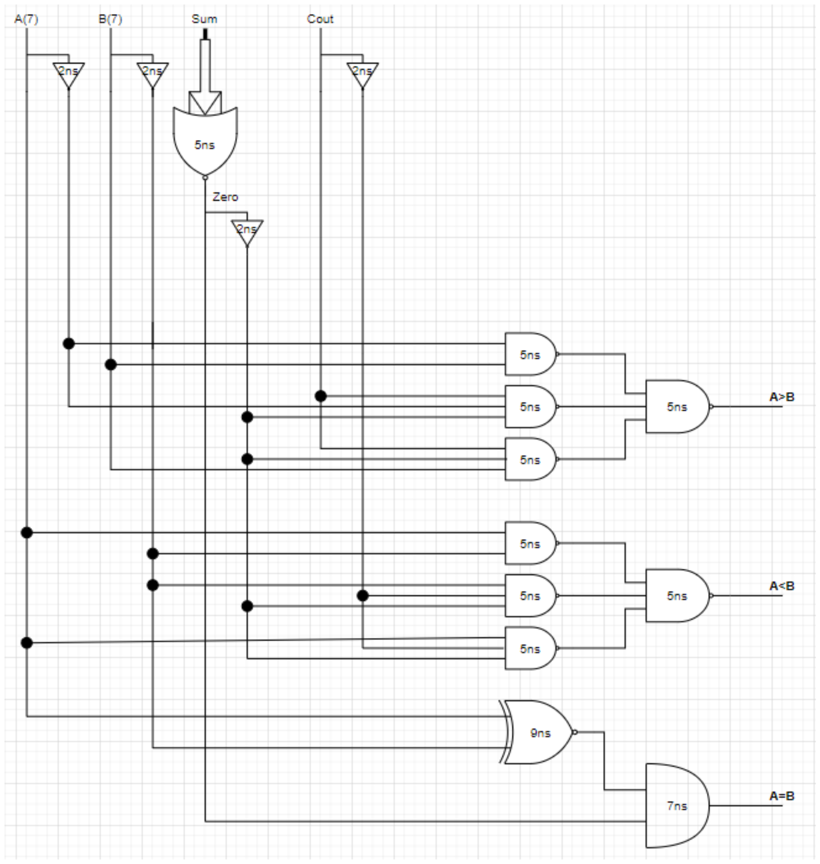
Using K-maps we reach circuits that will produce the required output:

$$(A > B) = A(7)`.B(7) + Cout.Zero`.A(7)` + Cout.Zero`.B(7)$$

$$(A < B) = A(7).B(7)` + Cout`.Zero`.A(7) + Cout`.Zero`.B(7)`$$

$$(A = B) = Zero.(A(7) \oplus B(7))`$$

The (A>B) and the (A<B) outputs are achieved using Nand gates using 2-level Nand design instead AndOr design, this is 4 ns faster.



The last Correct input to reach this part of the circuit is the Sum, and all gates depend on it, A>B and A<B have the same propagation time of 17 ns since the arrival of the Sum bus, A=B on other hands needs 16 ns.

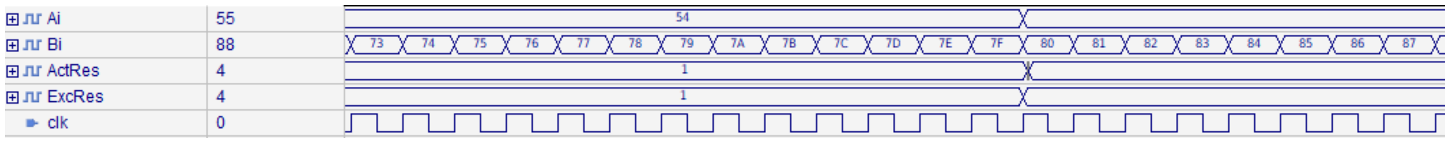
In total the circuit needs 101ns before producing the correct output.

Which means the circuit can have frequency of about 9.90 MHz.

<u>Tester Circuit</u>

The approach for testing the circuit was using a Test Vector Generator and a result analyzer for each possible vector for the circuit, 2^16 vectors are made which is doable on modern processers.

This is an excerpt of the simulation running on a clock that has 102ns between each adjacent positive edge.


Ai 55
Bi 88
ActRes 4
ExcRes 4
clk 0
54
73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F 80 81 82 83 84 85 86 87
1
1

ActRes and ExcRes are vectors that are comprised of the following bits (A<B, A=B, A>B), these two vectors are checked every rising edge with an assert statement.

Signed Comparator by Magnitude Comparator

We use the same method we used for the adder, compare the first 7 bits, and then manipulate the result of the magnitude comparator using the sign bit, we use a truth table and K-maps to produce the design that will give us the output we need:

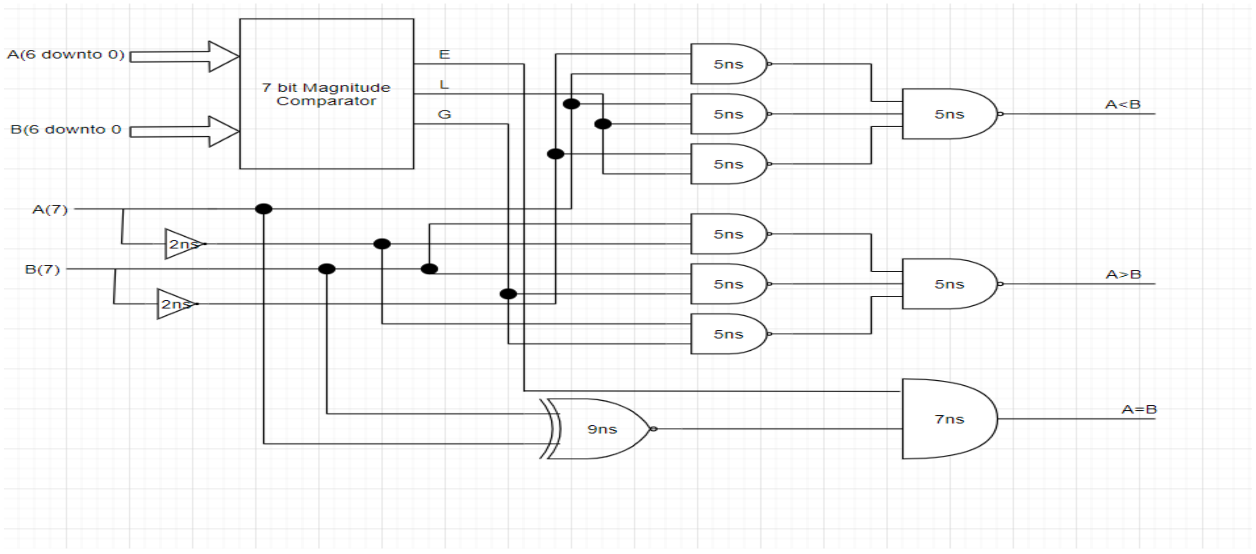
A(7)	B(7)	G	L	E	Grt	Eq	Less
0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1
0	1	0	0	0	1	0	0
0	1	1	1	1	1	0	0
1	0	0	0	0	0	0	1
1	0	1	1	1	0	0	1
1	1	0	0	0	0	0	0
1	1	1	1	1	1	1	1

Where G, L and E are the output bits from the magnitude comparator, the design that produces this out is as follows :

Grt = A(7)`.B(7) + B(7).G + A(7)`.G

Eq = (A(7)⊕B(7))`.E

Less = A(7).B(7)` + A(7).L + B(7)`.L

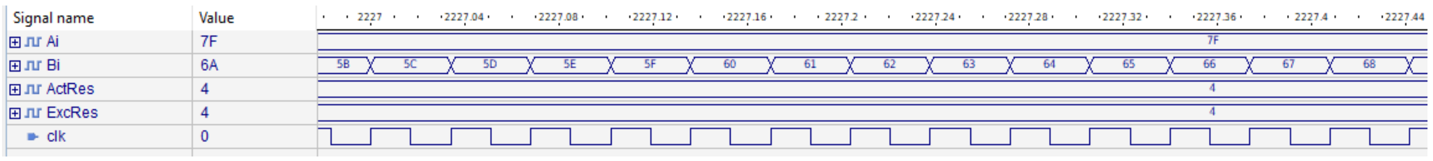


Magnitude Comparator needs 23ns and the output need the A>B and A<B need another 10ns, so in the end the propagation time is 33ns, the A=B bit needs 16+7 = 23ns, so we take the largest propagation time, and that is 33ns.

Testing Circuit

We test this circuit in the same method used to test the adder circuit.

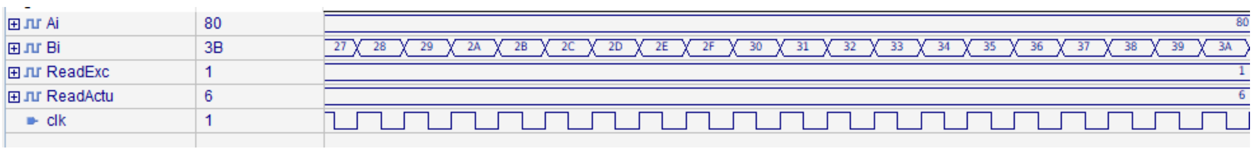
This is an excerpt from the simulation of the result analyzer with the test vector generator, which is running on the pace of a clock which has 34ns between each adjacent rising edge, which means it runs at 29.411MHz.

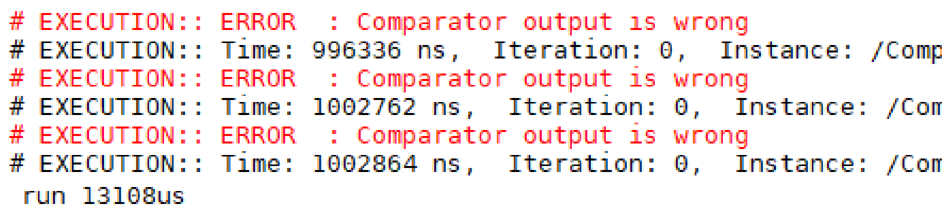


Flawed Circuits

Flawed Adder Signed Comparator

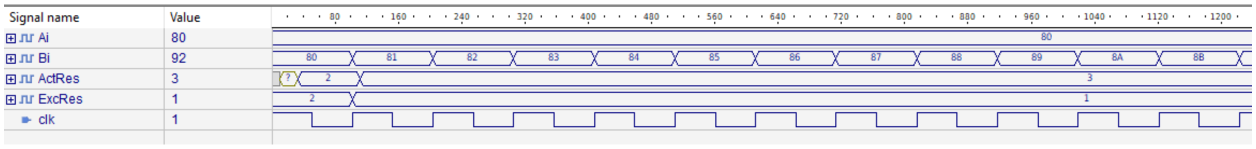
We simulate a flawed signed comparator, the difference in this flawed circuit is that A>B and A<B minterms are put into an And circuit instead of a Nand gate, also the A=B is inverted.

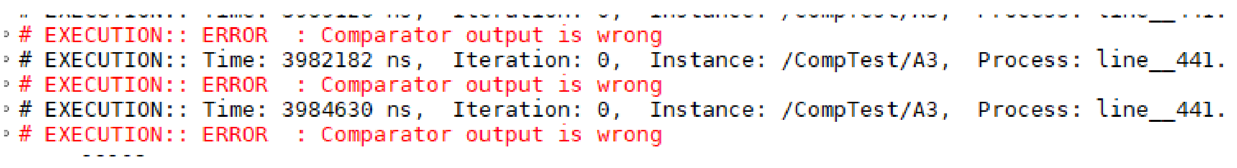



```
# EXECUTION:: ERROR  : Comparator output is wrong
# EXECUTION:: Time: 996336 ns,  Iteration: 0,  Instance: /Comp
# EXECUTION:: ERROR  : Comparator output is wrong
# EXECUTION:: Time: 1002762 ns,  Iteration: 0,  Instance: /Com
# EXECUTION:: ERROR  : Comparator output is wrong
# EXECUTION:: Time: 1002864 ns,  Iteration: 0,  Instance: /Com
 run 13108us
```

Flawed Magnitude Comparator Signed Comparator

We simulate flawed signed comparator, the difference is that in this circuit the x7 bit, used to make sure check for equality between the the signed bits, also x7 bit is nanded with the E bit instead of anded.




```
# EXECUTION:: ERROR  : Comparator output is wrong
# EXECUTION:: Time: 3982182 ns,  Iteration: 0,  Instance: /CompTest/A3,  Process: line__441.
# EXECUTION:: ERROR  : Comparator output is wrong
# EXECUTION:: Time: 3984630 ns,  Iteration: 0,  Instance: /CompTest/A3,  Process: line__441.
# EXECUTION:: ERROR  : Comparator output is wrong
```

Conclusion

Signed Comparator circuits can be created with many different approaches, the adder way the approach taken here in both cases, is comparing the least significant bits and then seeing how the sign bit changes the output, through the adder it’s slightly slower and more complex to implement than with the magnitude comparator, but it could be useful if only an adder is available.

	Adder	Comparator
Propagation Delay	33ns	101ns

Appendix:

Sources:,

Digital Design, By Morris Mano and Michael D.Ciletti, 2012, 5th Edition.