

**Electrical and Computer Engineering Department**

**ENCS533**

**Advanced Digital System Design**

**Adham Tamimi**

**1162429**

**Report of Summer Project**

**8/5/2019**

Contents

[ Introduction 2](#_Toc15898601)

[ Theoretical overview: 2](#_Toc15898602)

[ Digital logic | Magnitude Comparator: 2](#_Toc15898603)

[1-Bit Magnitude Comparator – 3](#_Toc15898604)

[2-Bit Magnitude Comparator – 4](#_Toc15898605)

[4-Bit Magnitude Comparator – 4](#_Toc15898606)

[Cascading Comparator – 4](#_Toc15898607)

[ Design philosophy 5](#_Toc15898608)

[ Equality Circuit: 5](#_Toc15898609)

[ A > B, A < B Circuit: 5](#_Toc15898610)

[ Analyzer and Generation Entities: 6](#_Toc15898611)

[ The full Circuit: 6](#_Toc15898612)

[ Simulation Results: 7](#_Toc15898613)

[ References 9](#_Toc15898614)

**Appendix was submitted externally**

# 

# Introduction

In this project, an 8 bit both signed and unsigned comparator was assembled structurally, from basic logic gates with specific delays, and then blocks of the comparator itself assembled from these gates. After that, both a value generation entity and a value analyzer entity was built in order to test out the results of all possible values entering the comparator.

# Theoretical overview:

## Digital logic | Magnitude Comparator:

A magnitude digital Comparator is a combinational circuit that **compares two digital or binary numbers** in order to find out whether one binary number is equal, less than or greater than the other binary number. We logically design a circuit for which we will have two inputs one for A and other for B and have three output terminals, one for A > B condition, one for A = B condition and one for A < B condition.

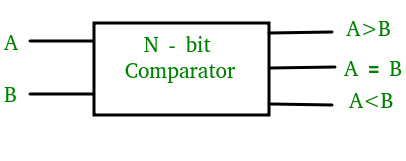


Figure N bit Comparator

### 1-Bit Magnitude Comparator –

A comparator used to compare two bits is called a single bit comparator. It consists of two inputs each for two single bit numbers and three outputs to generate less than, equal to and greater than between two binary numbers.

The truth table for a 1-bit comparator is given below:

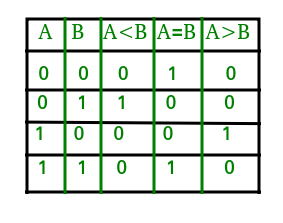


Figure Truth table of 1 bit comparator

From the previous truth table , we are able to get the Boolean expressions and therefore the digital circuit that represents this comparator , which looks as follows :

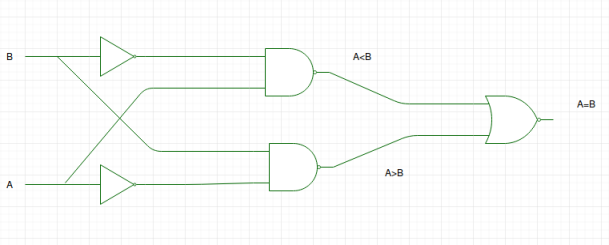


Figure 2 bit comparator

Keep in mind there are many variations of this circuit, each depends on Its’ own Boolean expression.

### 2-Bit Magnitude Comparator –

A comparator used to compare two binary numbers each of two bits is called a 2-bit Magnitude comparator. It consists of four inputs and three outputs to generate less than, equal to and greater than between two binary numbers.

It’s implemented in the same way of getting the truth table and using the k-map method to get its Boolean expression in order to end up with the following digital circuit.

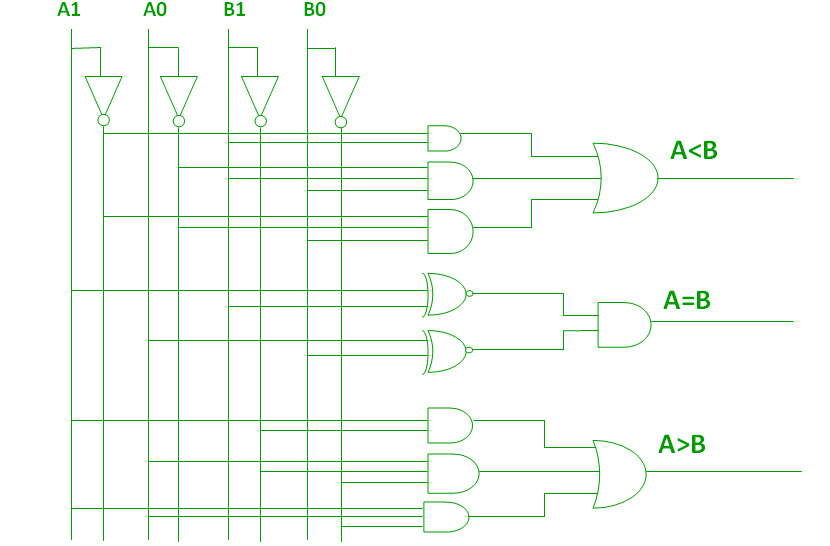


Figure 2 bit magnitude comparator

### 4-Bit Magnitude Comparator –

The 4 –bit magnitude comparator is assembled in the same way, but it will be skipped because its implementation goes the same way all over .

### Cascading Comparator –

A comparator performing the comparison operation to more than four bits by cascading two or more 4-bit comparators is called cascading comparator. When two comparators are to be cascaded, the outputs of the lower-order comparator are connected to corresponding inputs of the higher-order comparator.

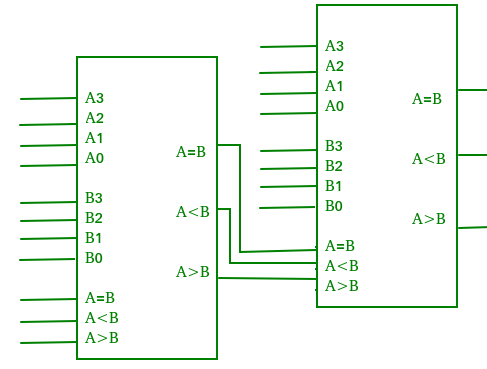


Figure cascade comparator

# Design philosophy

In my design , I looked for a design that didn’t have to consist of 2 separate circuits for signed and unsigned numbers, I build my design on a circuit that included both , which makes the design faster and the enable input or the MUXs didn’t have to be included . I tried to skip stages as much as I could to end up with the following 2 circuits:

## Equality Circuit:

The main idea of 2 numbers being equal is that all of their corresponding bits have to be identical , and the XNOR gate is a gate that only gives an output of ‘1’ if the 2 inputs were the same , based on this idea , this is the design I ended up with :

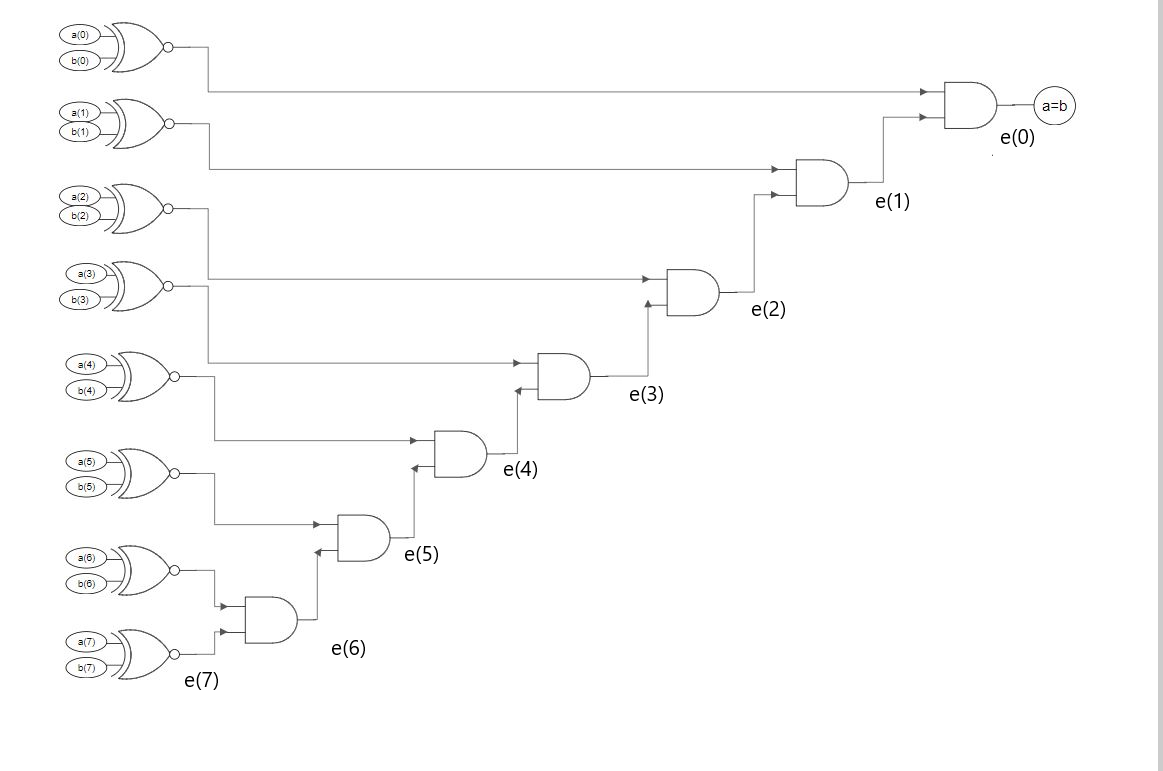


Figure A = B

## A > B, A < B Circuit:

The main Idea that was used in this design, is that if a bit in A in bigger than a bit in B in the same position, and all other bits are equal or A>B in them, then A > B, whether the numbers are signed or unsigned, as for the difference between the signed and the unsigned comparison difference it occurs only in the sign bit (MSB), which I analyzed separately in the circuit. Finally, A < B if A is not bigger than B NOR equal to it , that’s why I used the last not gate.

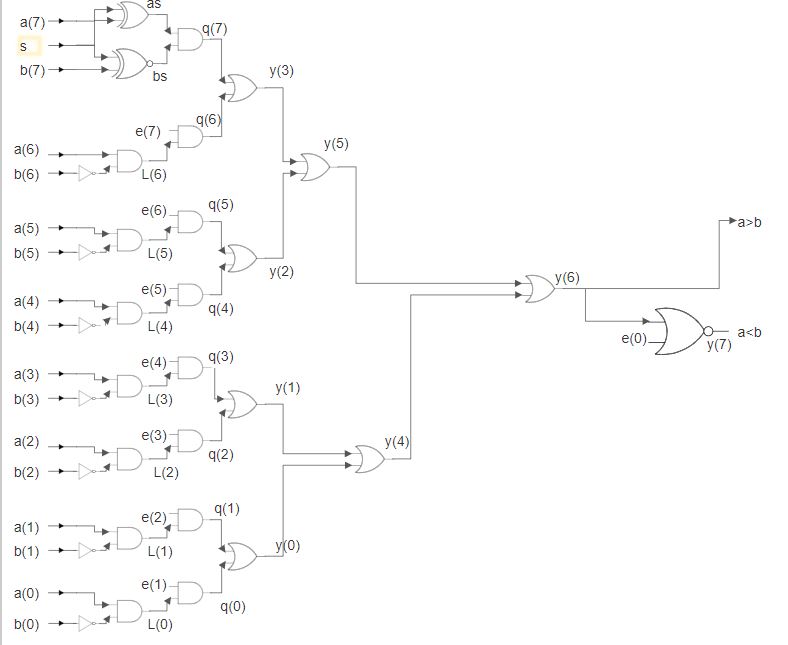


Figure Comparison circuit

## Analyzer and Generation Entities:

The generation entity was used to generate all possible values for a and b to test them plus generating the real value of comparison to be compared in the Analyzer circuit.

## The full Circuit:

After combining the equality and comparison circuit, registers were added to the inputs and outputs to avoid glitches and fast input changing.

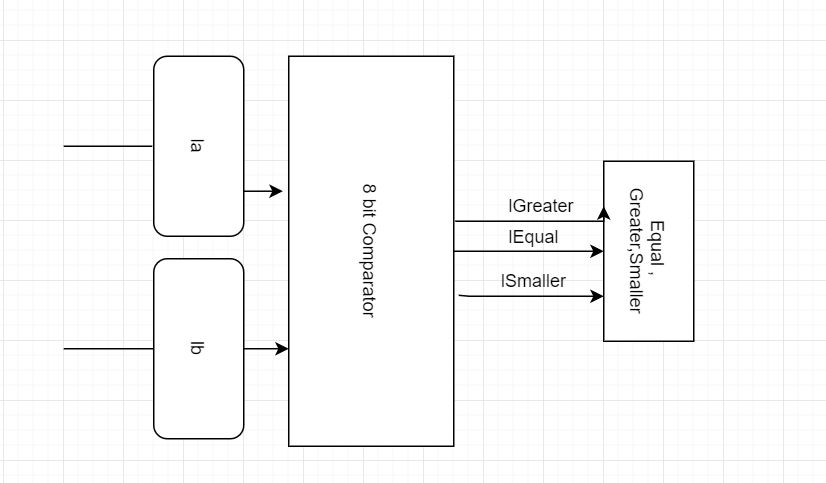


Figure Full Comparator Circuit

And then finally, analyzer and generator were added to end up with the final assembly:

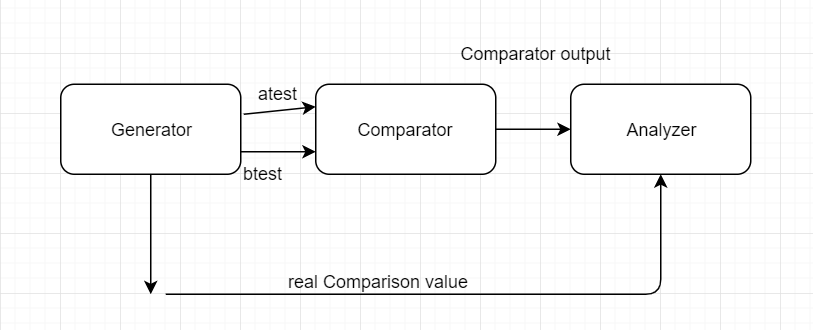


Figure Final Assembly

# Simulation Results:

The simulation for the gates are not necessary to be included , but the delay is shown in the code which is included in the appendix.

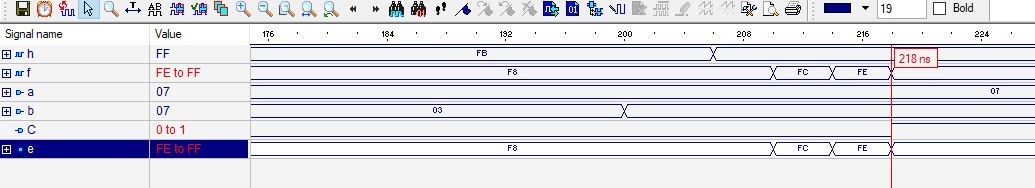


Figure a=b Simulation with 18ns delay

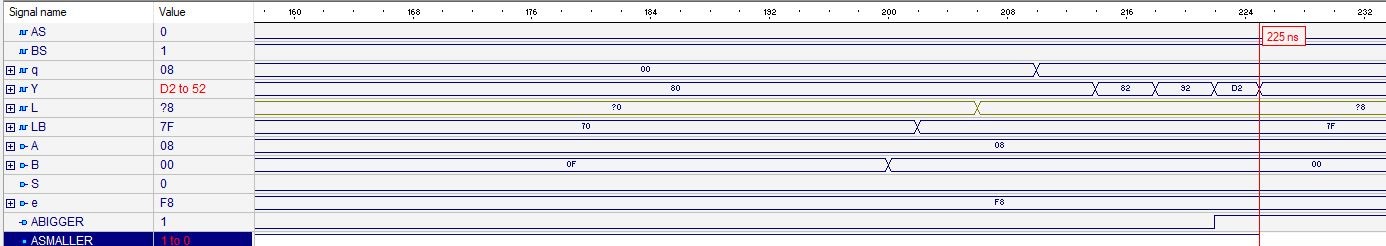


Figure From Smaller to bigger in 25 NS

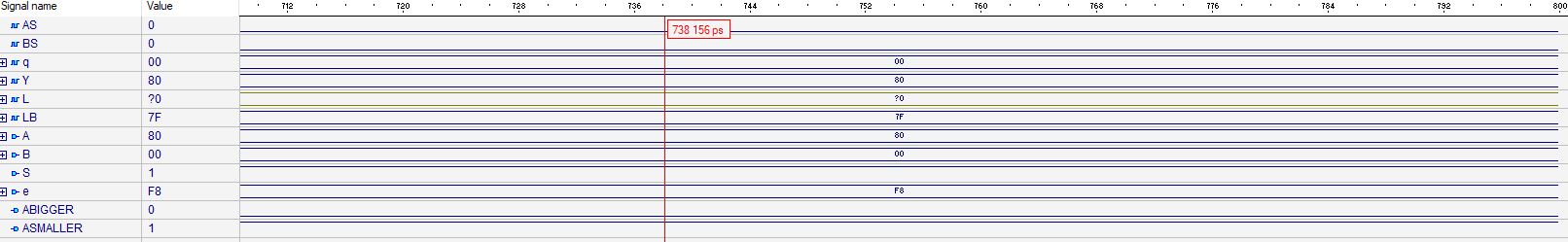


Figure Full comparator signed case

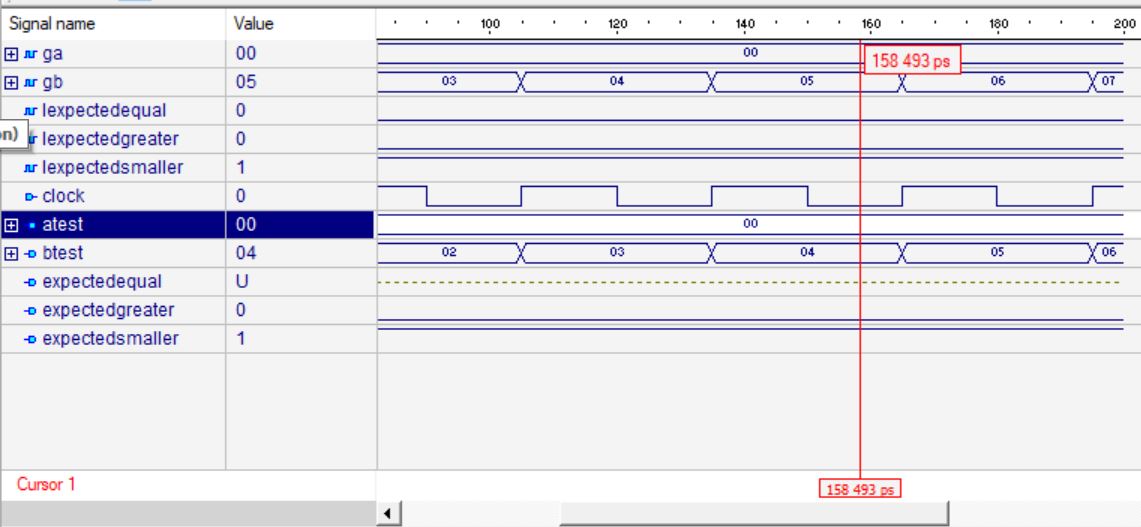


Figure Generation Simulation

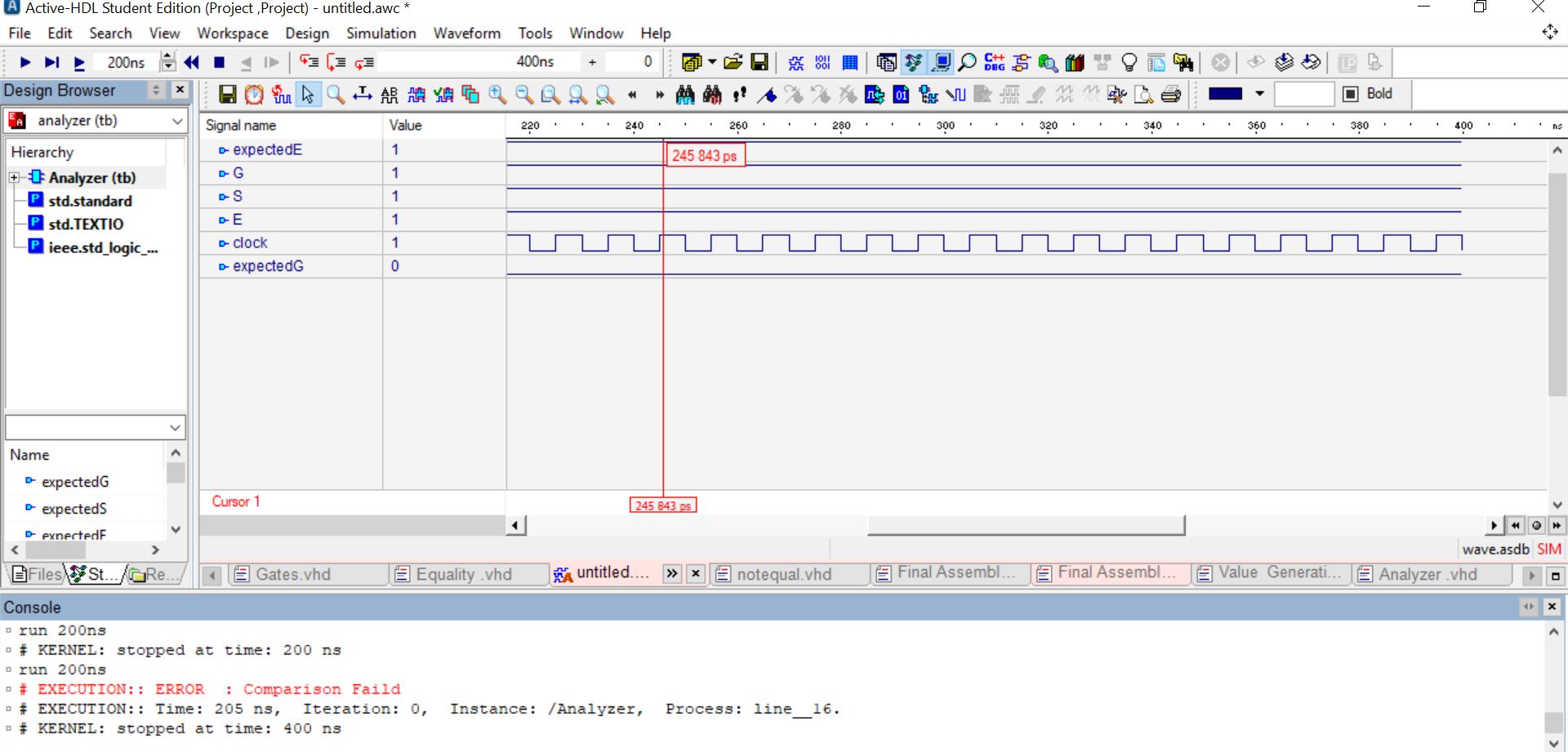


Figure Analyzer simulation with error shown

And finally , the full testing simulation with no error in console :

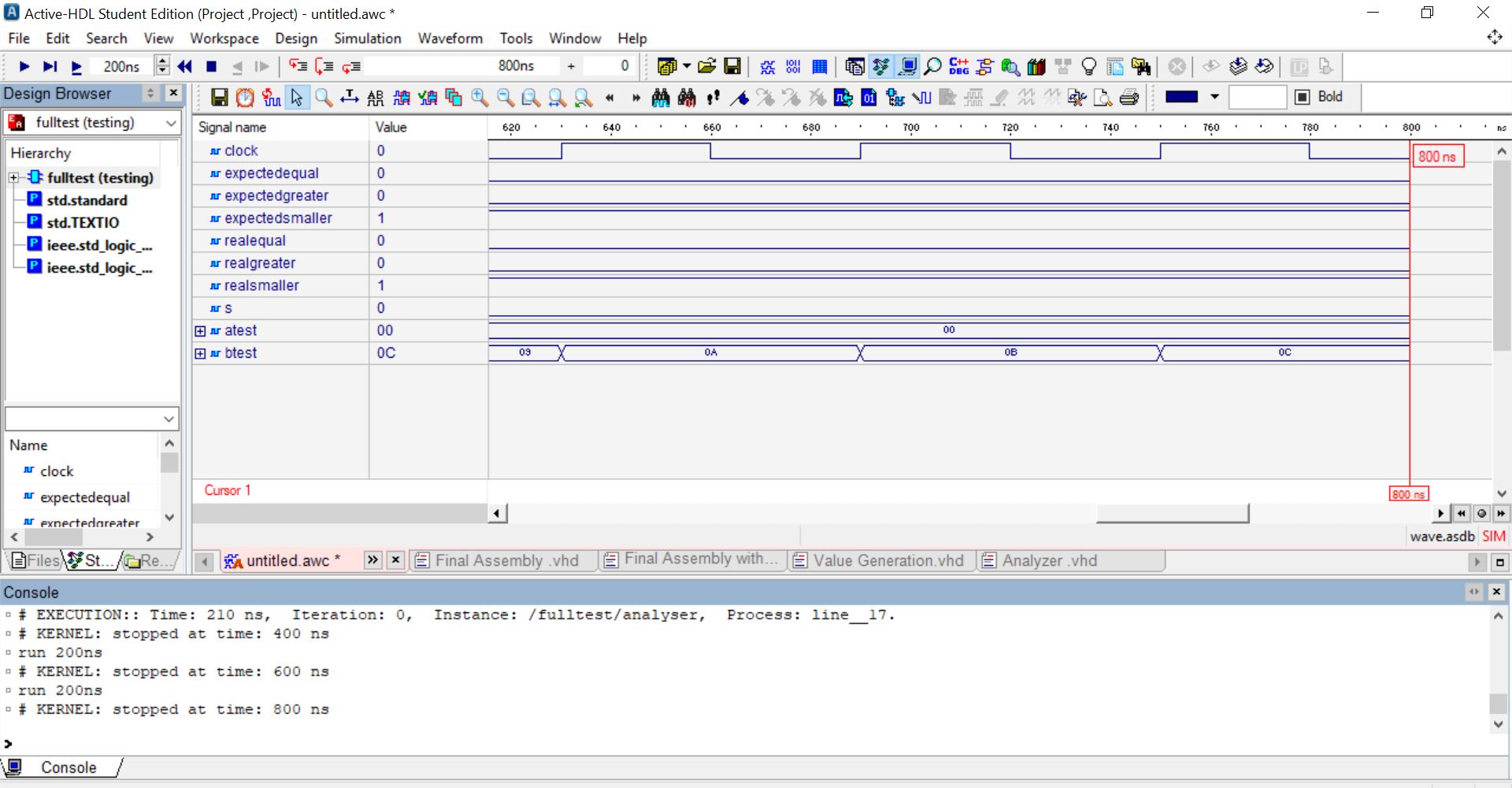


Figure Full Testing Simulation

# Future work and conclusions:

* Design fulfilled the requirements and can compare 2 8-bit numbers signed and unsigned.
* Design had 25 ns total delay which Is minimal.
* Design could have used multiple input gates which could have skipped some stages that might have saved some time
* Design had many parallel systems to save time.
* Components could have been used to easily change gates inside the system, but were included in a separate VHDL file to ease changing them.
* Design clock was randomly chosen based on worst case senarios seen, but there could have been a better way to calculate its period.
* Delay was added to the real value of the generator because the comparator needed 2 clocks to calculate the final output so to avoid collision in the analyzer.
* Variables could have been named better to express what they represent.
* Same variable names were used in the diagrams to make simulation results and codes more readable.

# References

1. <https://epgp.inflibnet.ac.inepgpdata/uploads/epgp_content/S000574EE/P001494/M015069/ET/1459849153et09.pdf> (8/5/2019)
2. Advanced Digital System Design slides of Dr. Abed Allatif (8/5/2019) .