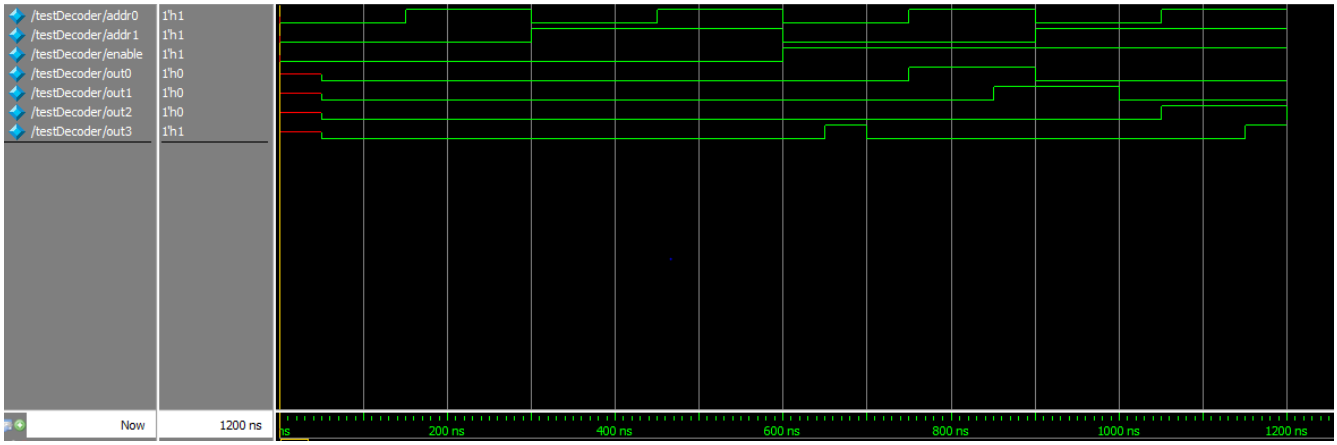
Comparch HW 2

Decoder:

```
# Top level modules:
#       behavioralDecoder
#       testDecoder
# End time: 17:49:58 on Sep 25,2015, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
# vsim
# Start time: 17:49:58 on Sep 25,2015
# Loading work.testDecoder
# Loading work.structuralDecoder
# En A0 A1| O0 O1 O2 O3 | Expected Output
# 0  0  0 |  0  0  0  0 | All false
# 0  1  0 |  0  0  0  0 | All false
# 0  0  1 |  0  0  0  0 | All false
# 0  1  1 |  0  0  0  0 | All false
# 1  0  0 |  1  0  0  0 | O0 Only
# 1  1  0 |  0  1  0  0 | O1 Only
# 1  0  1 |  0  0  1  0 | O2 Only
# 1  1  1 |  0  0  0  1 | O3 Only
# 0 ns
# 1260 ns
```

The decoder was supposed to output on the wire specified by the addresses, assuming that it was enabled. That is exactly the output shown here, exhaustively tested for correctness.



The timing diagram above shows the delay – the maximum possible delay was 150 ns.

Multiplexer:

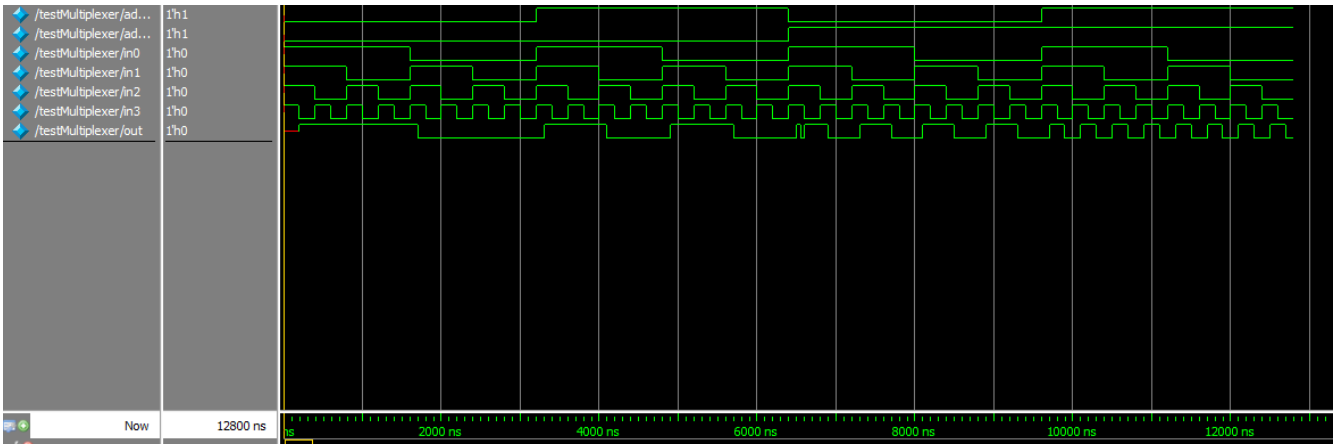
```
# Top level modules:
#       behavioralMultiplexer
#       testMultiplexer
# End time: 18:07:46 on Sep 25,2015, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
# vsim
# Start time: 18:07:46 on Sep 25,2015
# Loading work.testMultiplexer
# Loading work.structuralMultiplexer
# Testing address 00
# A1 A0 in0 in1 in2 in3 | Out | Expected Output
# 0  0  1   1   1   1   |  1  |  1
# 0  0  1   1   1   0   |  1  |  1
# 0  0  1   1   0   1   |  1  |  1
# 0  0  1   1   0   0   |  1  |  1
# 0  0  1   0   1   1   |  1  |  1
# 0  0  1   0   1   0   |  1  |  1
# 0  0  1   0   0   1   |  1  |  1
# 0  0  1   0   0   0   |  1  |  1
# 0  0  0   1   1   1   |  0  |  0
# 0  0  0   1   1   0   |  0  |  0
# 0  0  0   1   0   1   |  0  |  0
# 0  0  0   1   0   0   |  0  |  0
# 0  0  0   0   1   1   |  0  |  0
# 0  0  0   0   1   0   |  0  |  0
# 0  0  0   0   0   1   |  0  |  0
# 0  0  0   0   0   0   |  0  |  0
```

```
# Testing address 01
# A1 A0 in0 in1 in2 in3 | Out | Expected Output
# 0  1  1   1   1   1   |  1  |  1
# 0  1  1   1   1   0   |  1  |  1
# 0  1  1   1   0   1   |  1  |  1
# 0  1  1   1   0   0   |  1  |  1
# 0  1  1   0   1   1   |  0  |  0
# 0  1  1   0   1   0   |  0  |  0
# 0  1  1   0   0   1   |  0  |  0
# 0  1  1   0   0   0   |  0  |  0
# 0  1  0   1   1   1   |  1  |  1
# 0  1  0   1   1   0   |  1  |  1
# 0  1  0   1   0   1   |  1  |  1
# 0  1  0   1   0   0   |  1  |  1
# 0  1  0   0   1   1   |  0  |  0
# 0  1  0   0   1   0   |  0  |  0
# 0  1  0   0   0   1   |  0  |  0
# 0  1  0   0   0   0   |  0  |  0
```

```
# Testing address 10
# A1 A0 in0 in1 in2 in3 | Out | Expected Output
# 1  0  1   1   1   1   |  1  |  1
# 1  0  1   1   1   0   |  1  |  1
# 1  0  1   1   0   1   |  0  |  0
# 1  0  1   1   0   0   |  0  |  0
# 1  0  1   0   1   1   |  1  |  1
# 1  0  1   0   1   0   |  1  |  1
# 1  0  1   0   0   1   |  0  |  0
# 1  0  1   0   0   0   |  0  |  0
# 1  0  0   1   1   1   |  1  |  1
# 1  0  0   1   1   0   |  1  |  1
# 1  0  0   1   0   1   |  0  |  0
# 1  0  0   1   0   0   |  0  |  0
# 1  0  0   0   1   1   |  1  |  1
# 1  0  0   0   1   0   |  1  |  1
# 1  0  0   0   0   1   |  0  |  0
# 1  0  0   0   0   0   |  0  |  0
```

```
# Testing address 11
# A1 A0 in0 in1 in2 in3 | Out | Expected Output
# 1  1  1   1   1   1   |  1  |  1
# 1  1  1   1   1   0   |  0  |  0
# 1  1  1   1   0   1   |  1  |  1
# 1  1  1   1   0   0   |  0  |  0
# 1  1  1   0   1   1   |  1  |  1
# 1  1  1   0   1   0   |  0  |  0
# 1  1  1   0   0   1   |  1  |  1
# 1  1  1   0   0   0   |  0  |  0
# 1  1  0   1   1   1   |  1  |  1
# 1  1  0   1   1   0   |  0  |  0
# 1  1  0   1   0   1   |  1  |  1
# 1  1  0   1   0   0   |  0  |  0
# 1  1  0   0   1   1   |  1  |  1
# 1  1  0   0   1   0   |  0  |  0
# 1  1  0   0   0   1   |  1  |  1
# 1  1  0   0   0   0   |  0  |  0
# 0 ns
# 13440 ns
```

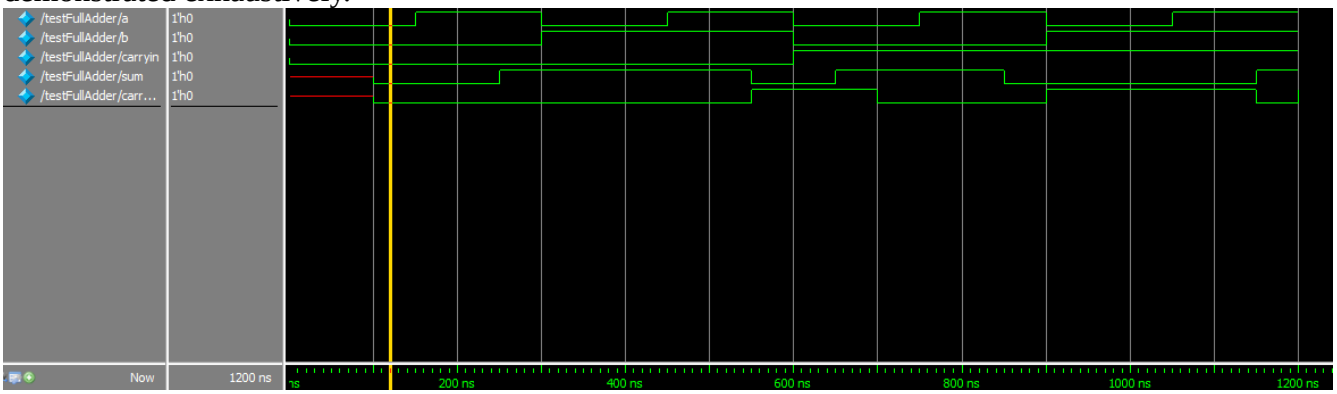
The Multiplexer takes in four inputs, with a two input address. If the input corresponding to the address is 1, then the output is 1. Again, that is what is observed here, proved exhaustively with all 64 possible input combinations. The timing is a little trickier to see here, (too many input changes) but as it shows in the timing diagram, the maximum delay was 200 ns.



Adder:

```
# Top level modules:
#       behavioralFullAdder
#       testFullAdder
# End time: 18:25:11 on Sep 25,2015, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
# vsim
# Start time: 18:25:11 on Sep 25,2015
# Loading work.testFullAdder
# Loading work.structuralFullAdder
# a b ci | sum co | Expected Output
# 0 0 0  |  0  0  | 0 0
# 1 0 0  |  1  0  | 1 0
# 0 1 0  |  1  0  | 1 0
# 1 1 0  |  0  1  | 0 1
# 0 0 1  |  1  0  | 1 0
# 1 0 1  |  0  1  | 0 1
# 0 1 1  |  0  1  | 0 1
# 1 1 1  |  1  1  | 1 1
# 0 ns
# 1260 ns
```

The full adder (for one bit) takes in two bits to add, as well as the carry-in bit, in case it is part of a multi-bit addition. It adds them all to get 00, 01, 10, or 11. In this case this functionality was demonstrated exhaustively.



This is the timing diagram for the adder, where the maximum delay is 150 ns.