

EE224 Assignment 1 Booth Multiplier

Pranjal Jain

1 Components

- Hardware to implement 2's complement
- 4 Multiplexers taking 4 inputs, giving one output
- 4 Adders taking 16 bit inputs and giving a 16 bit output
- 3 Left shifters that shift left by 2 bits

2 Algorithm

The Booth encoded multiplication speeds up multiplication by implementing Radix-4 multiplication of two binary numbers. Rather than calculating partial products of multiplicand with one bit of multiplier at a time, we multiply multiplicand with 2 bits at a time.

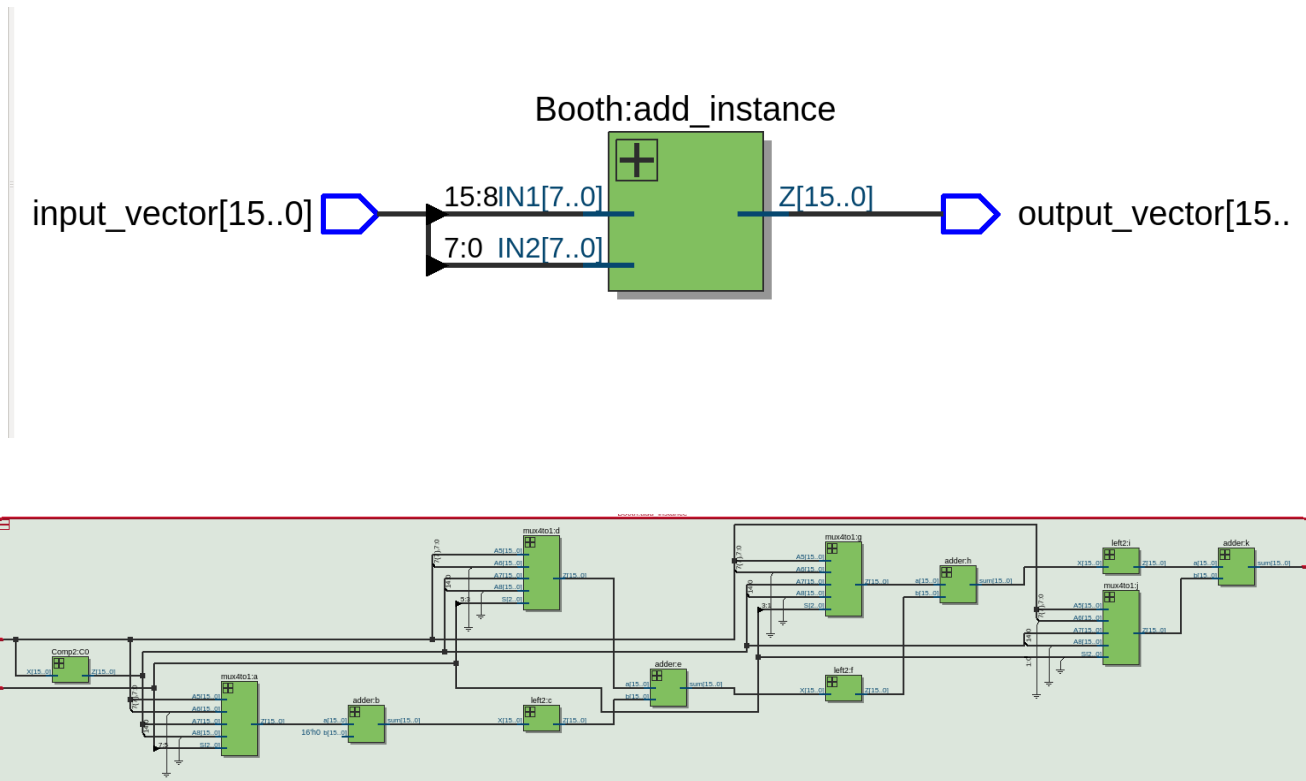
Since $2^2 = 4$, the multiplication is Radix-4, although the partial products are finally written in binary.

Sign extension of the multiplicand is done, by extending 8 bits to 16 bits. This is done by observing the most significant bit, and adding 8 bits which are the same as MSB to the left of the MSB.

The MUX has a three selection inputs, so that it can check which of the 8 combinations of 3 bits is used, the two most significant bits corresponding to the current bits of the multiplier being considered, and the LSB corresponding to the previous bit, so it may be appropriate to call it an Encoder, rather than a multiplier.

3 Circuit

The RTL viewer under the netlist viewer subsection gives us the following circuits.



4 Waveforms

On RTL simulation, the following waveforms were obtained.

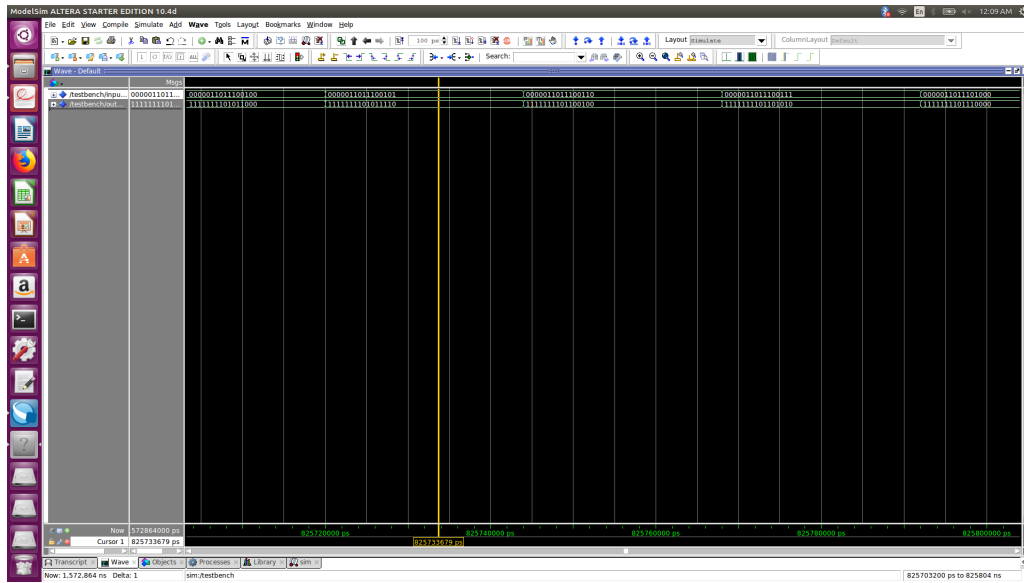


Figure 1: Zoomed in for specific testcases

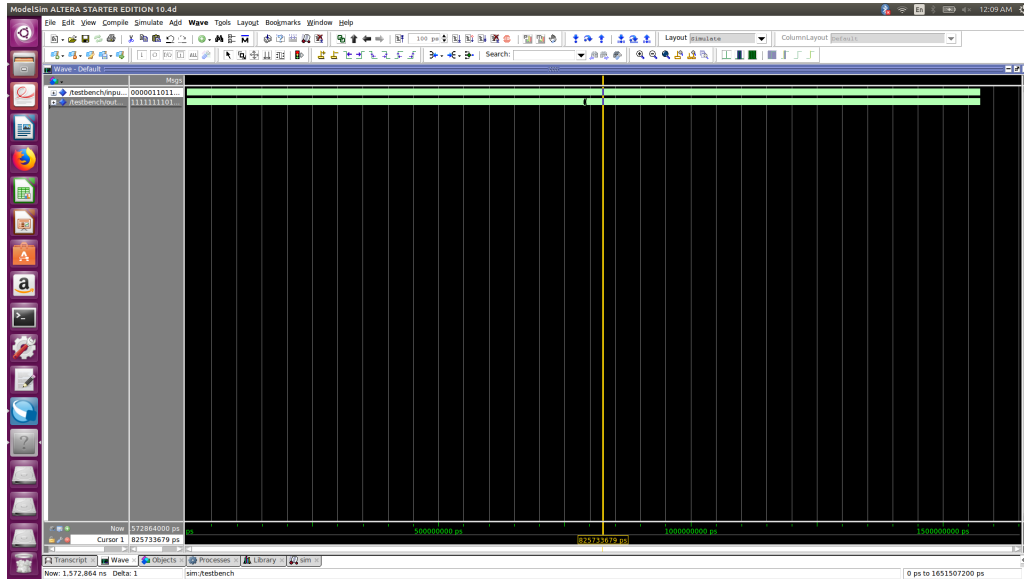
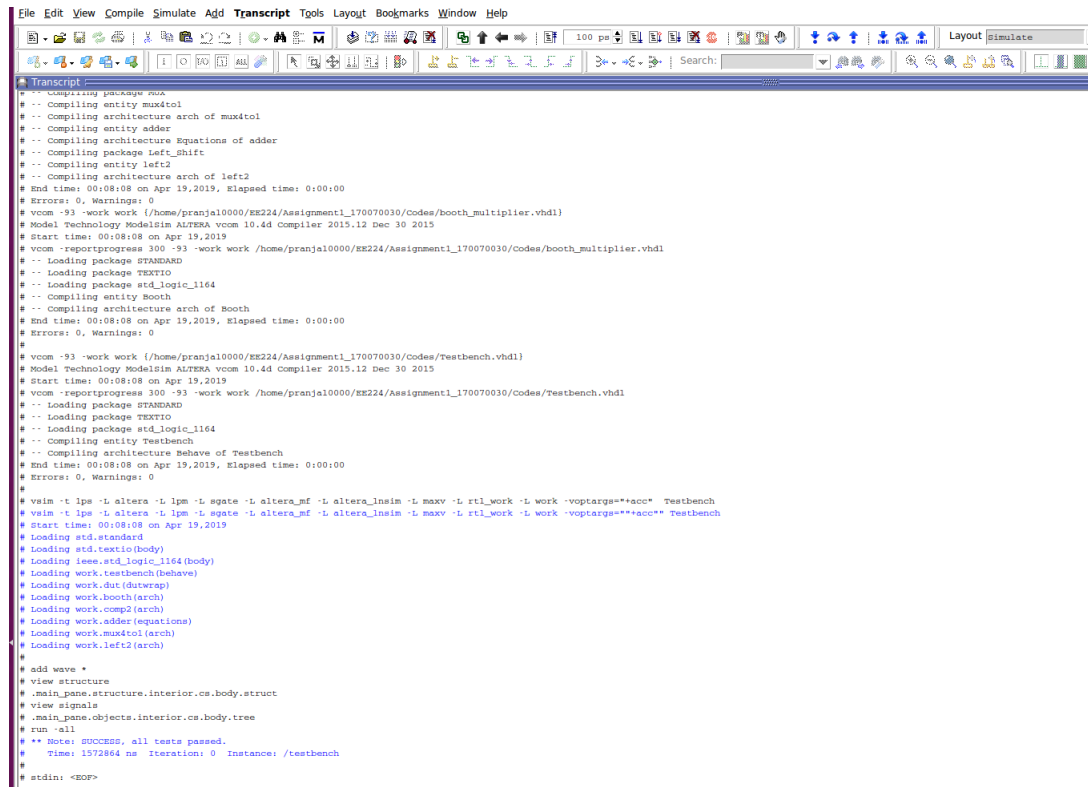


Figure 2: The wave for all testcases

5 Success

The transcript of the RTL simulation is shown below, all test cases were successful.

The image shows a screenshot of a software interface with a menu bar (File, Edit, View, Compile, Simulate, Add, Transcript, Tools, Layout, Bookmarks, Window, Help) and a toolbar. The 'Transcript' window is active, displaying the following text:

```
# -- Compiling package mux
# -- Compiling entity mux2to1
# -- Compiling architecture arch of mux2to1
# -- Compiling entity adder
# -- Compiling architecture Equations of adder
# -- Compiling package Left_Shift
# -- Compiling entity left2
# -- Compiling architecture arch of left2
# End time: 00:08:08 on Apr 19, 2019, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
# vcom -93 -work work [/home/pranjal0000/EE224/Assignment1_170070030/Codes/booth_multiplier.vhdl]
# Model Technology ModelSim ALTERA vcom 10.4d Compiler 2015.12 Dec 30 2015
# Start time: 00:08:08 on Apr 19, 2019
# vcom -reportprogress 300 -93 -work work [/home/pranjal0000/EE224/Assignment1_170070030/Codes/booth_multiplier.vhdl]
# -- Loading package STANDARD
# -- Loading package TEXTIO
# -- Loading package std_logic_1164
# -- Compiling entity Booth
# -- Compiling architecture arch of Booth
# End time: 00:08:08 on Apr 19, 2019, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
#
# vcom -93 -work work [/home/pranjal0000/EE224/Assignment1_170070030/Codes/Testbench.vhdl]
# Model Technology ModelSim ALTERA vcom 10.4d Compiler 2015.12 Dec 30 2015
# Start time: 00:08:08 on Apr 19, 2019
# vcom -reportprogress 300 -93 -work work [/home/pranjal0000/EE224/Assignment1_170070030/Codes/Testbench.vhdl]
# -- Loading package STANDARD
# -- Loading package TEXTIO
# -- Loading package std_logic_1164
# -- Compiling entity Testbench
# -- Compiling architecture behave of Testbench
# End time: 00:08:08 on Apr 19, 2019, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
#
# vsim -t lps -L altera -l lpm -L sgate -L altera_mf -l altera_insim -L maxv -L rtl_work -L work -voptargs="+acc" Testbench
# vsim -t lps -L altera -l lpm -L sgate -L altera_mf -l altera_insim -L maxv -L rtl_work -L work -voptargs="+acc" Testbench
# Start time: 00:08:08 on Apr 19, 2019
# Loading std.standard
# Loading std.textio(body)
# Loading ieee.std_logic_1164(body)
# Loading work.testbench(behave)
# Loading work.dut(dutwrap)
# Loading work.booth(arch)
# Loading work.comp2(arch)
# Loading work.adder(equations)
# Loading work.mux2to1(arch)
# Loading work.left2(arch)
#
# add wave *
# view structure
# .main_pane.structure.interior.cs.body.struct
# view signals
# .main_pane.objects.interior.cs.body.tree
# run -all
# ** Note: SUCCESS, all tests passed.
# Time: 1572864 ns Iteration: 0 Instance: /testbench
#
# stdin: <EOF>
```

Figure 3: RTL transcript

```
# Loading work.left2(arch)
#
# add wave *
# view structure
# .main_pane.structure.interior.cs.body.struct
# view signals
# .main_pane.objects.interior.cs.body.tree
# run -all
# ** Note: SUCCESS, all tests passed.
# Time: 1572864 ns Iteration: 0 Instance: /testbench
#
# stdin: <EOF>
```

Figure 4: RTL success