EE 789: Algorithmic Design of Digital Systems
Assignment 2

Jayesh Choudhary 170070038

1 Division using repetitive subtraction

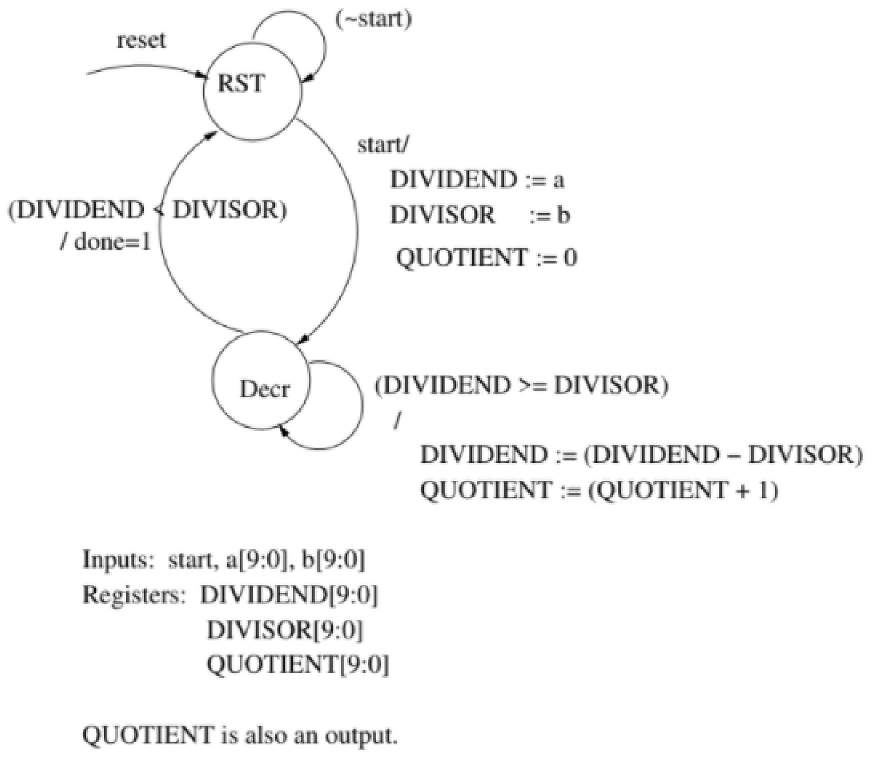


Figure 1: RTL machine for a simple divider

1.1 Discrete Time equations

There are 2 states : RST (0) and DECR (1), C is the boolean condition (a $\geq$ b), R is Reset, S is Start, TS is the current state, NS is the next state, D is Done, Q_n is Quotient

(TS)	R	S	C	NS	D	Q_n
X	1	X	X	0	0	0
0	0	0	X	0	0	0
0	0	1	X	1	0	0
1	0	X	1	1	0	Q_{n-1}+1
1	0	X	0	0	1	Q_{n-1}

Table 1: Transitions

D = $(\overline{R}).(\overline{C}).(TS)$

$Q_n = Q_{n-1}$.(TS) + 1.(C)

NS = **RST**.(R + $\overline{R}.((\overline{TS}).\overline{S} + (TS).(\overline{C})) +$ **DECR**.$(\overline{R}((TS).(C) + (\overline{TS}).(S)))$

or NS = $\overline{R}((TS).(C) + (\overline{TS}).(S)$

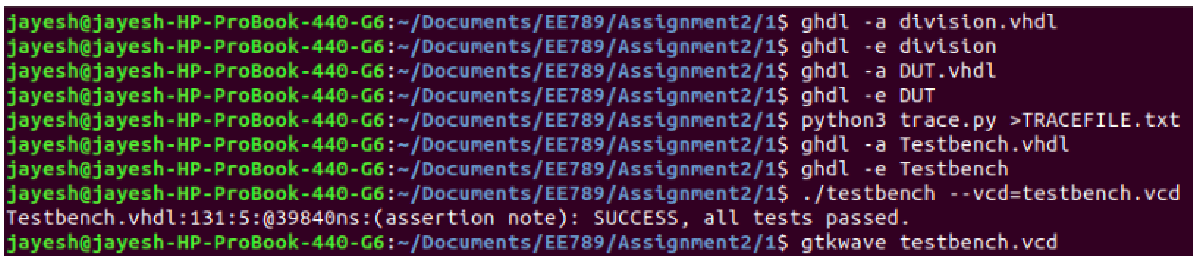
1.2 RTL machine in VHDL

VDHL files are provided along with report

1.3 Testbench

A python script was created to generate TRACEFILE.txt and all the cases to be tested are mentioned in that script itself. TRACEFILE was carefully created considering the extra clock cycles.

1.4 VHDL Simulation



```
jayesh@jayesh-HP-ProBook-440-G6:~/Documents/EE789/Assignment2/1$ ghdl -a division.vhdl
jayesh@jayesh-HP-ProBook-440-G6:~/Documents/EE789/Assignment2/1$ ghdl -e division
jayesh@jayesh-HP-ProBook-440-G6:~/Documents/EE789/Assignment2/1$ ghdl -a DUT.vhdl
jayesh@jayesh-HP-ProBook-440-G6:~/Documents/EE789/Assignment2/1$ ghdl -e DUT
jayesh@jayesh-HP-ProBook-440-G6:~/Documents/EE789/Assignment2/1$ python3 trace.py >TRACEFILE.txt
jayesh@jayesh-HP-ProBook-440-G6:~/Documents/EE789/Assignment2/1$ ghdl -a Testbench.vhdl
jayesh@jayesh-HP-ProBook-440-G6:~/Documents/EE789/Assignment2/1$ ghdl -e Testbench
jayesh@jayesh-HP-ProBook-440-G6:~/Documents/EE789/Assignment2/1$ ./testbench --vcd=testbench.vcd
Testbench.vhdl:131:5:@39840ns:(assertion note): SUCCESS, all tests passed.
jayesh@jayesh-HP-ProBook-440-G6:~/Documents/EE789/Assignment2/1$ gtkwave testbench.vcd
```

Figure 2: Terminal screen for the simulation

As we can see **all the test cases were passed** for the automatically generated tracefile using code.

In the waveforms below:

- The first line represents the clock
- The second line the dividend **a** whose value keeps on decreasing by a factor of b
- The third line represents the divisor **b**
- The fourth line represents the the quotient which keeps on increasing by 1
- The next 11 consecutive lines are of output

As we can see the 11^{th} bit of output goes to 1 when we reach the end condition.

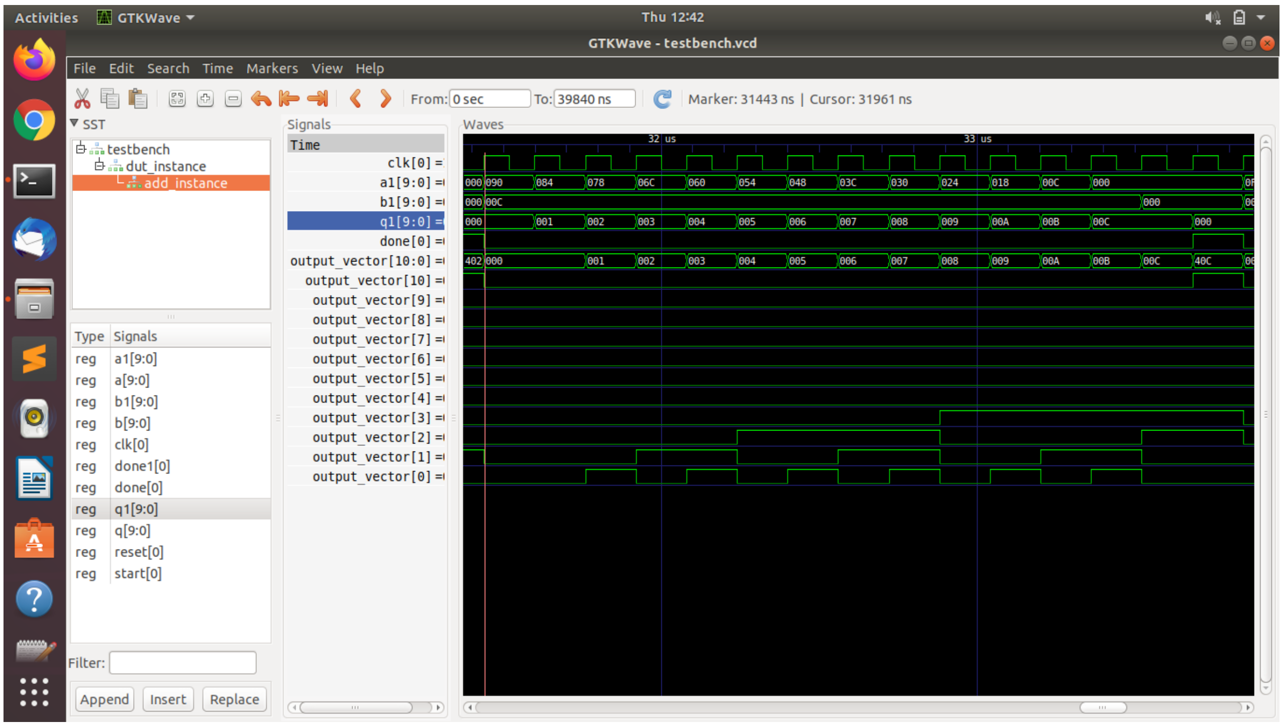


Figure 3: Waveform for a particular case

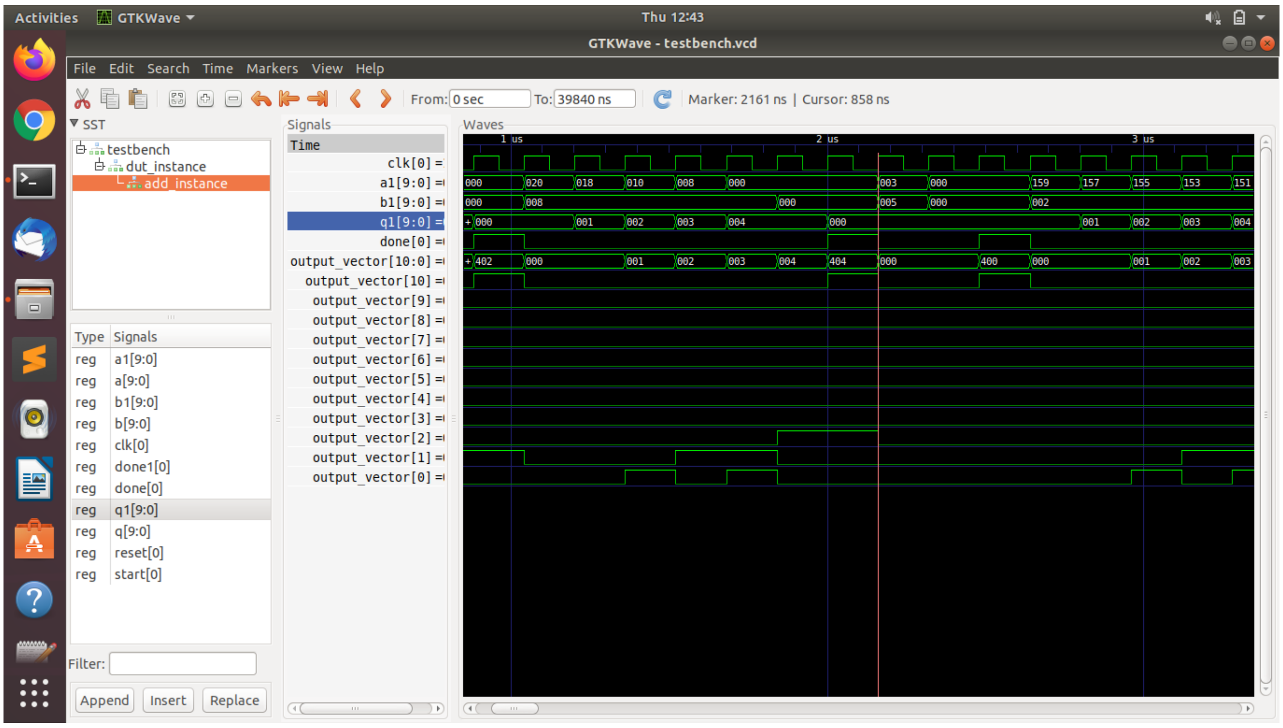


Figure 4: Waveform for another case

Here we have 2 other test cases one normal and another emphasizing what happens when divisor is bigger than the dividend. As expected, the quotient remains 0 and 1 cycle after **Start** is given (the next cycle after DONE bit is 1), we get the DONE bit as 1.

Total number of clock cycles required = quotient + 3
Extra cycles are required due to the fact that we have to use another set of registers and then transfer the value to the output

2 8-bit Shift and Add Multiplier

2.1 Algorithm

We are multiplying a and b.

$$p = \Sigma 2^k * b_k * a$$

This could be done by:

- S0[15:0] = $(0 + (2^8 * b_0 * \text{a})) * 2^{-1}$
- S1[15:0] = $(\text{S0} + (2^8 * b_1 * \text{a})) * 2^{-1}$
- S2[15:0] = $(\text{S1} + (2^8 * b_2 * \text{a})) * 2^{-1}$
- S3[15:0] = $(\text{S2} + (2^8 * b_3 * \text{a})) * 2^{-1}$
- S4[15:0] = $(\text{S3} + (2^8 * b_4 * \text{a})) * 2^{-1}$
- S5[15:0] = $(\text{S4} + (2^8 * b_5 * \text{a})) * 2^{-1}$
- S6[15:0] = $(\text{S5} + (2^8 * b_6 * \text{a})) * 2^{-1}$
- S7[15:0] = $(\text{S6} + (2^8 * b_7 * \text{a})) * 2^{-1}$

These are the shifts that we would use as described in the FSM below:

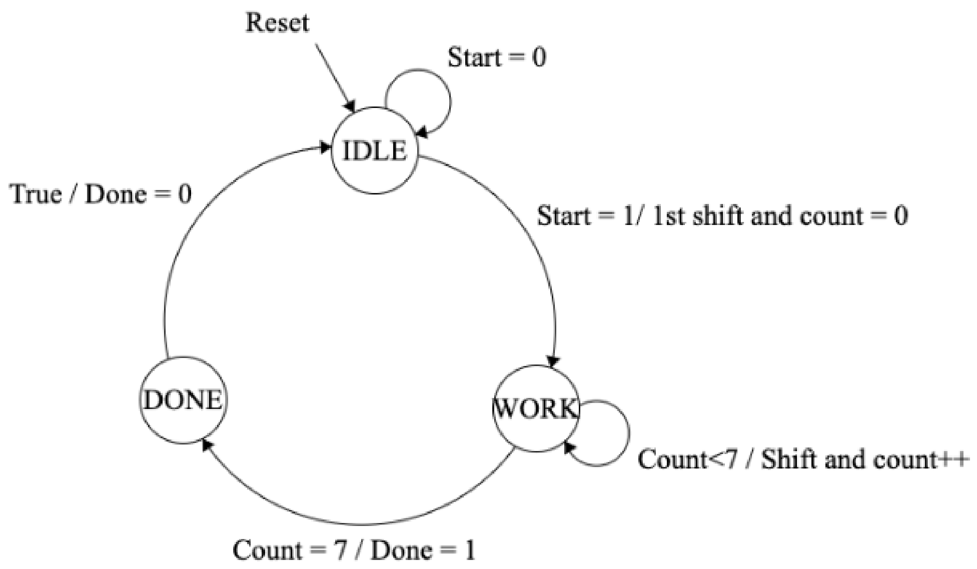

Reset
Start = 0
IDLE
True / Done = 0
Start = 1/ 1st shift and count = 0
DONE
WORK
Count<7 / Shift and count++
Count = 7 / Done = 1

Figure 5: FSM for shift and add multiplier

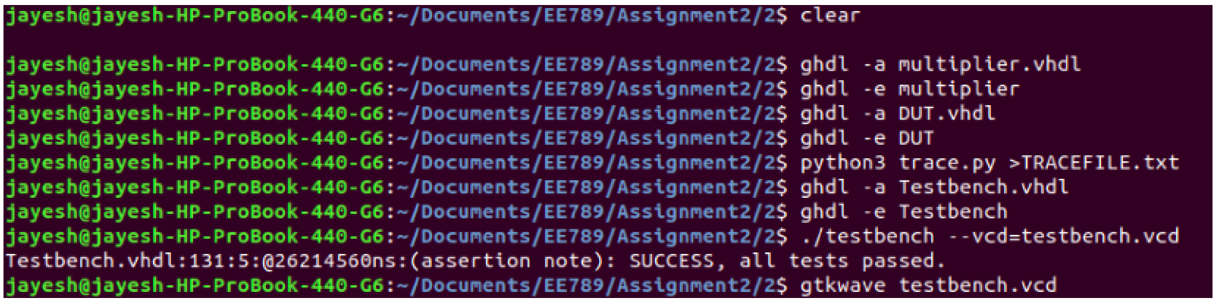
2.2 VHDL Implementation

VHDL files are provided with this report

2.3 Testbench

A python script was created to generate TRACEFILE.txt with all possible cases. TRACE-FILE was carefully created considering the extra clock cycles.

2.4 Simulation using GHDL simulator



```
jayesh@jayesh-HP-ProBook-440-G6:~/Documents/EE789/Assignment2/2$ clear

jayesh@jayesh-HP-ProBook-440-G6:~/Documents/EE789/Assignment2/2$ ghdl -a multiplier.vhdl
jayesh@jayesh-HP-ProBook-440-G6:~/Documents/EE789/Assignment2/2$ ghdl -e multiplier
jayesh@jayesh-HP-ProBook-440-G6:~/Documents/EE789/Assignment2/2$ ghdl -a DUT.vhdl
jayesh@jayesh-HP-ProBook-440-G6:~/Documents/EE789/Assignment2/2$ ghdl -e DUT
jayesh@jayesh-HP-ProBook-440-G6:~/Documents/EE789/Assignment2/2$ python3 trace.py >TRACEFILE.txt
jayesh@jayesh-HP-ProBook-440-G6:~/Documents/EE789/Assignment2/2$ ghdl -a Testbench.vhdl
jayesh@jayesh-HP-ProBook-440-G6:~/Documents/EE789/Assignment2/2$ ghdl -e Testbench
jayesh@jayesh-HP-ProBook-440-G6:~/Documents/EE789/Assignment2/2$ ./testbench --vcd=testbench.vcd
Testbench.vhdl:131:5:@26214560ns:(assertion note): SUCCESS, all tests passed.
jayesh@jayesh-HP-ProBook-440-G6:~/Documents/EE789/Assignment2/2$ gtkwave testbench.vcd
```

Figure 6: Terminal screen for the simulation

As we can see **all the test cases were passed** for the automatically generated tracefile using code.

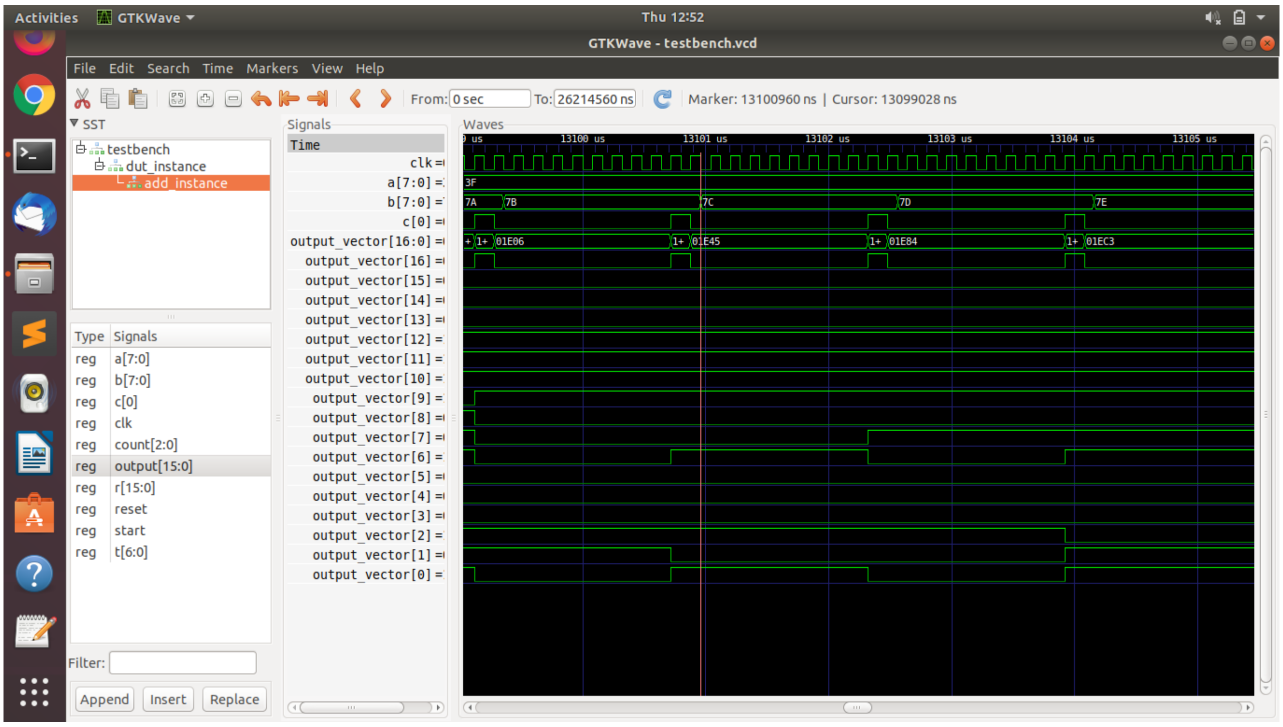


Figure 7: Waveform for the multiplier

Here:

- The first line represents the clock
- The second and third line represent the input **a** and **b**
- The fourth line represents the the done register
- The next 17 consecutive lines are of output

As we can see the 17^{th} bit of output goes to 1 when we reach the end condition.

Total number of clock cycles required = 10
8 for shift-and-adding and 2 extra cock cycles considering the reasning mentioned in previous case