University of Colorado Boulder

ECEN 5863: Programmable Logic Design

Homework E

Date: 19th October 2017

Richard Noronha

Richard.Noronha@colorado.edu

Q1) FIFO

Implementation:

1) The FIFO is designed as 8x9bit memory device with the signals as mentioned
2) In the design, the read pointer increments automatically when a read is made and also increments when the read increment(rdinc) signal is applied. In both scenarios, the read pointer increments by 1.
3) The write pointer also increments like the read pointer. When data is written to a memory location, the write pointer moves ahead by 1. It can also be moved ahead by 1 by setting the write pointer increment(wrinc) signal to high.
4) The ReadPtr is reset to 0 by using the RdPtrClr signal
5) The WritePtr is reset to 0 by using the WrPtrClr signal
6) To read the entire contents of the buffer, reset the readptr to 0 and then keep the read enable signal high for greater than 7 clock cycles.
7) When data is not being read, the output on the data out is High Impedance(Z).
8) This is the case even when data is being written to the device

Verilog code: Provided in submission folder

Screenshots:

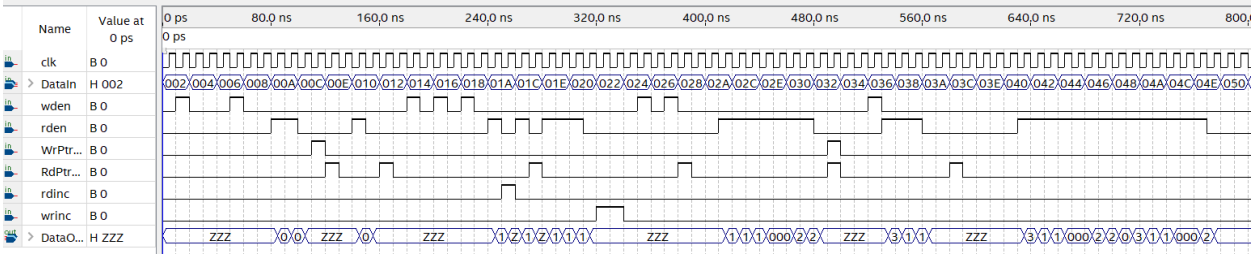

Fig 1. Entire implementation

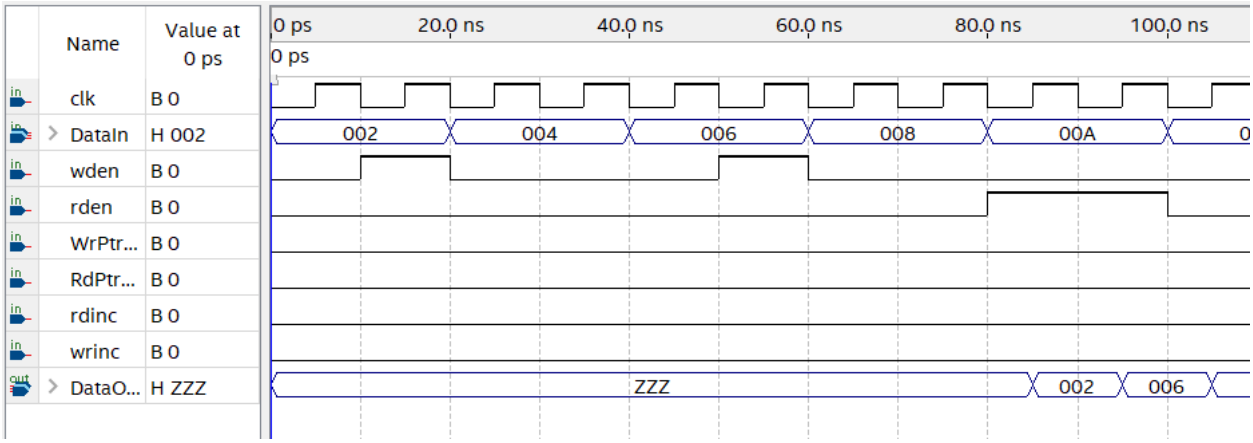

Fig 2. Basic write and read

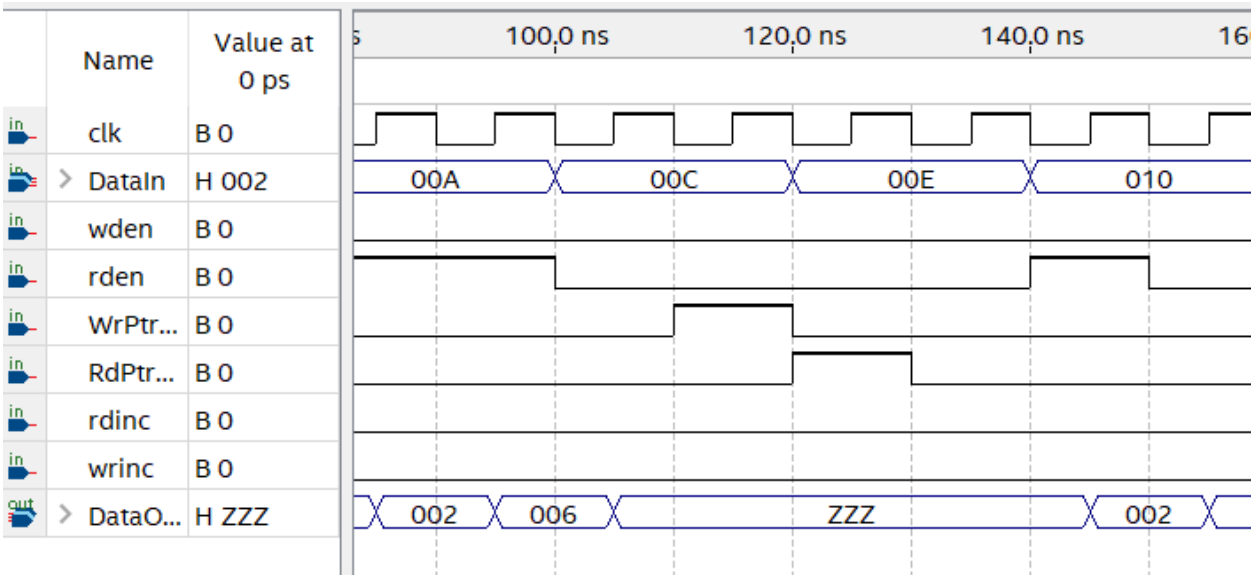

Fig 3. Clearing the read pointer and write pointer. The read pointer again points to location 0

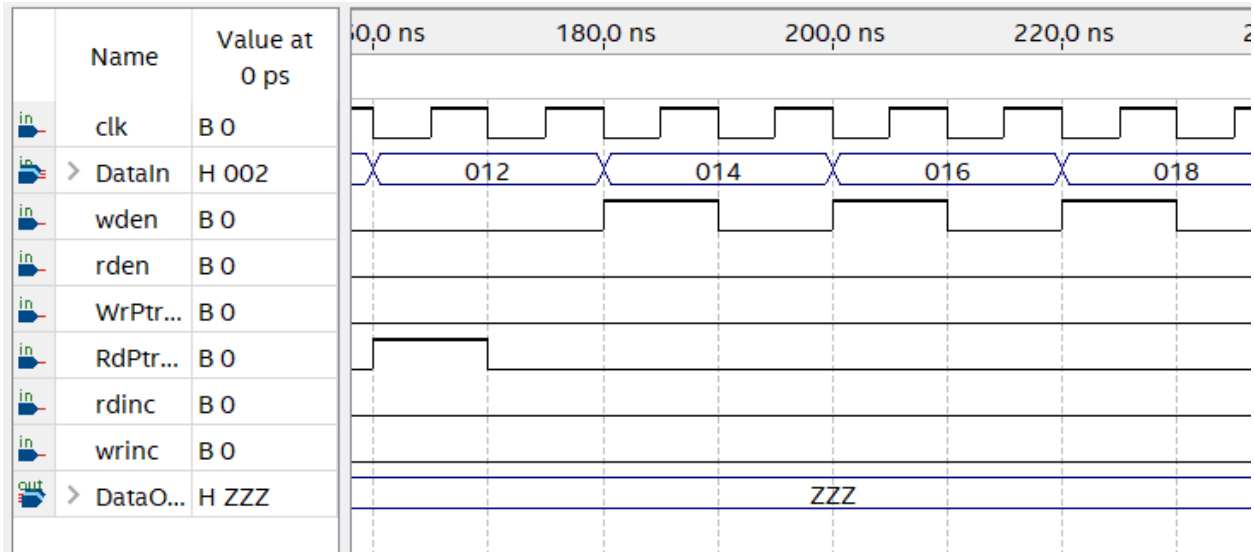

Fig 4. Again reset the read pointer. Write 3 data to location 0, 1, 2

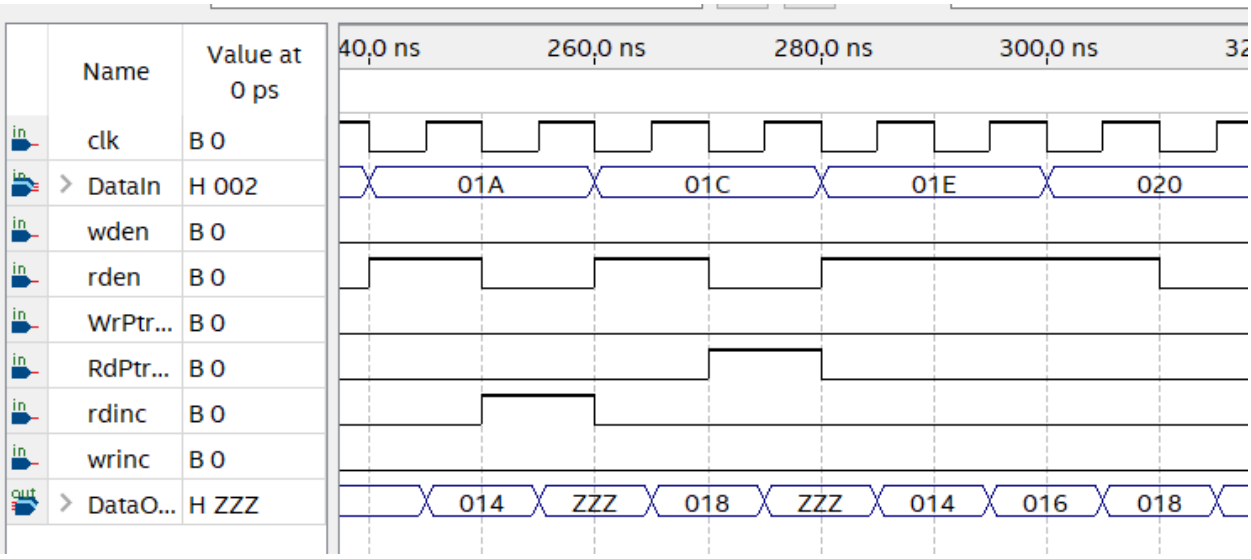

Fig 5. Read location 0, increment the read pointer(therefore Z), read location 2. Also note the data at location 1 is still present.

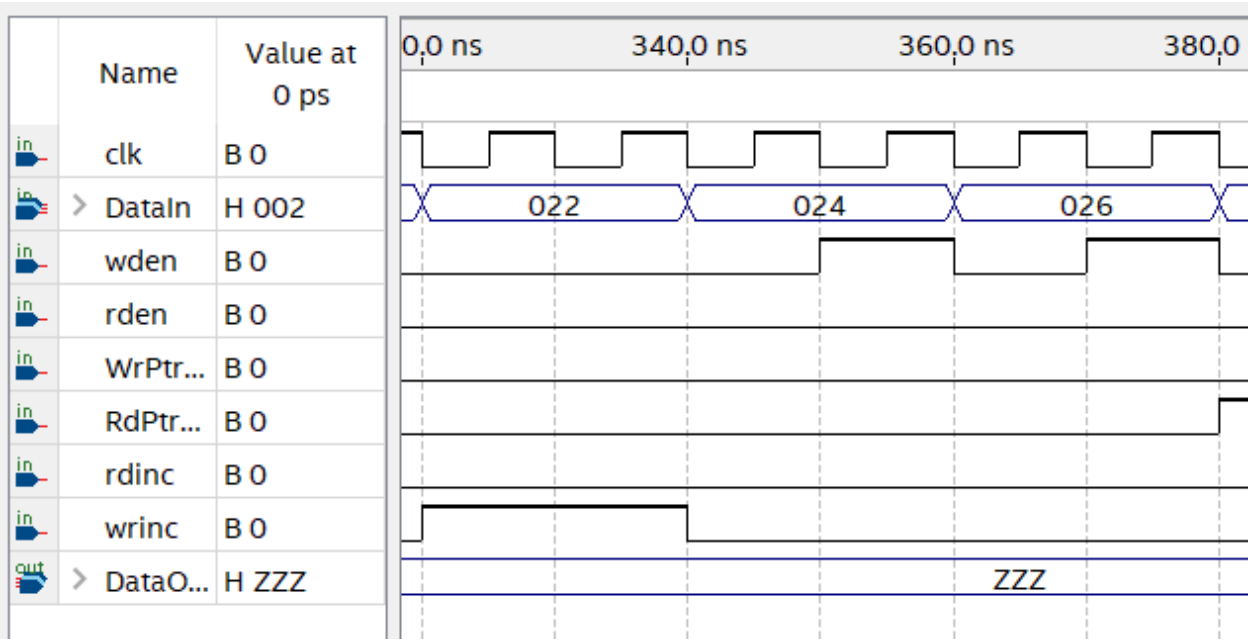

Fig 6. Increment the read pointer by 2(zeros will be present at those locations). Then write to the two new locations

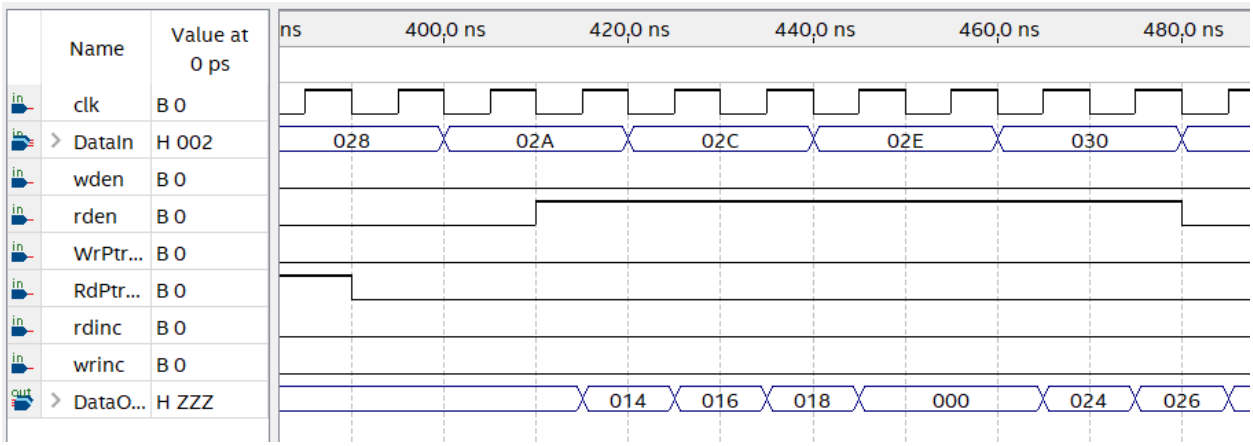

Fig 7. Read all the values in the FIF0. Notice the 0s because of the write inc command

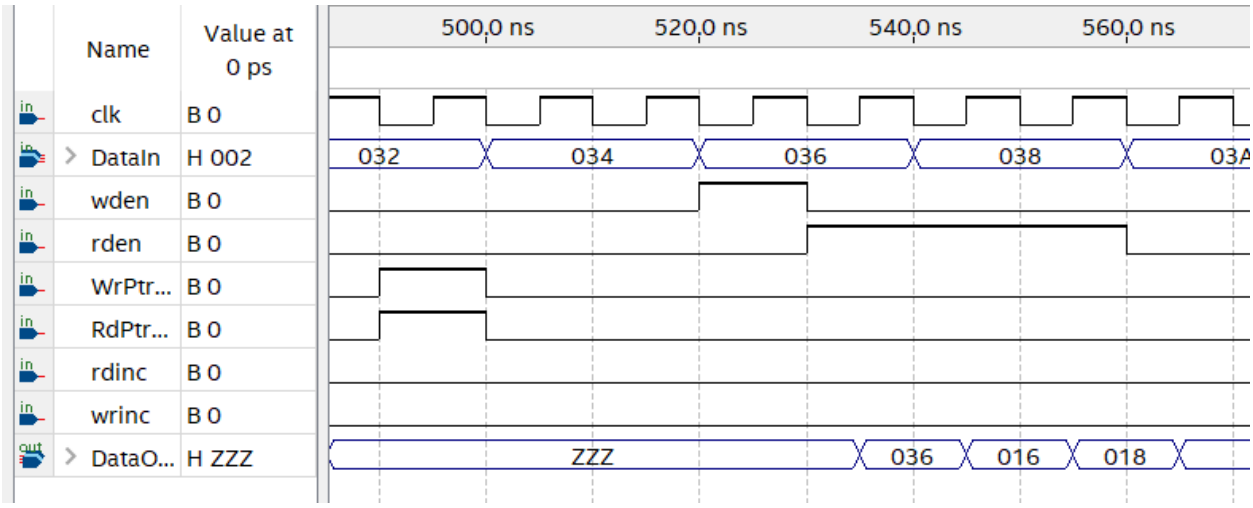

Fig 8. Reset write and read pointers. Write 0x36 to location 1. Rest data is still in FIFO

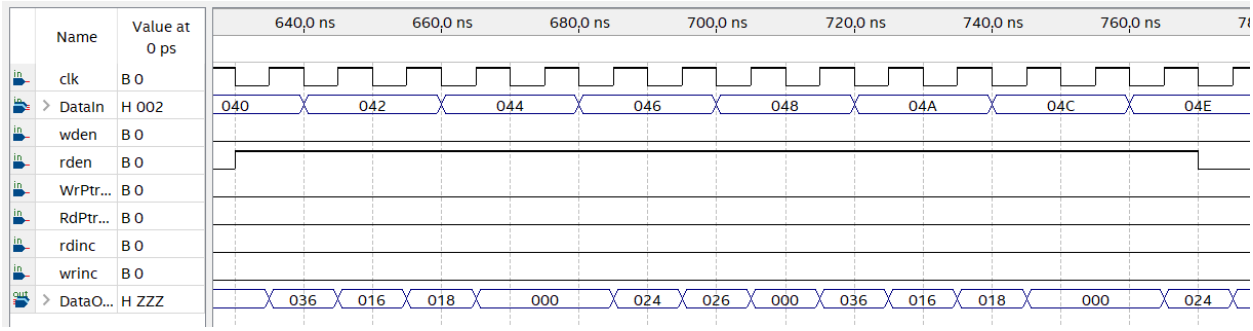

Fig 9. Read entire buffer till it overflows. The data repeats once the max value is reached

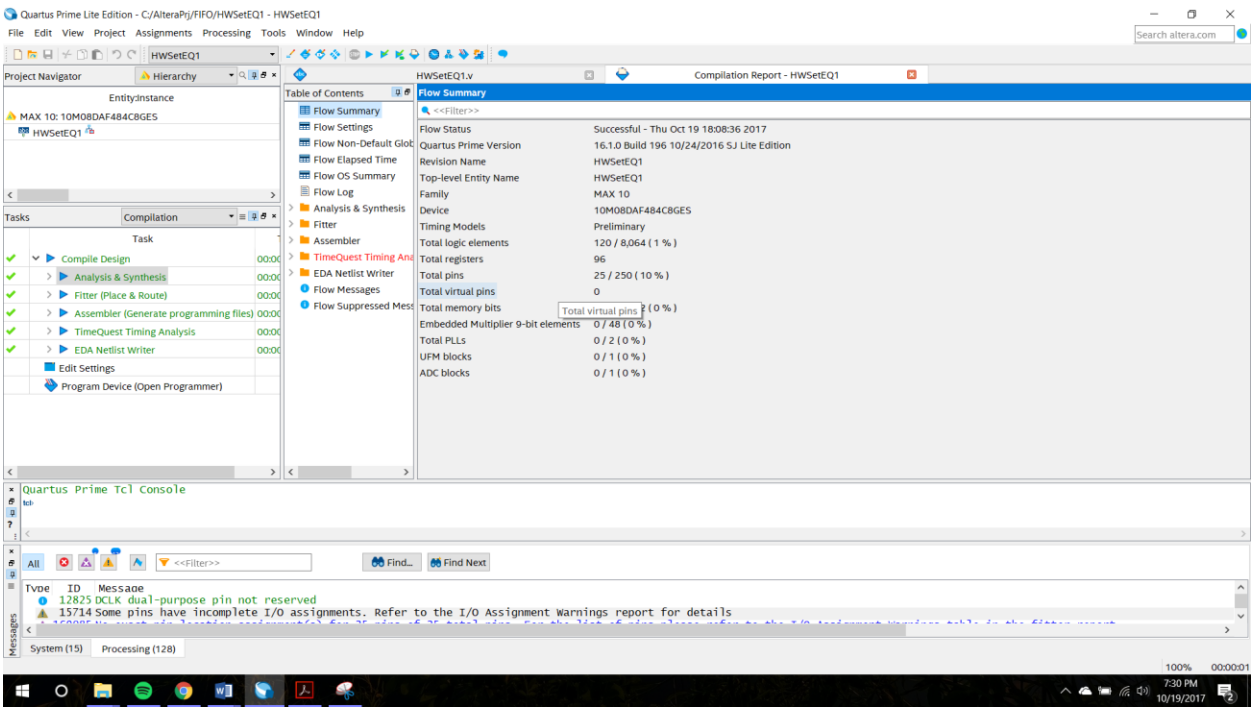

Fig 10. Compilation Report

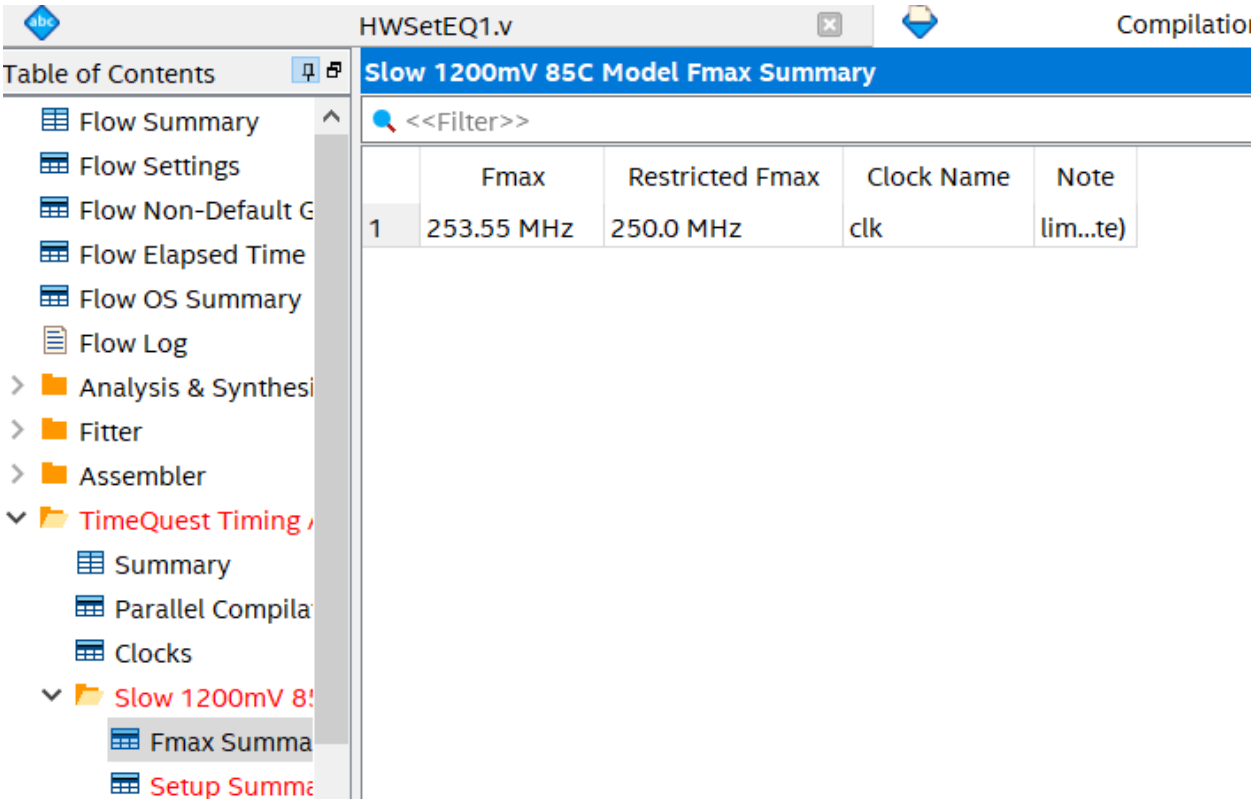

Fig 11. Fmax is 253.55MHz

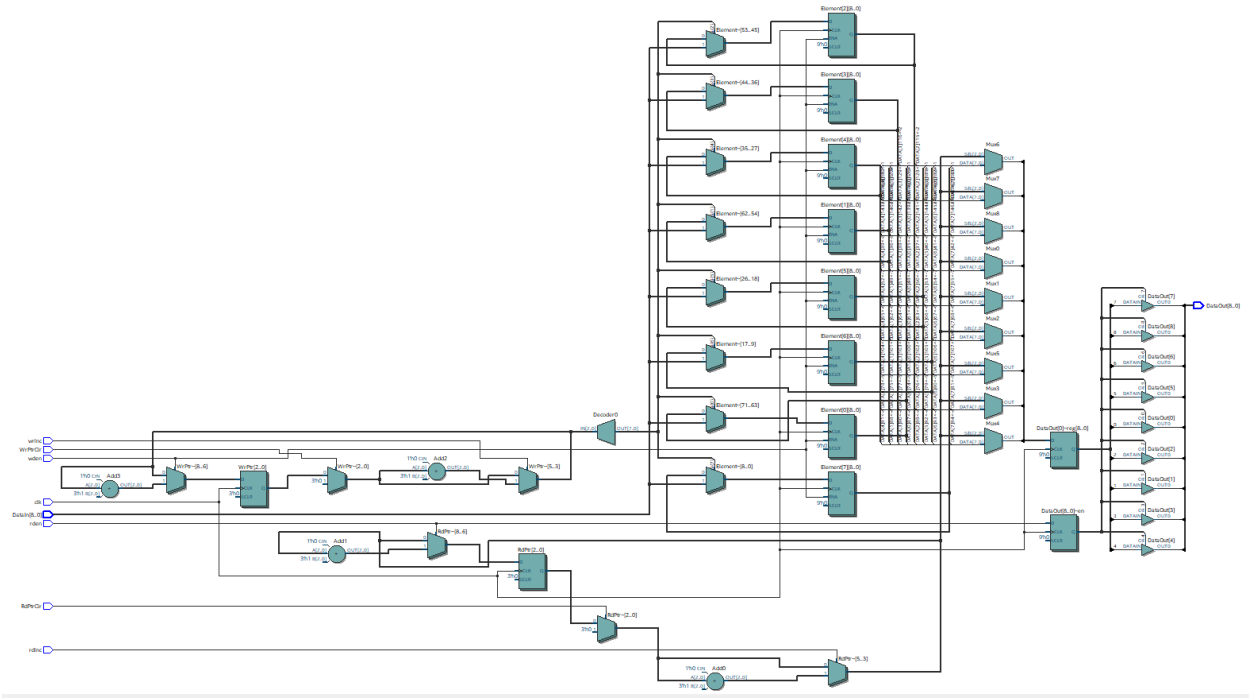

Fig 12. RTL for the FIFO

Q2)

2. Write a Verilog model for a 32-bit, arithmetic logic unit (ALU). Verify correct operation with a simulation using the Altera CAD tools. A and B are 32-bit inputs to the ALU, and Y is the output. A shift operation follows the arithmetic and logical operation. The opcode controls ALU functions as follows:

Opcode	Operation	Function
000XX	ALU_OUT <= A	Pass A
001XX	ALU_OUT <= A + B	Add
010XX	ALU_OUT <= A-B	Subtract
011XX	ALU_OUT <= A AND B	Logical AND
100XX	ALU_OUT <= A OR B	Logical OR
101XX	ALU_OUT <= A + 1	Increment A
110XX	ALU_OUT <= A-1	Decrement A
111XX	ALU_OUT <= B	Pass B
XXX00	Y <= ALU_OUT	Pass ALU_OUT
XXX01	Y<= SHL(ALU_OUT)	Shift Left
XXX10	Y<=SHR(ALU_OUT)	Shift Right (unsigned-zero fill)
XXX11	Y<= 0	Pass 0's

Implementation

1) The ALU project is created
2) Based on what operation is needed, the wires from the Operation bus is sliced
3) For the shifting operations a clock is used
4) To obtain Fmax another clock is used
5) The outputs can be observed after functional analysis

Screenshots:

Outputs can be seen on Y_temp as well as Y(in the next clock edge)

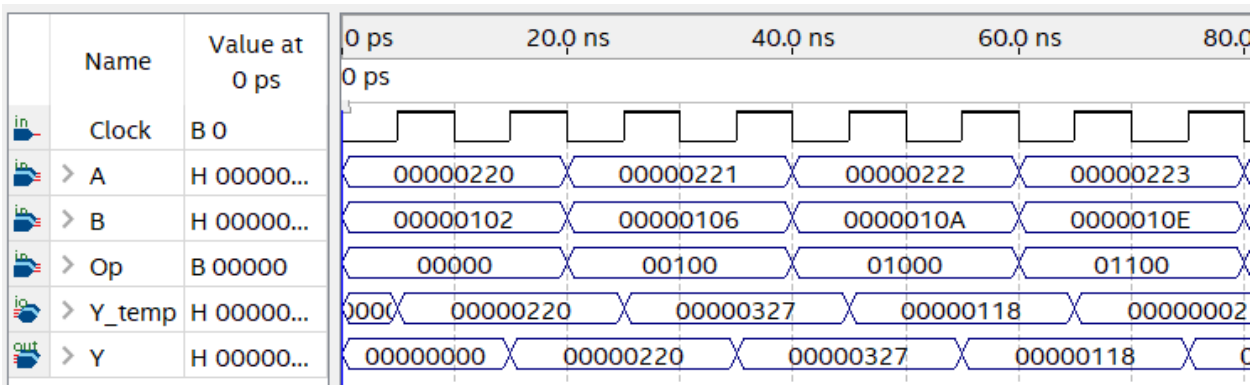

Fig 1. Pass A, Addition, Subtraction

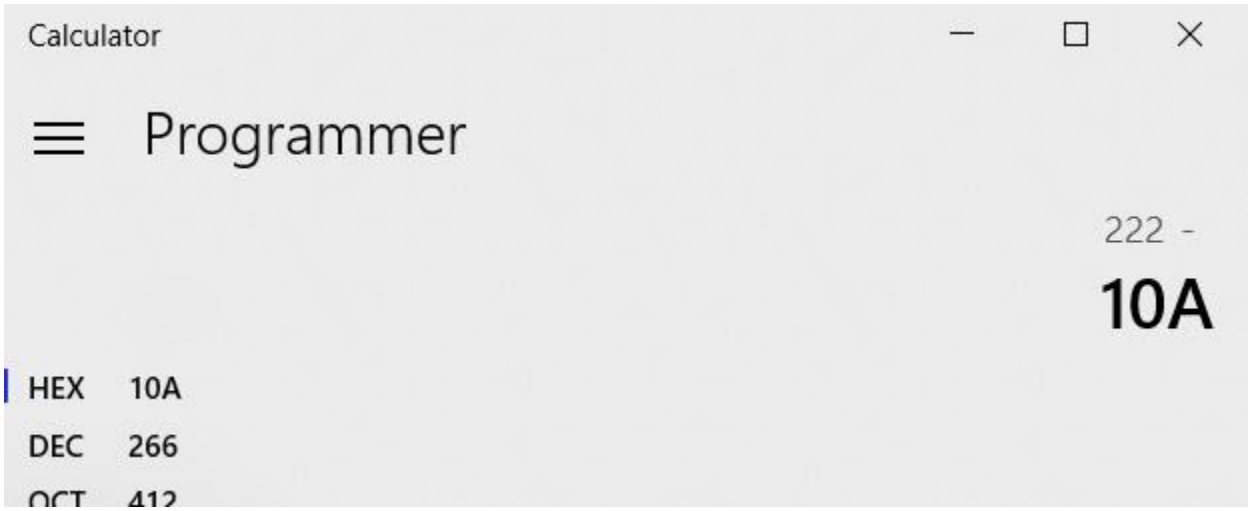

Fig 2. Proof of subtraction

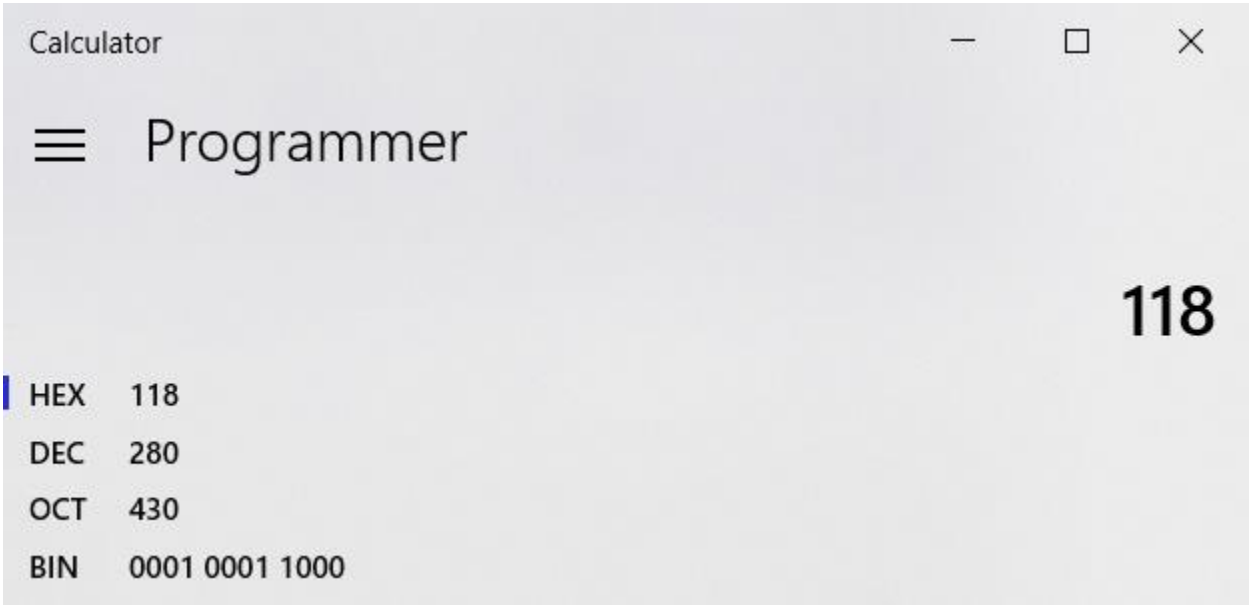

Fig 3. Answer

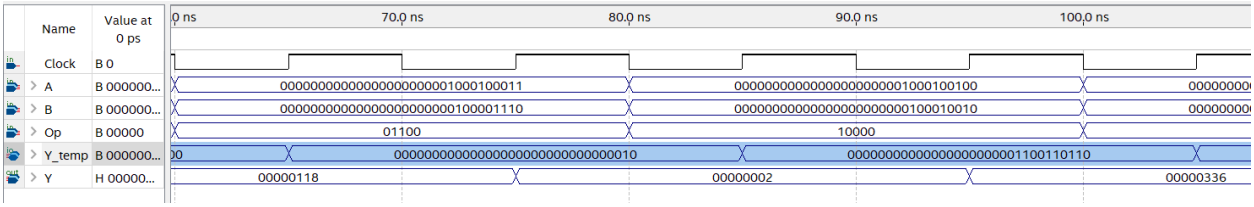

Fig 4. AND operation and OR operation(converted to binary for convenience

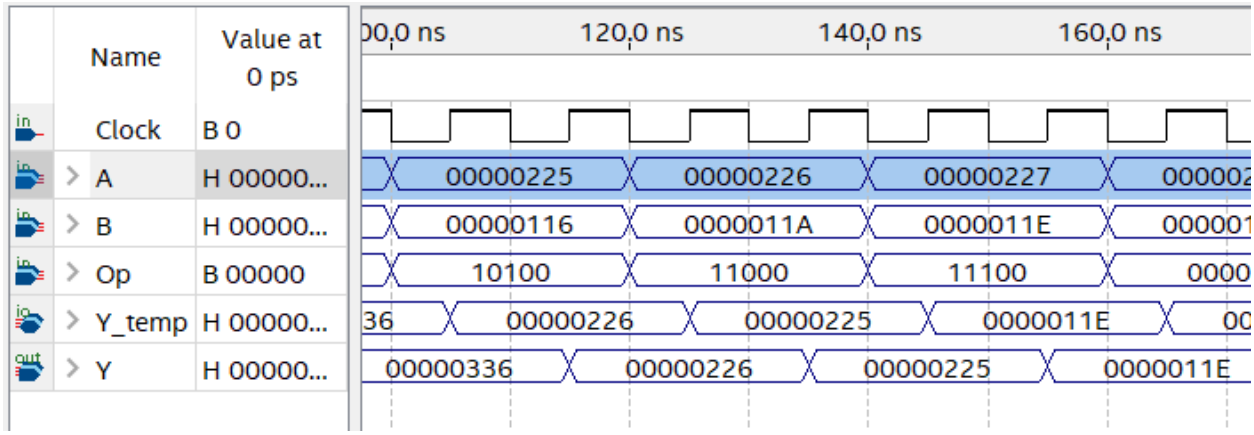

Fig 5. Increment by 1, Decrement by 1, Pass B

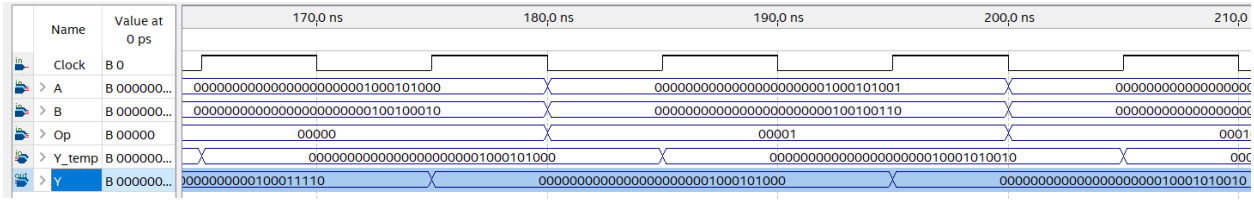

Fig 6. Output passed out, Shift left by 1

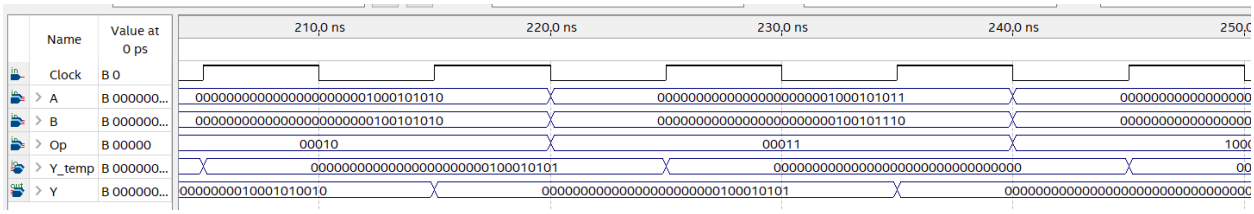

Fig 7. Shift right by 1, Pass 0s

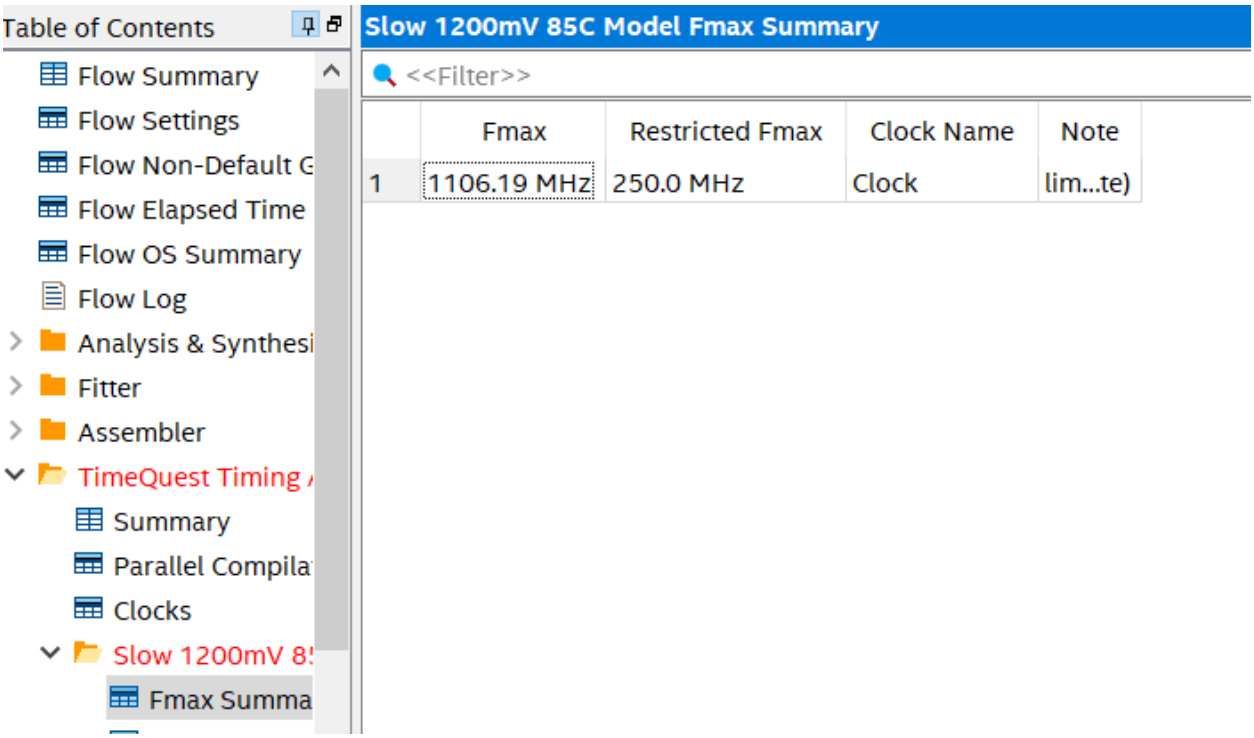

Fig 8. Fmax is 1106.19MHz

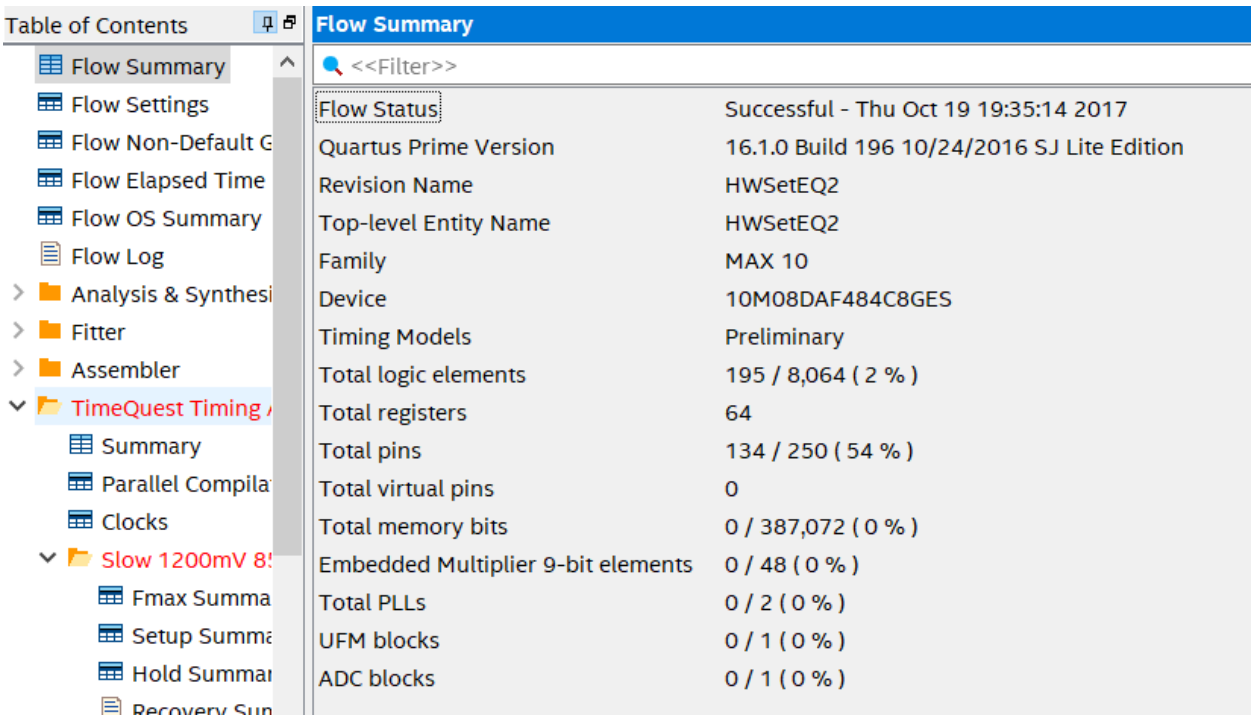

Fig 9. %usage is 2% Number of Logic elements: 195

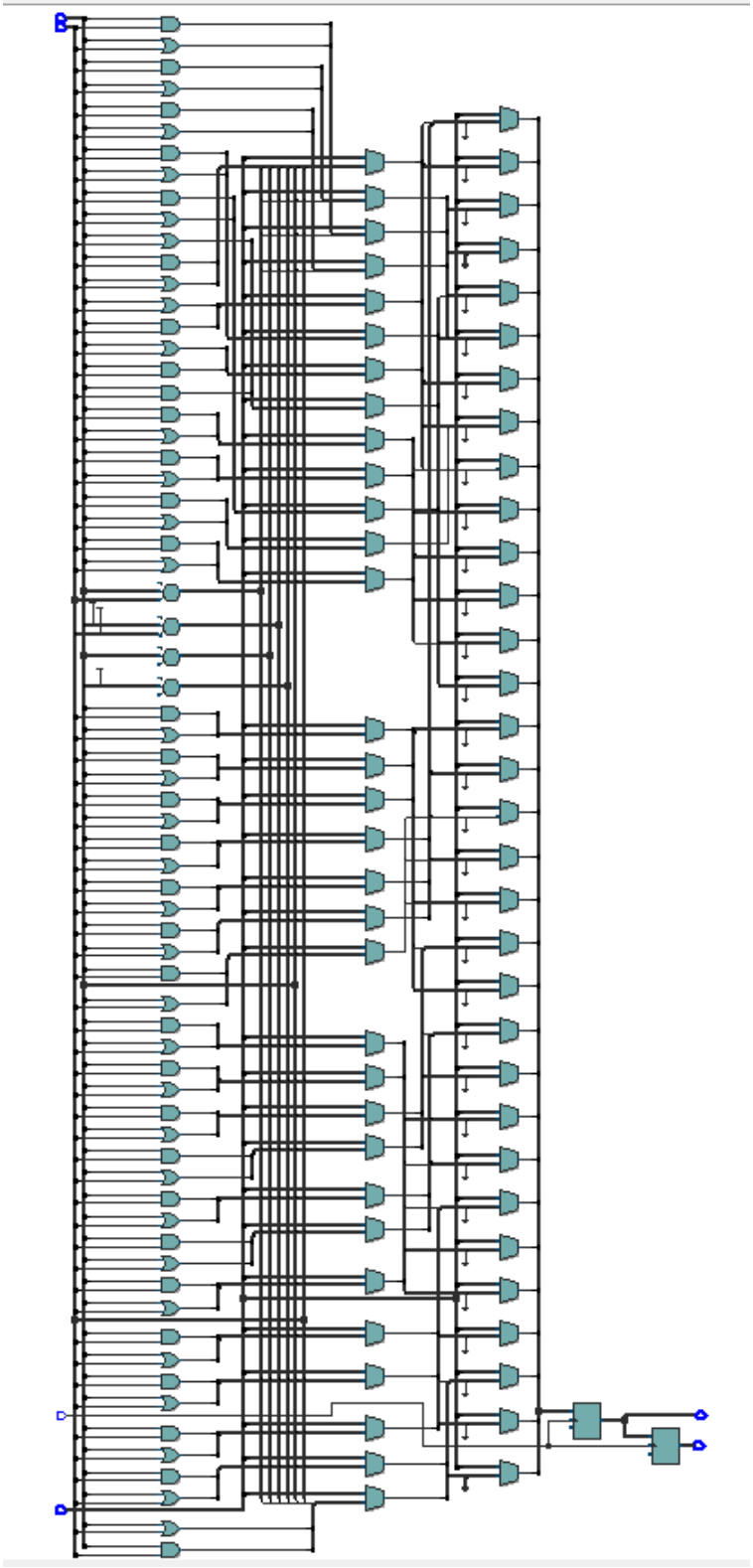

Fig 10. RTL of ALU

3) BAUD GENERATOR

Implementation

1) The following table is used to calculate the required period for the particular Baud rate
2) Based on the input frequency the dividing ratio is calculated
3) The Verilog project is set up
4) Based on the input case, the baud is selected
5) In Modelsim the output can be observed

Frequency	**Time Period**
115200	8.6805us
57600	17.361us
38400	26.042us
28800	34.72us
19200	52.083us
14400	69.44us
9600	104.166us
4800	208.33us
2400	416.667us
1200	833.333us

The following values are compiled for the given input frequency of 14.7456MHz

Clock Frequency	**Dividing clock ratio**
115200	64
57600	128
38400	192
28800	256
19200	384
14400	512
9600	768
4800	1536
2400	3072
1200	6144

Please refer to the above two tables to interpret the screenshots

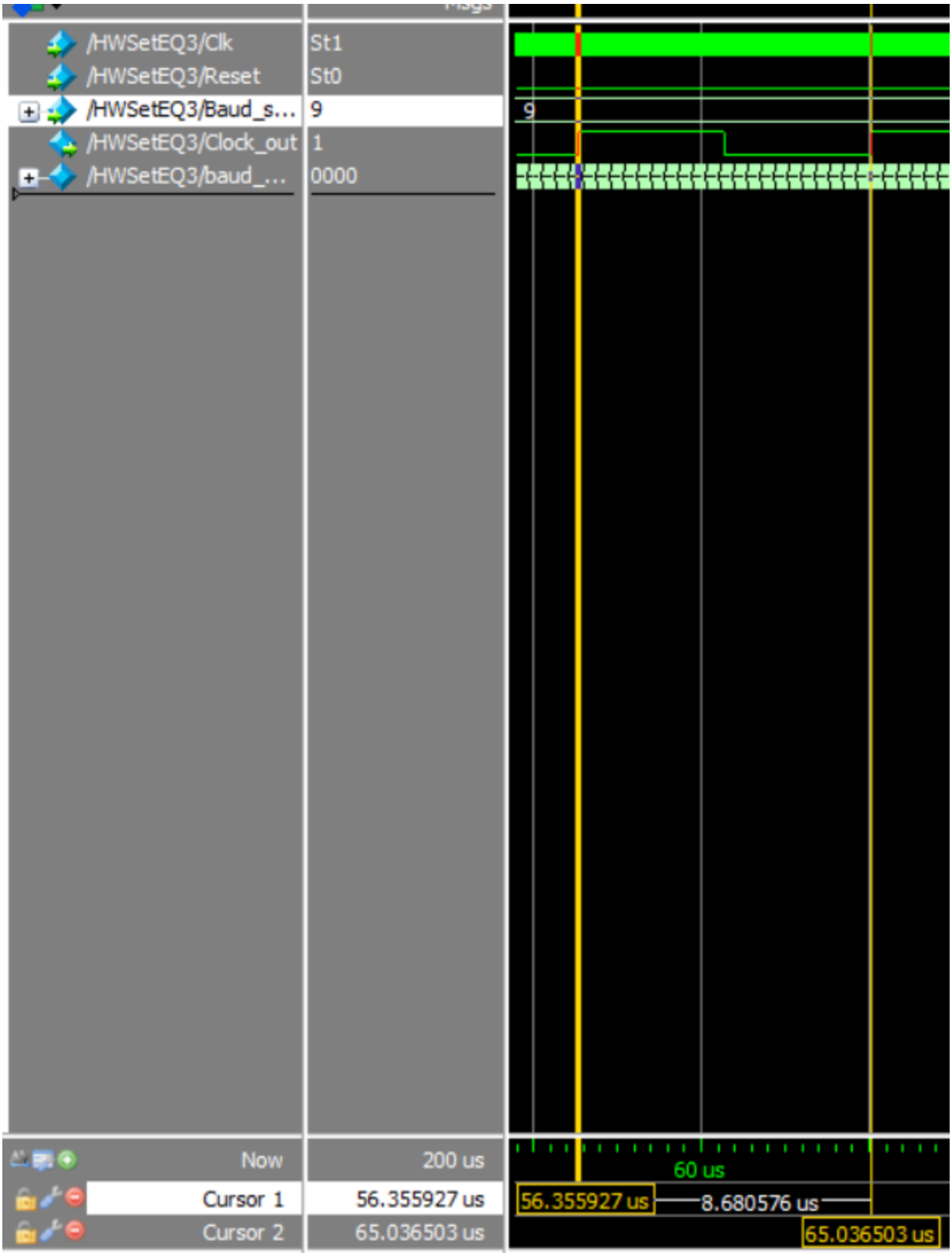

Fig 1. Baud 115200

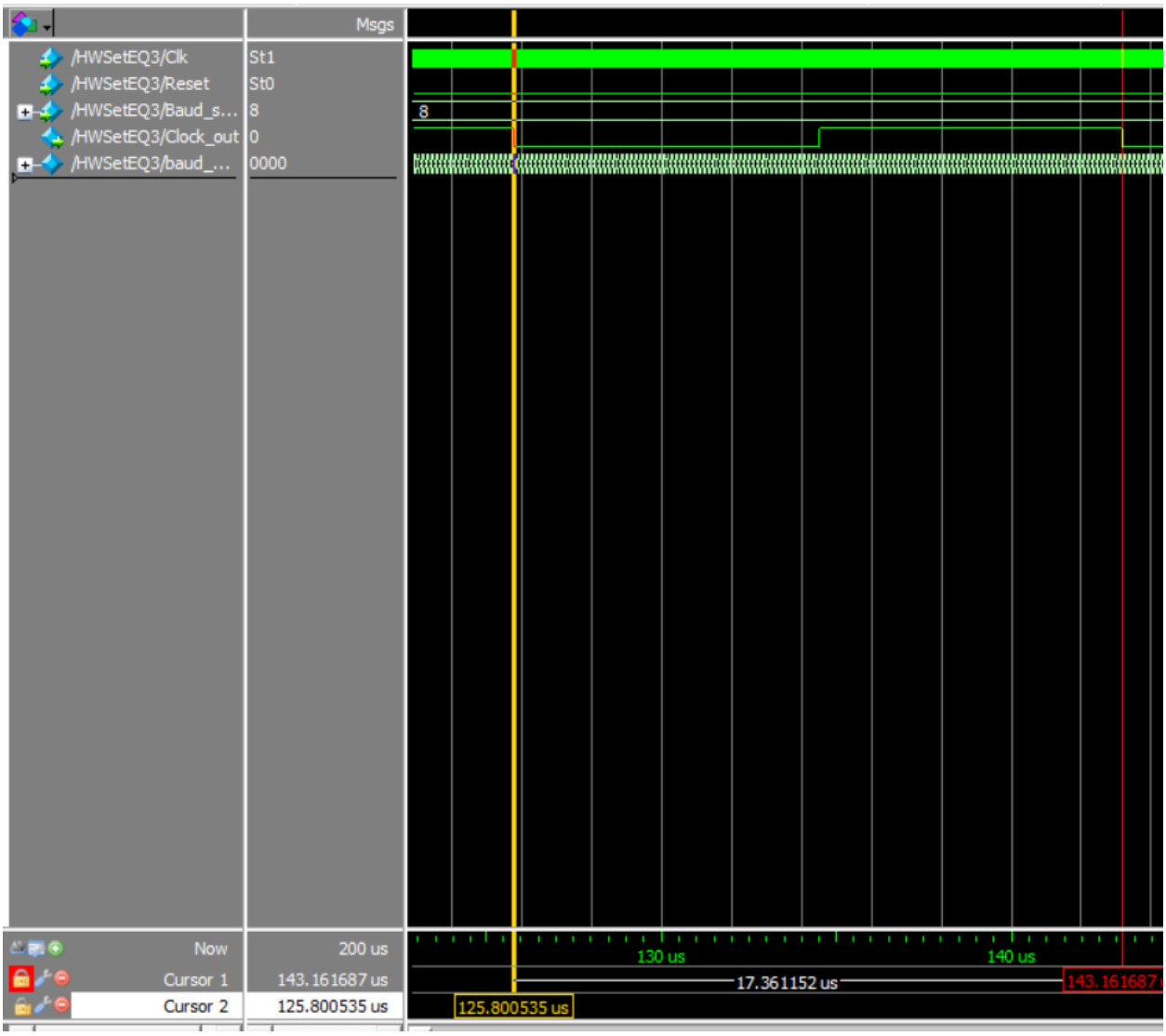

Fig 2: Baud 57600

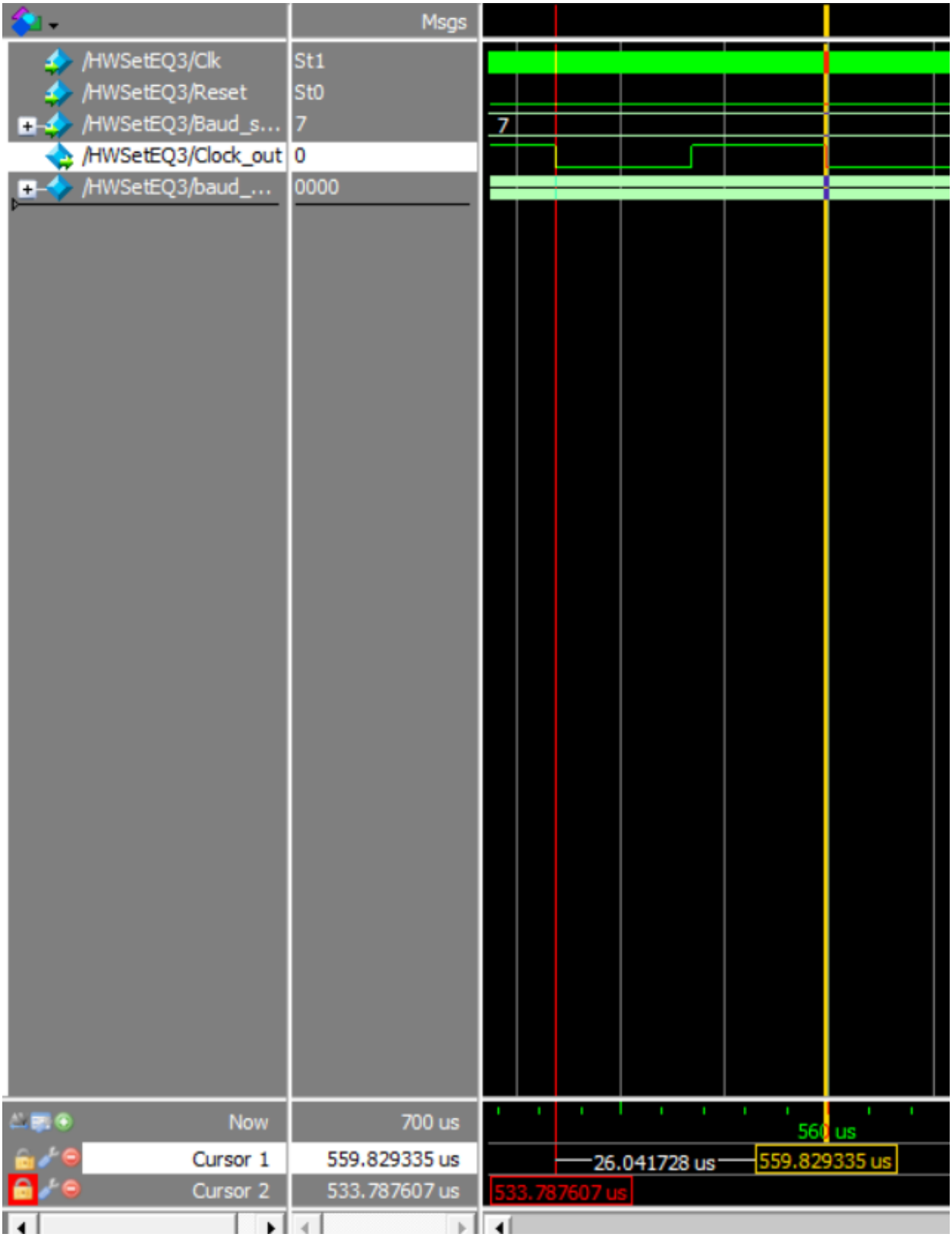

Fig 3. Baud 38400

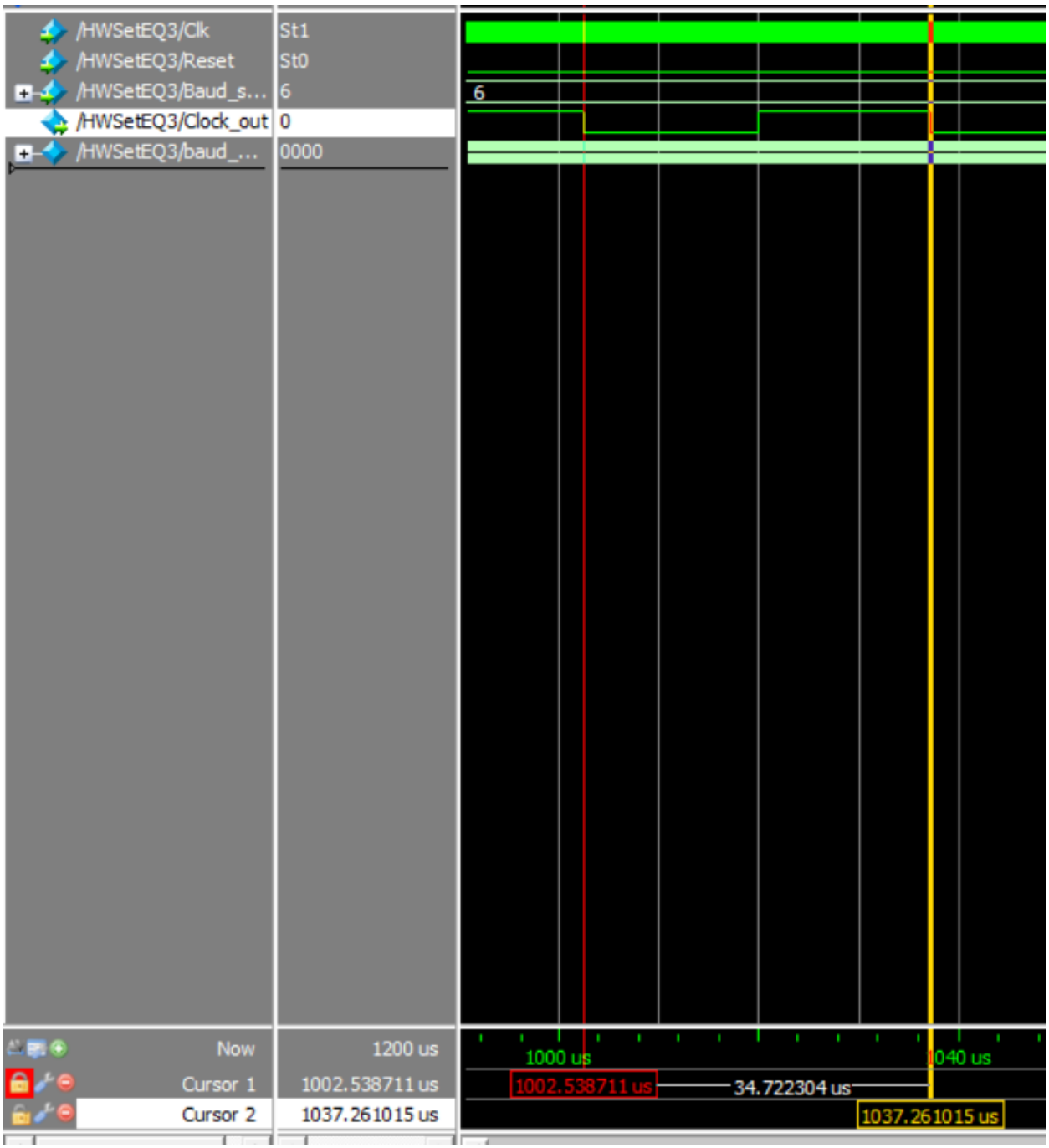

Fig 4. Baud 28800

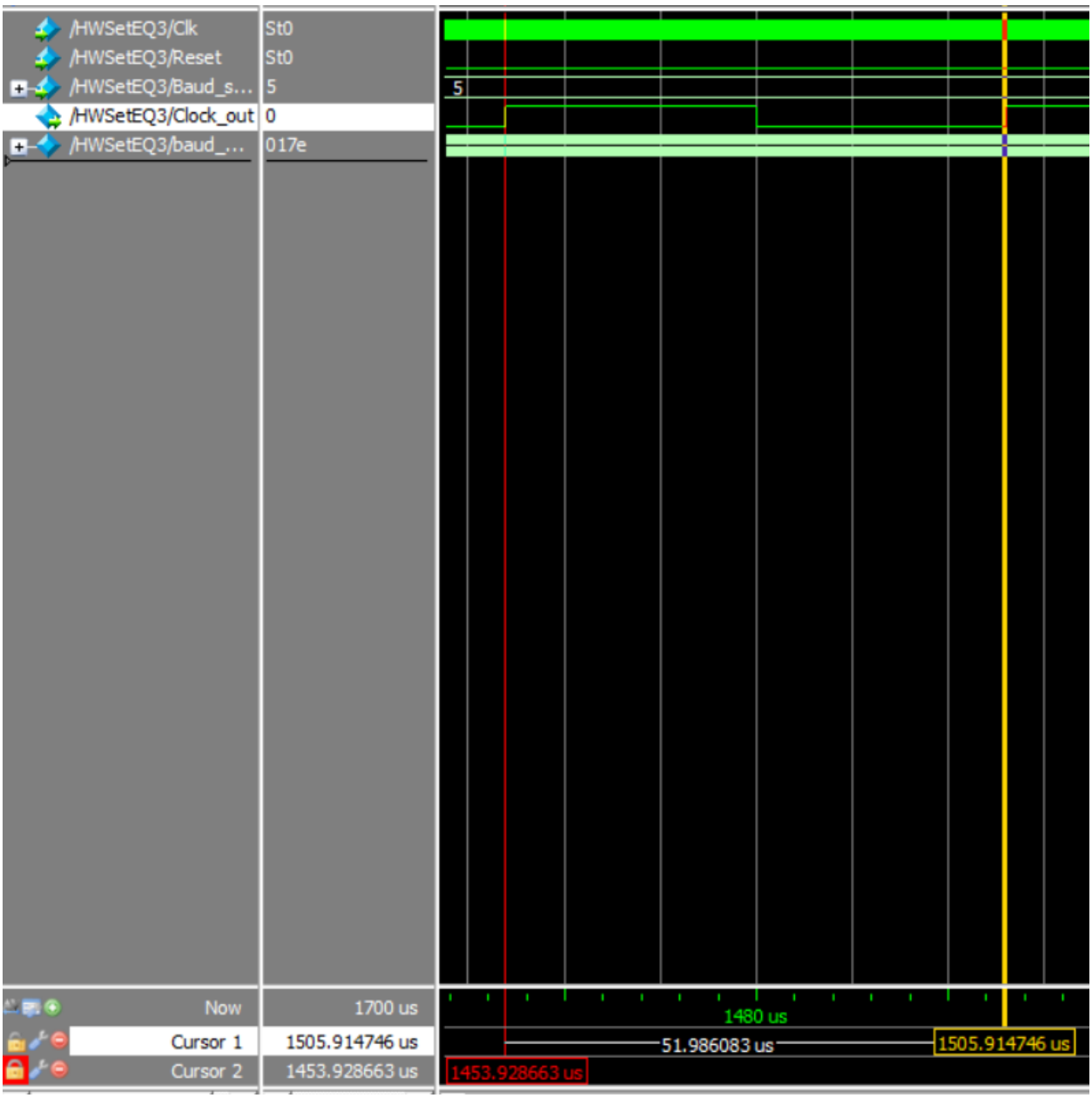

Fig 5. Baud 19200

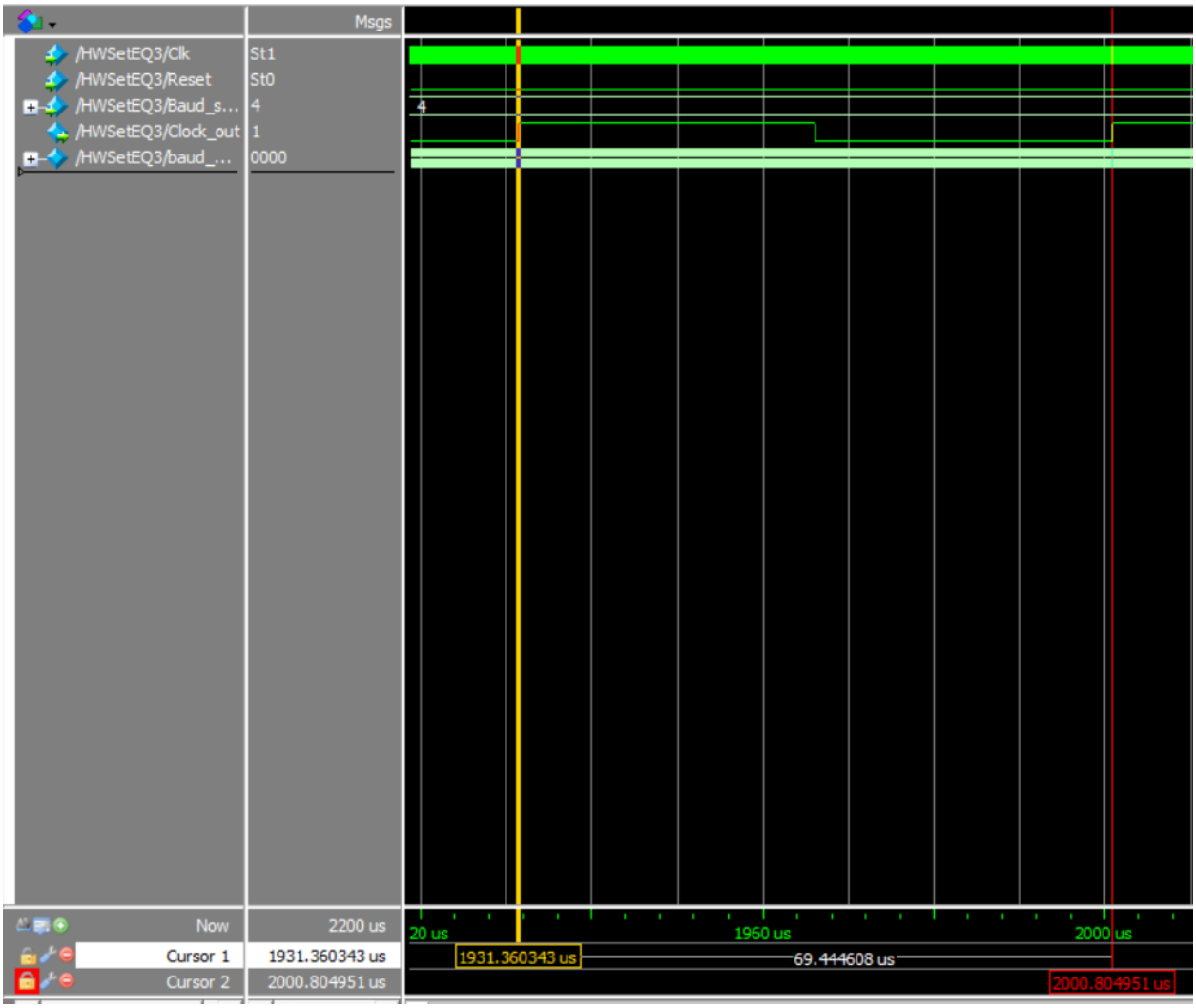

Fig 6. Baud 14400

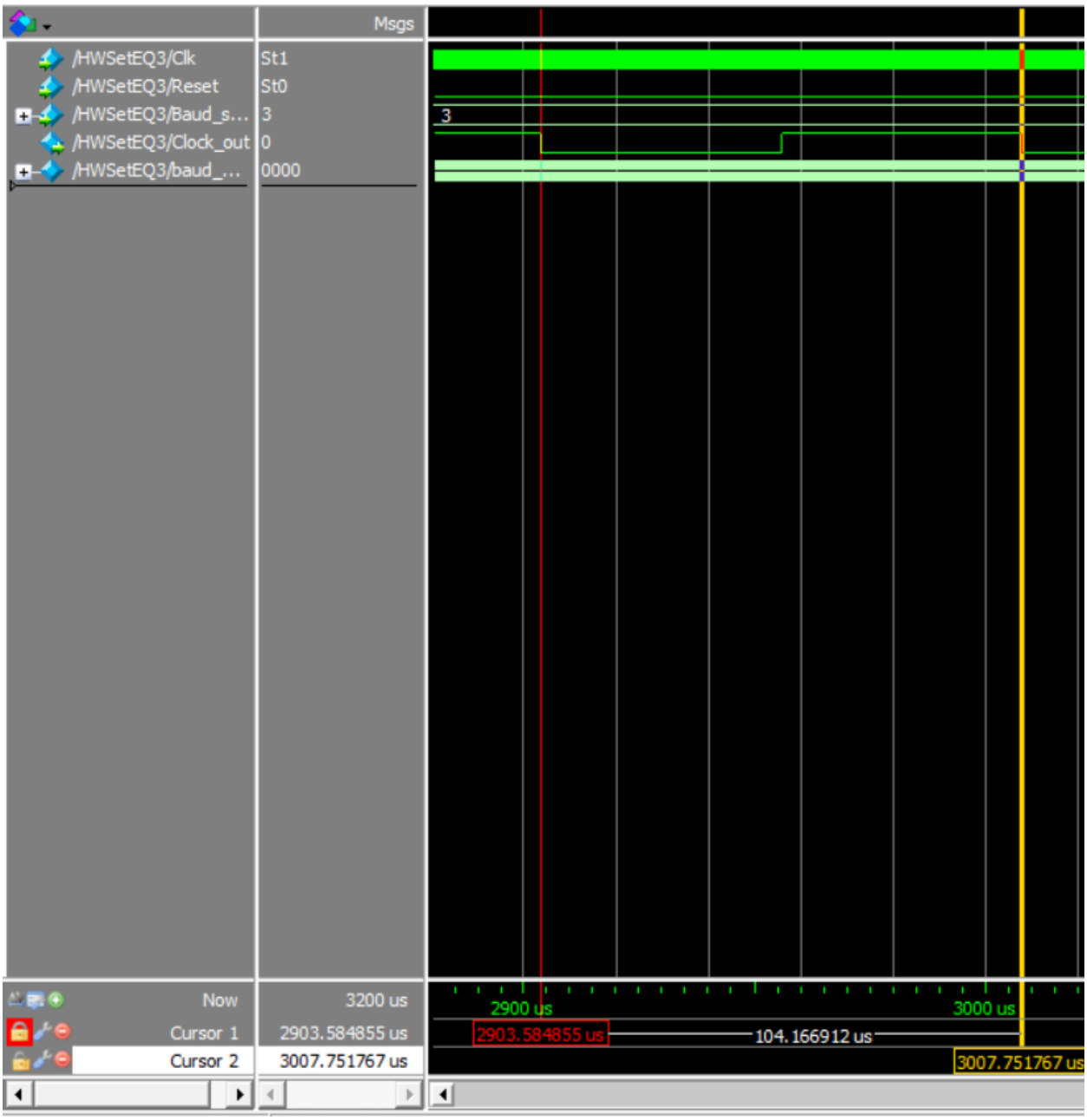

Fig 7. Baud 9600

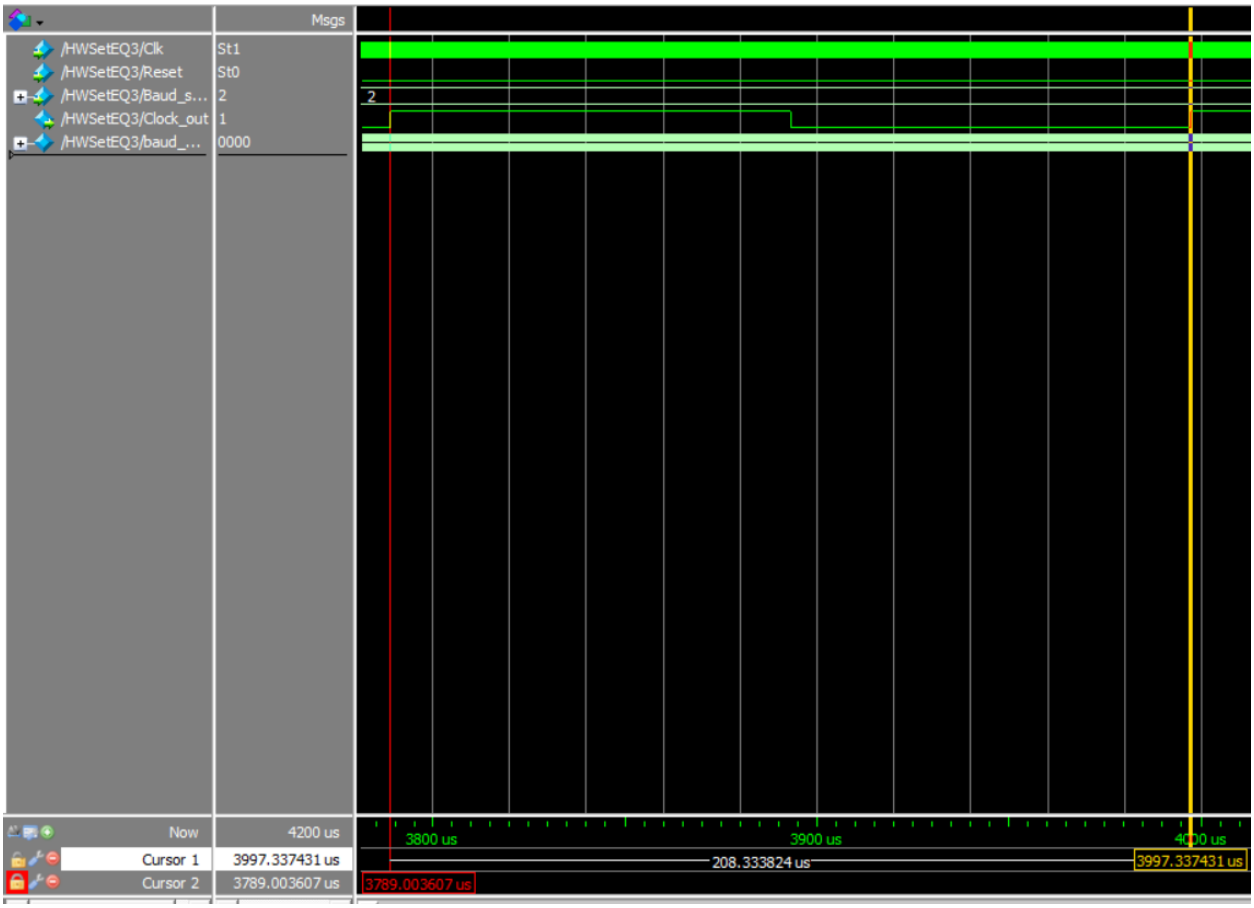

Fig 8. Baud 4800

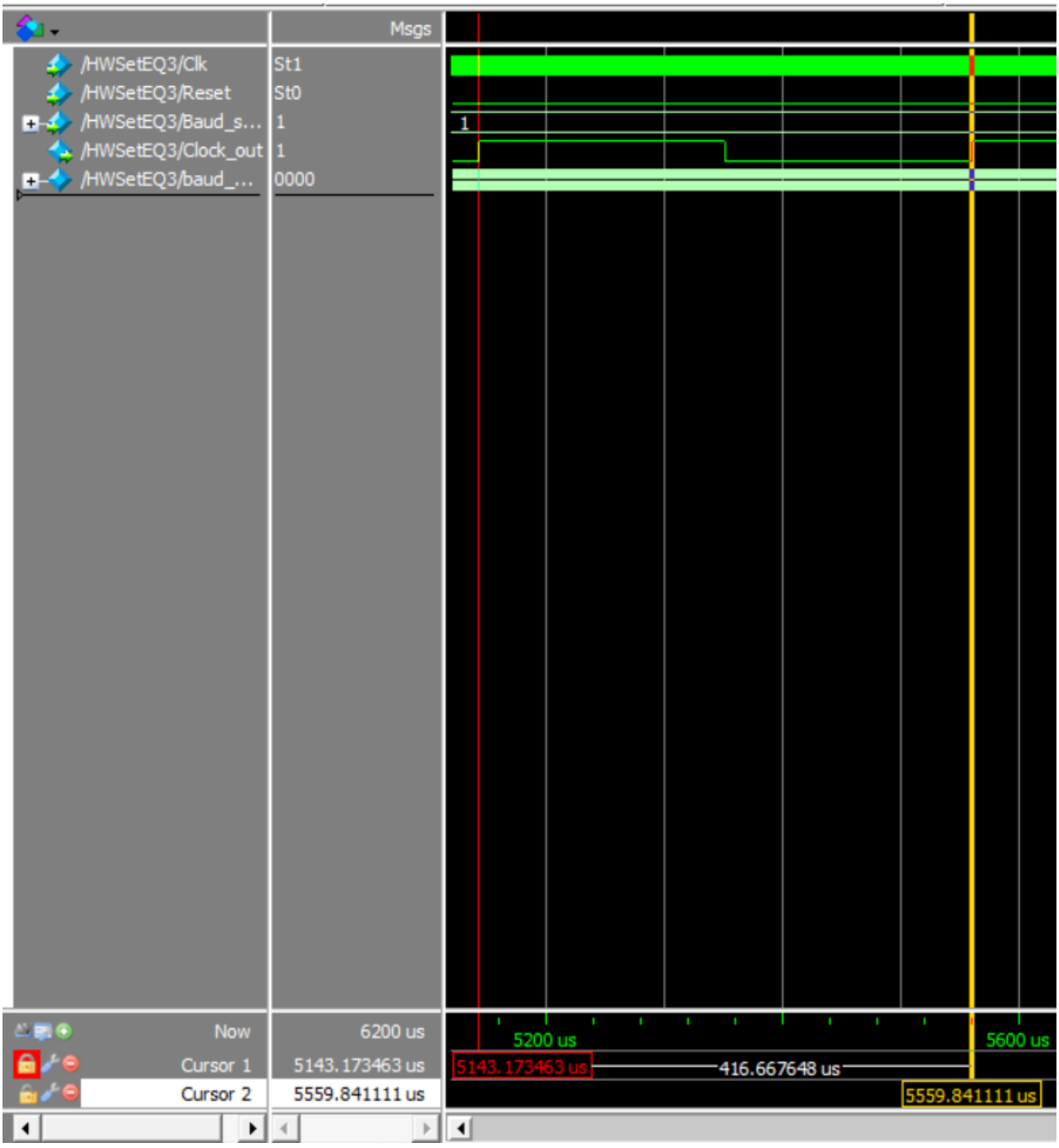

Fig 9. Baud 2400

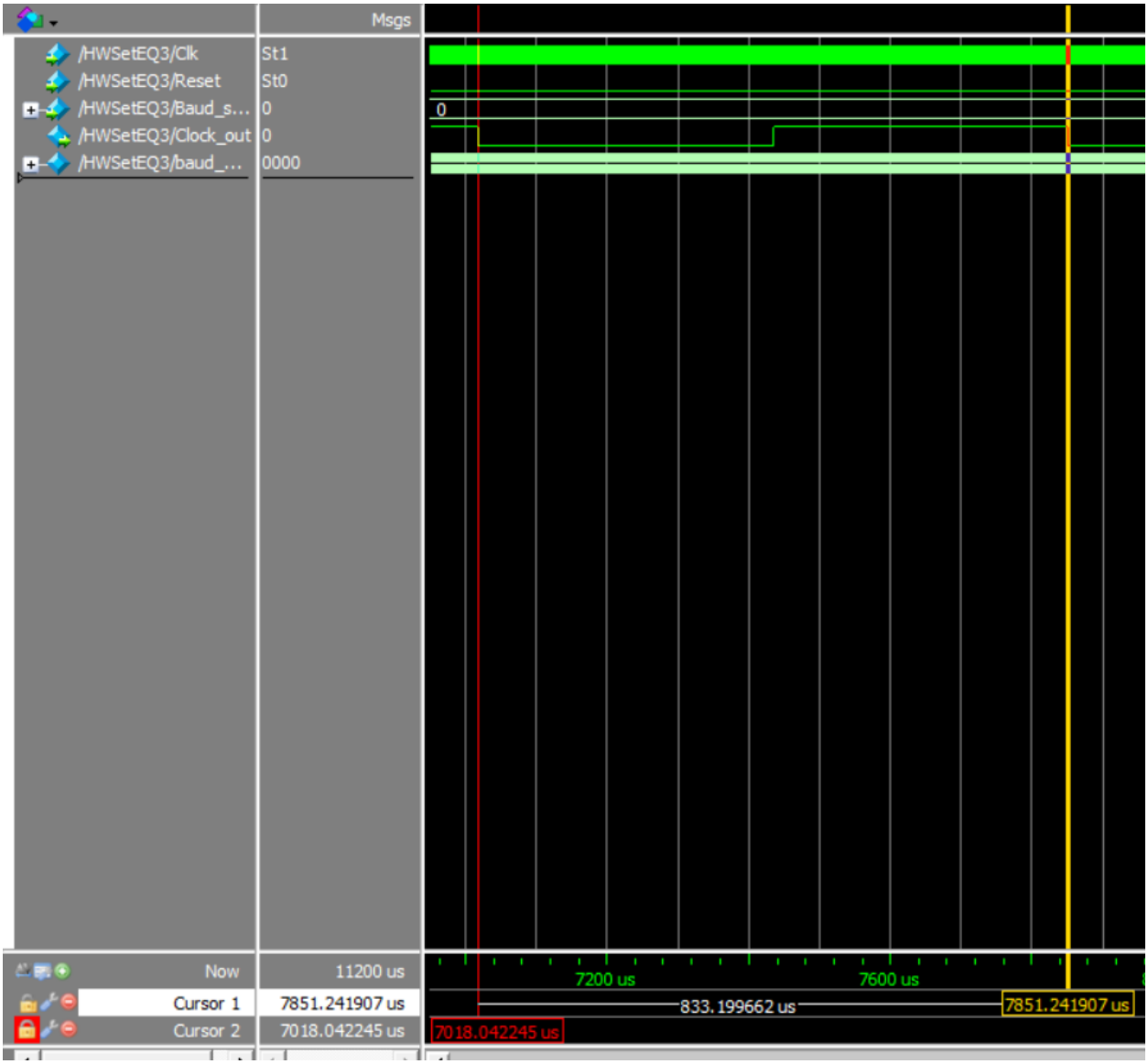

Fig 10. Baud 1200

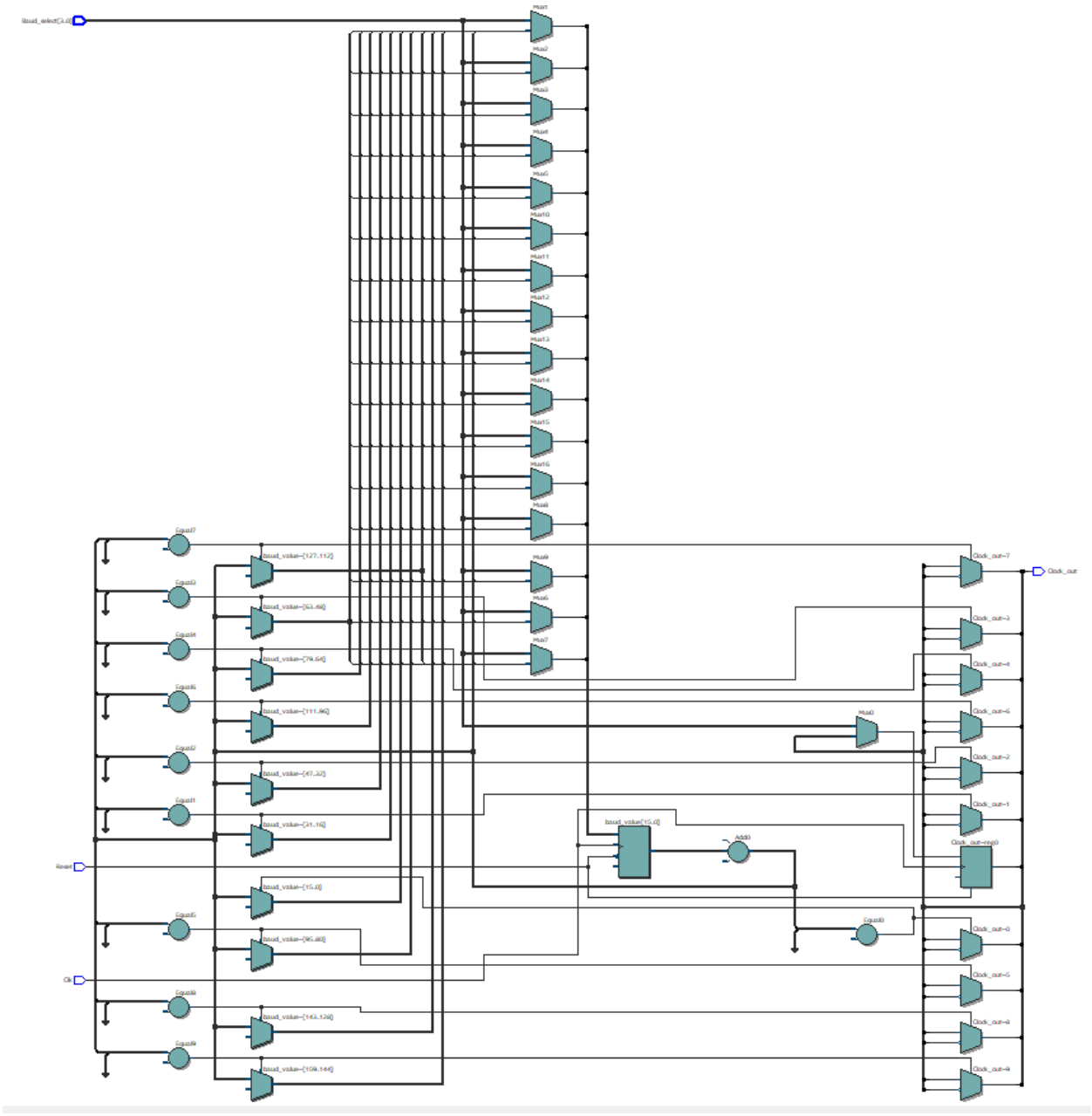

Fig 11. RTL for Baud generator