ARM Exercise 1 Report

Tsotne Putkaradze

7 October 2015

1 Task Given

Write a piece of Synthesizable Verilog design to generate the following outputs when a "Go" pulse is observed at the inputs. OutD is generated synchronous to clk_in. The "Z" in the waveform suggests that the output is in high impedance state.

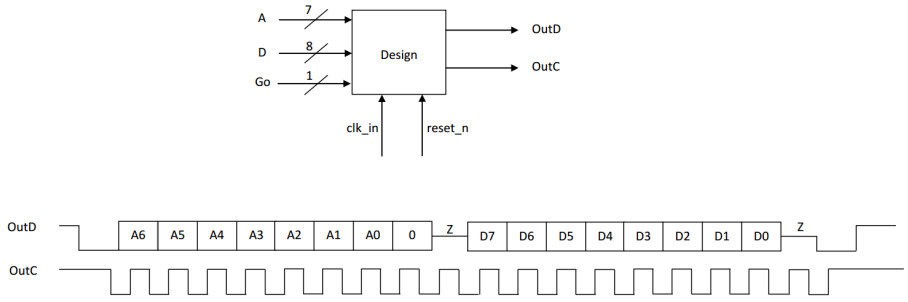


Figure 1: task given

According to fig: 1 (if circuit is designed on rising edge)

1. First value receiver of this data "clocks"(with OutC) is Highest(Left Most) Bit of A input
2. Last Value receiver of this data "clocks"(with OutC) is '0'

2 Solution

Until writing code, I came up with two possible solutions that are:

- Solution A : using 19bit-shift register
- Solution B : using 3bit-counter with FSM

Verilog solutions were simulated using Open Source Software: Icarus Verilog. Both, Verilog and VHDL solutions were synthesized and simulated using Vivado 2015 Synthesis and Simulation Tools, with Tallinn University of Technology Licence.

2.1 Solution A : Solution With Shift Register

Circuit works like this: after Go signal arrives, shift register "*concatenatedInputs* " receives value:
$'0' + A(7bitInput) +' Z' + D(8bitInput) +' Z' +' 0'$
and on every clock cycle we do following:

- on output OutD, we send out *concatenatedInputs* [Left_Most_Bit]
- vector *concatenatedInputs* is shifted to the left and '1' is inserted from right side ('1' is because in idle state our output should be '1' anyways, so when *concatenatedInputs* will be shifted "1+7+1+8+1+1"-times output will be constant '1', because shift register will consist of only '1's. output will be '1' until we reset the circuit or give it new input (with Go signal).
- we observe value of *concatenatedInputs* , if it consists of all 1s, then we stop propagating clk_in into OutC.

With this solution I don't have to use any adder, fsm or any other relatively complicated logic.

2.2 Solution B : Solution With 3bit Counter and FSM

This solution is following, after Go signal arrives, we store the data in registers. We construct one - 3 bit adder. we use this adder to address the appropriate bit of stored inputs, like this:
$OutD <= stored\_Input\_A[counterValue]$; and FSM controls which vector will be sent to output, $stored\_Input\_A$ or $stored\_Input\_D$

3 Decision

However I decided to write code for solution A, because it looked much simpler in circuit point of view, i was using registers, that was definitely necessary to use(unless inputs are not changing during transmission of data), and i am using them as shift register, so I'm not adding anything heavy.

First I wrote VHDL code because i have VHDL background, then i learnt Verilog and wrote the code there.

In both solutions everything is parametrised: length of delay before and after inputs, length of high impedance output, length of input A and input D.

P.S. In attached files, there is also folder called "Additional" there, I've constructed circular serial communication between N number of FPGAs, there I've used the logic of Solution B - storing vector and selecting right bit with counter value.

4 Verification

For validating the correctness of logic, in both languges I generated test-benches, that take both output OutD and OutC, and reconstruct the provided data, in other words, we use OutC as a clock to sample/receive OutD. and I used Assert statements to compare the provided and reconstructed data.

5 Simulation

On the figures below, fig: 2, 3, 4, 5, you can see the simulations when following inputs are given: $A = 1000001, D = 10000001$ Codes are also synthesizable, you can view the synthesis reports in corresponding folders. We used only this one pattern, because in this particular case, its enough to observe only first and last bits of inputs - if they are lost or not. This is enough because the module we wrote do not modify the data, it is storing and outputting it only, so - if first and last bits are correct, then data inside is unaffected. I could have written a function that generates test patterns but It is completely out of necessity.

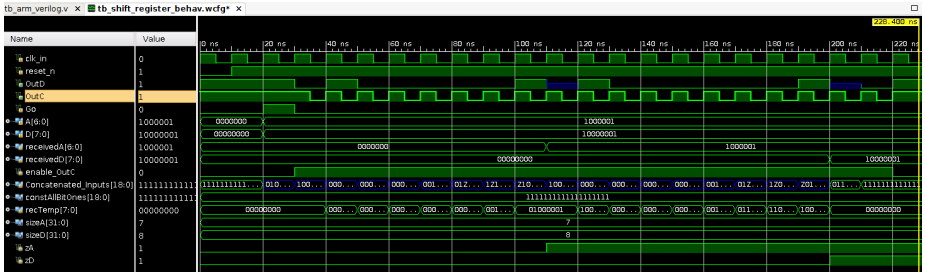


Figure 2: waveform of outputs generated from above mentioned inputs, logic is written in Verilog Language, simulation tool: Vivado 2015.2

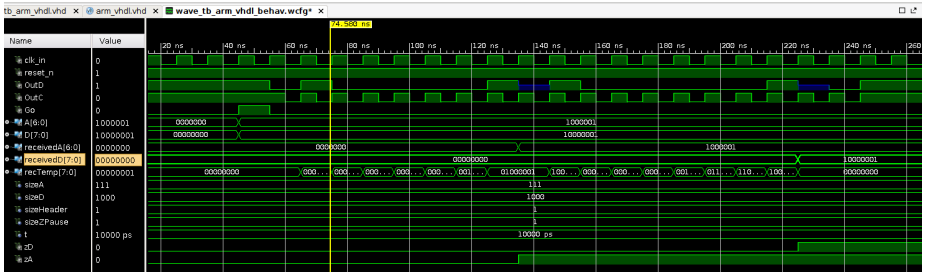
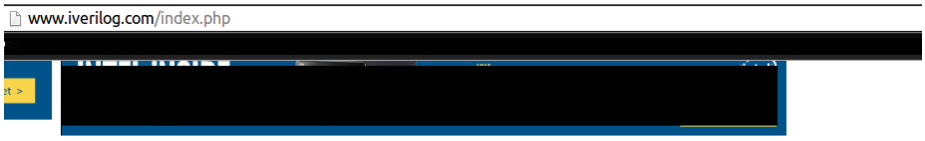


Figure 3: waveform of outputs generated from above mentioned inputs, logic is written in VHDL Language, simulation tool: Vivado 2015.2



TERMINAL OUTPUT

```
to make things simple: when OutC = 0 then output should be valid. it should start and end with 0, after A & D it should output 'Z'
Time = 10, outputs: OutC = 1 OutD = 1
Time = 20, outputs: OutC = 1 OutD = 1
Time = 30, outputs: OutC = 1 OutD = 1
Time: 30, outputs: START test pattern N1: A = 7'b1000001 D = 8'b10000001
Time = 40, outputs: OutC = 0 OutD = 0
Time = 50, outputs: OutC = 0 OutD = 1
Time = 60, outputs: OutC = 0 OutD = 0
Time = 70, outputs: OutC = 0 OutD = 0
Time = 80, outputs: OutC = 0 OutD = 0
Time = 90, outputs: OutC = 0 OutD = 0
Time = 100, outputs: OutC = 0 OutD = 0
Time = 110, outputs: OutC = 0 OutD = 1
Time = 120, outputs: OutC = 0 OutD = z
Time = 130, outputs: OutC = 0 OutD = 1
Time = 140, outputs: OutC = 0 OutD = 0
Time = 150, outputs: OutC = 0 OutD = 0
Time = 160, outputs: OutC = 0 OutD = 0
Time = 170, outputs: OutC = 0 OutD = 0
Time = 180, outputs: OutC = 0 OutD = 0
Time = 190, outputs: OutC = 0 OutD = 0
Time = 200, outputs: OutC = 0 OutD = 1
Time = 210, outputs: OutC = 0 OutD = z
Time = 220, outputs: OutC = 0 OutD = 0
Time = 230, outputs: OutC = 1 OutD = 1
Time = 240, outputs: OutC = 1 OutD = 1
Time = 250, outputs: OutC = 1 OutD = 1
Time = 260, outputs: OutC = 1 OutD = 1
Time = 270, outputs: OutC = 1 OutD = 1
OUTPUT DO MATCH WITH INPUT (A = 7'b1000001; D = 8'b10000001)
Time = 280, outputs: OutC = 1 OutD = 1
Time = 290, outputs: OutC = 1 OutD = 1
Time = 300, outputs: OutC = 1 OutD = 1
Time = 310, outputs: OutC = 1 OutD = 1
Time = 320, outputs: OutC = 1 OutD = 1
Time = 330, outputs: OutC = 1 OutD = 1
Time = 340, outputs: OutC = 1 OutD = 1
Time = 350, outputs: OutC = 1 OutD = 1
Time = 360, outputs: OutC = 1 OutD = 1
Time = 370, outputs: OutC = 1 OutD = 1
Time = 380, outputs: OutC = 1 OutD = 1
Time = 390, outputs: OutC = 1 OutD = 1
```

Figure 4: waveform of outputs generated from above mentioned inputs, logic is written in Verilog Language, simulation tool: Icarus Verilog - www.iverilog.com

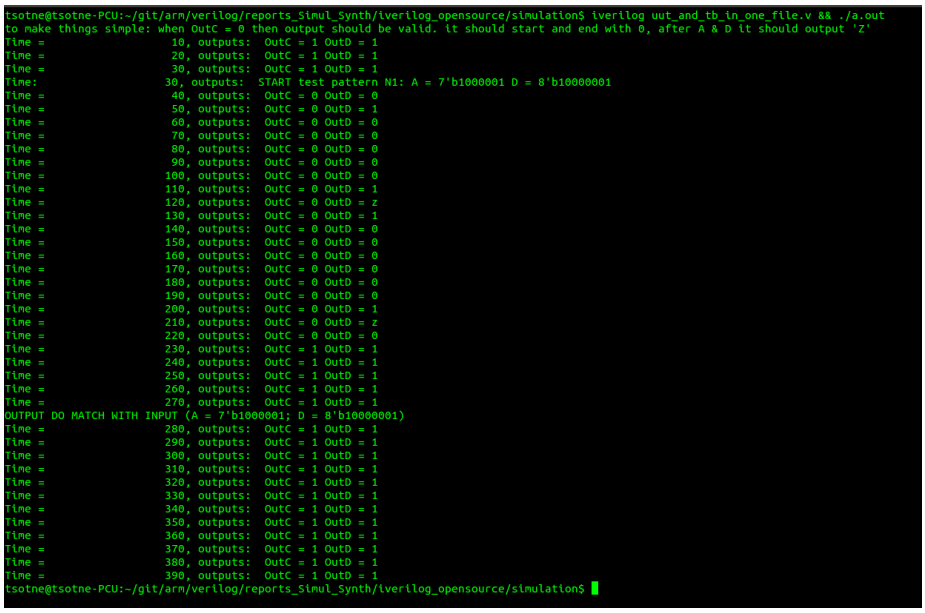


Figure 5: waveform of outputs generated from above mentioned inputs, logic is written in Verilog Language, simulation tool: Icarus Verilog - http://iverilog.icarus.com