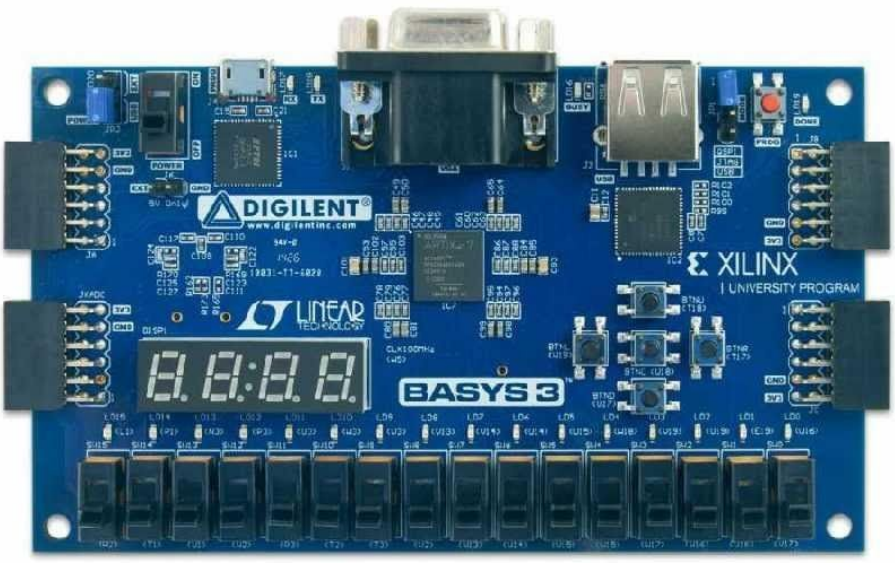


Dual-Clock FIFO

FPGA PROJECT

Kuruvilla Varghese | DFPGA | 08-06-20

PROJECT REPORT REPRESENTED BY-

ABHAY SHARMA, SR-NO-16708

BIKRAM KUMAR PANDA, SR-NO-17030

Introduction

Dual Clock FIFO, works as a bridge between the two circuits working at different frequencies to communicate with each other. Here, we have designed Dual Clock FIFO of different as well as equal port size of input and output.

DESIGN TOP VIEW

Here, we have designed DUAL Clock FIFO in sub parts and then connected them together. Sub parts (modules) of this design are:-

1. Write Address and Write Full Flag Generator. (wr_addrs_full.vhd)
2. Read Address and Read Empty Flag Generator. (rd_addrs_empty.vhd)
3. Synchronizer Read to Write Address. (sync_rd2wr.vhd)
4. Synchronizer Write to Read Address. (sync_wr2rd.vhd)
5. DPRAM (mem_fifo.vhd)

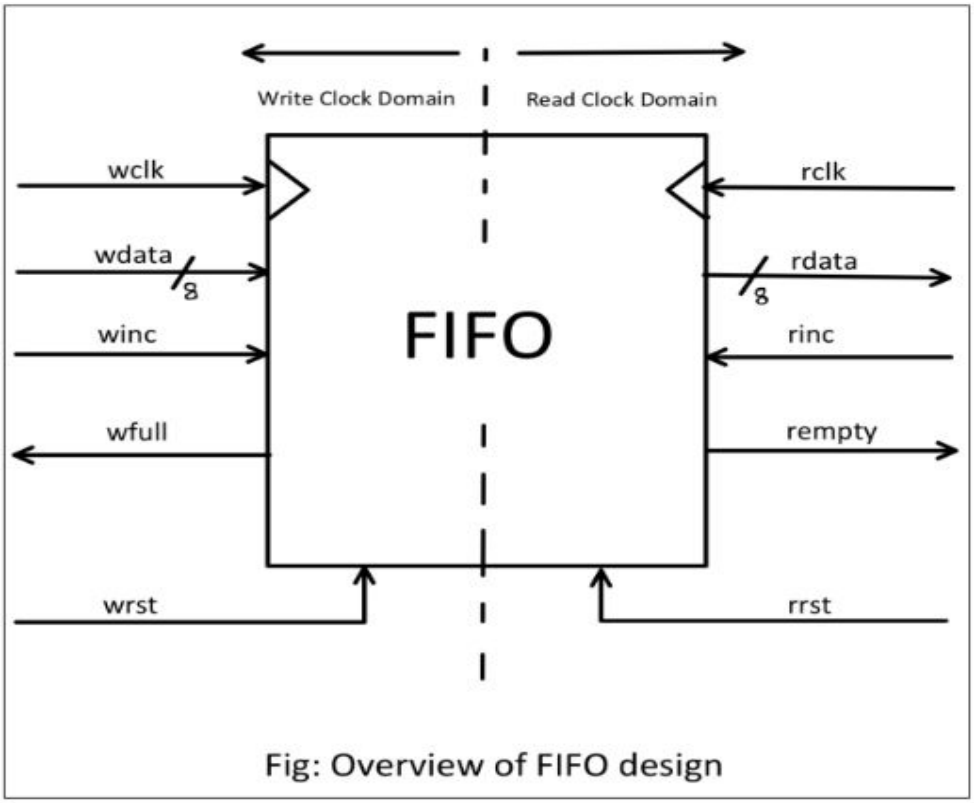


Fig: Overview of FIFO design

CONCEPT OF WRITE FULL FLAG GENERATION:-

The full flag is generated in the write clock domain using the comparison of the write pointer (**wgray_next**) and the synchronized read pointer (**wq2_rptr**). These address pointer values are in gray codes. Since in gray codes the consecutive state only differs by one bit, so synchronization process is more stable when we are synchronizing the read address pointer from read clock domain to write clock domain.

There are mainly three conditions to be satisfied to generate the write flag:-

1. The MSBs of the **wgray_next** and the **wq2_rptr** must be different, showing that the write counter (**wgray_next**) has gone one more time through the counting cycle than read pointer (**wq2_rptr**).

2. The 2nd MSBs of the **wgray_next** and the **wq2_rptr** must be different. Why?
   Suppose,
   wgray_next = 11000 (binary = 10000)
   wq2_rptr = 01000 (binary = 01111)
   Then if we don't include this condition-2, condition-1 and condition-2 are satisfied (i.e main MSB's are equal and all other bits are also equal) which will trigger the **wfull flag** to 1, which is wrong. Since we still have 14 places for the FIFO to write data in (since **read ptr** is at 15th position and **write ptr** is at 0th position, therefore we still have 14 places to write data in).

3. All other bits of **wgray_next** and **wq2_rptr** must be equal.

CONCEPT OF READ EMPTY FLAG GENERATION:-

The Empty flag is generated in the read clock domain using the comparison of the read pointer (**rgray_next**) and the synchronized write pointer (**rq2_wptr**). These address pointer values are in gray codes.

Unlike write full flag, read empty flag is easy to enable. The logic behind enabling the read empty flag is that, flag should be enable only when every data-set inside the DPRAM has been sent to the output port. This condition will arise only when the read and write pointer get align on the same location (i.e gray coded value of both the pointers should be same).

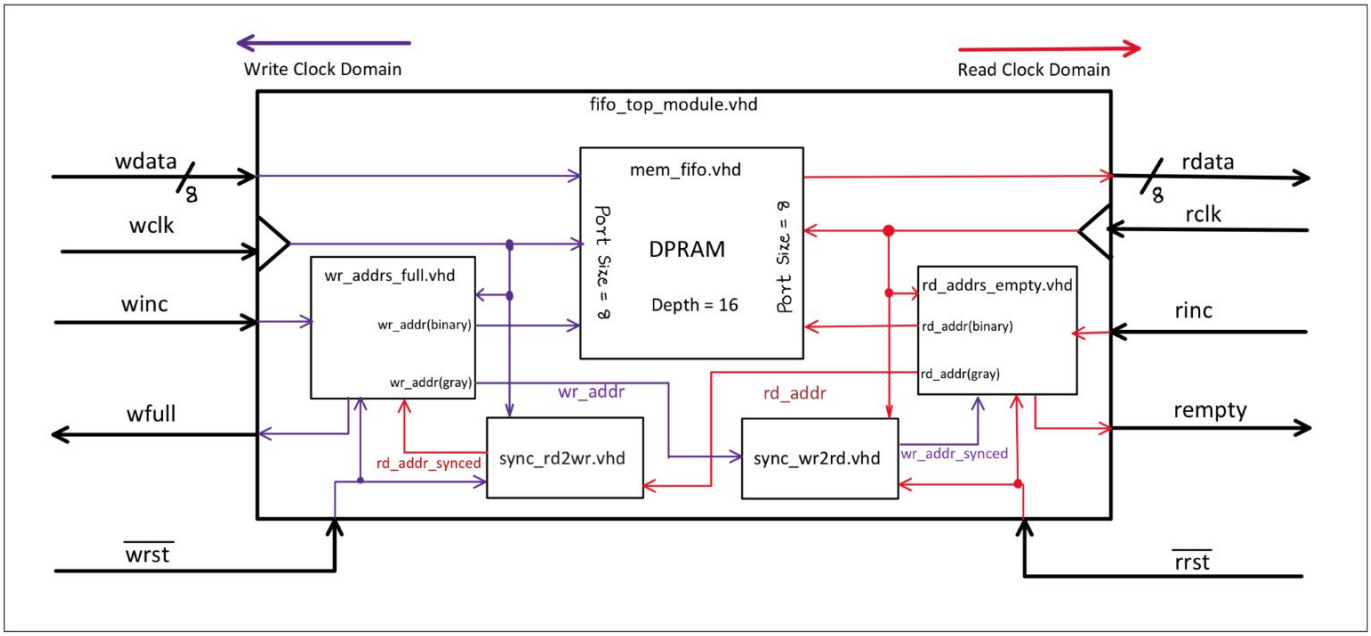


Fig: Detailed Block Diagram of our FIFO

WORKING OF SUB-MODULES

Now we will see the working of sub-modules individually.

1. Write Address and Write Full Flag Generator :-

In this module we have incremented the write address on each write clock pulse whenever winc signal is enable. Here, we have used gray code generator to convert the write address into gray code so that we can send gray code of write address on read module using 2-stage synchronizer. This gray-coded write address will further be used in read module to generate the read empty flag at output by using proper logic.

- *Signals in Write Address and Write Full Flag Module :-*

  wq2_rptr :- It's a 5-bit Input signal coming from **Synchronizer Read to Write Address**. This will be used to enable write full flag (i.e wfull).
  winc:- It's a 1-bit input signal. It will be used to increase the write address.
  wclk : - Input write clock to write the data in DPRAM.
  wrst :- Active low reset for write address module.
  wfull :- It's one bit output signal. It will be one when FIFO is FULL, else zero.
  waddr :- It's a 4-bit output signal that is goint into DPRAM as the write address.
  wptr :- It's a 5-bit O/p signal. It's the gray-coded value of write address.

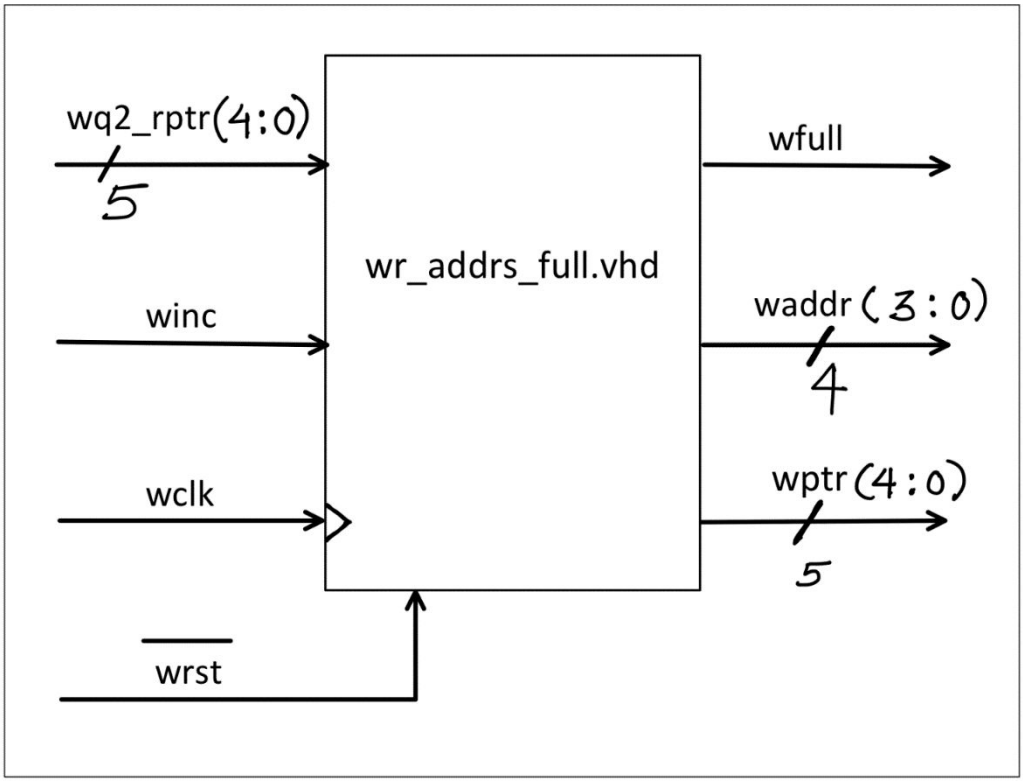
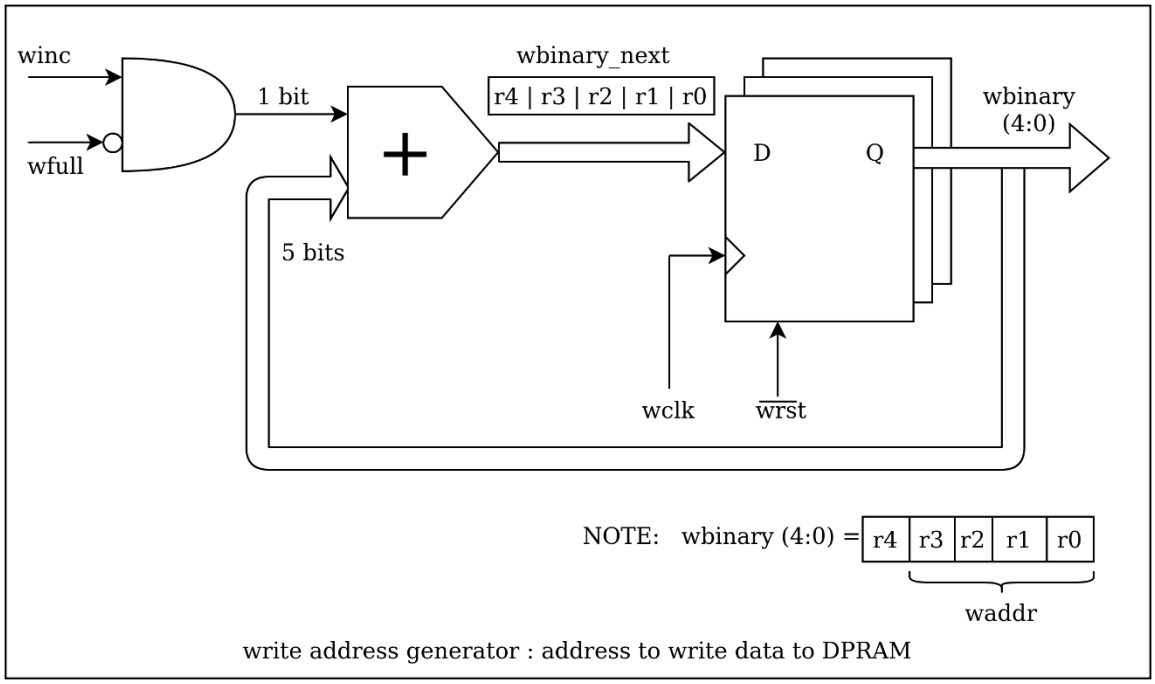
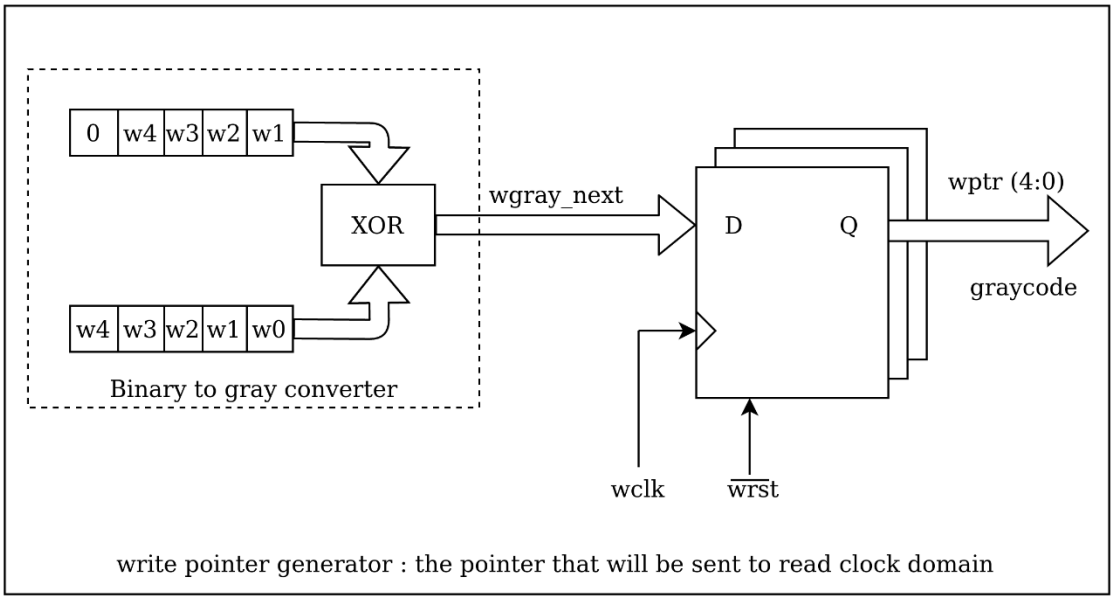


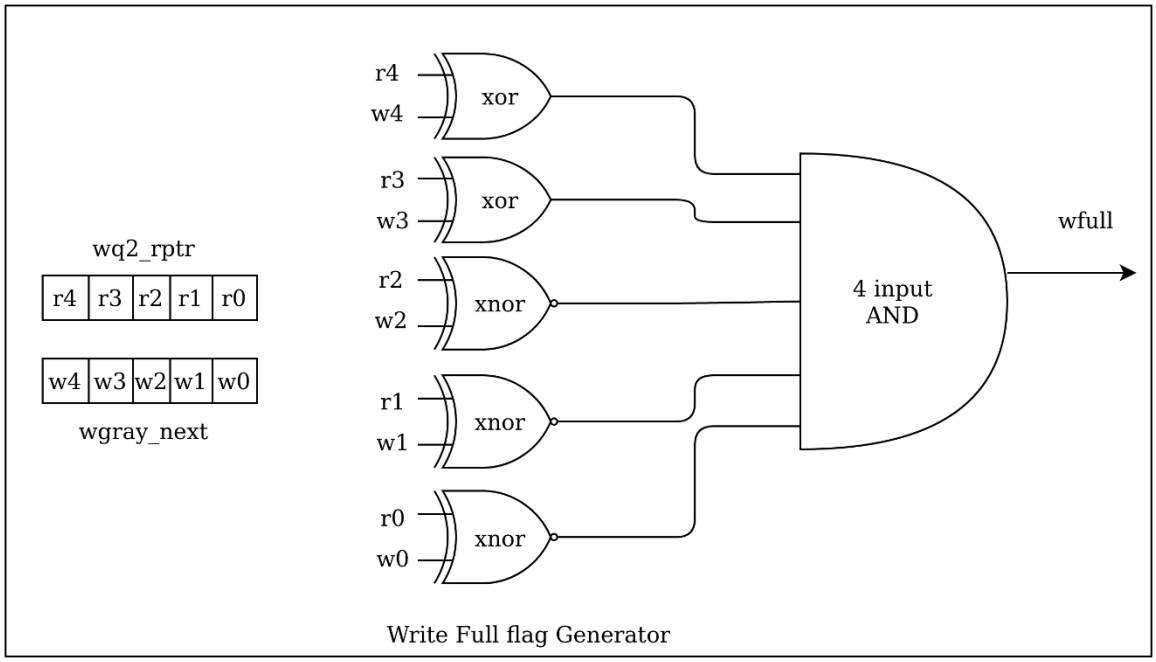
Fig: Write Address and Write Full Flag Generator



write address generator : address to write data to DPRAM



write pointer generator : the pointer that will be sent to read clock domain



Write Full flag Generator

2. Read Address and Read Empty Flag Generator :-

In this module we have incremented the read address on each read clock pulse whenever rinc signal is enable. Here, we have used gray code generator to convert the read address into gray code so that we can send gray code of read address on write module using

2-stage synchronizer. This gray-coded read address will further be used in write module to generate the write flag at output by using proper logic.

- *Signals in Read Address and Read Empty Flag Module :-*

rq2_wptr :- It's a 5-bit Input signal coming from **Synchronizer Write to Read Address**. This will be used to enable read empty flag (i.e rempty).
rinc:- It's a 1-bit input signal. It will be used to increase the read address.
rclk : - Input read clock to read the data from DPRAM.
rrst :- Active low reset for read address module.
rempty :- It's one bit output signal. It will be one when FIFO is empty, else zero.
raddr :- It's a 4-bit output signal that is going into DPRAM as the read address.
rptr :- It's a 5-bit O/p signal. It's the gray-coded value of read address.

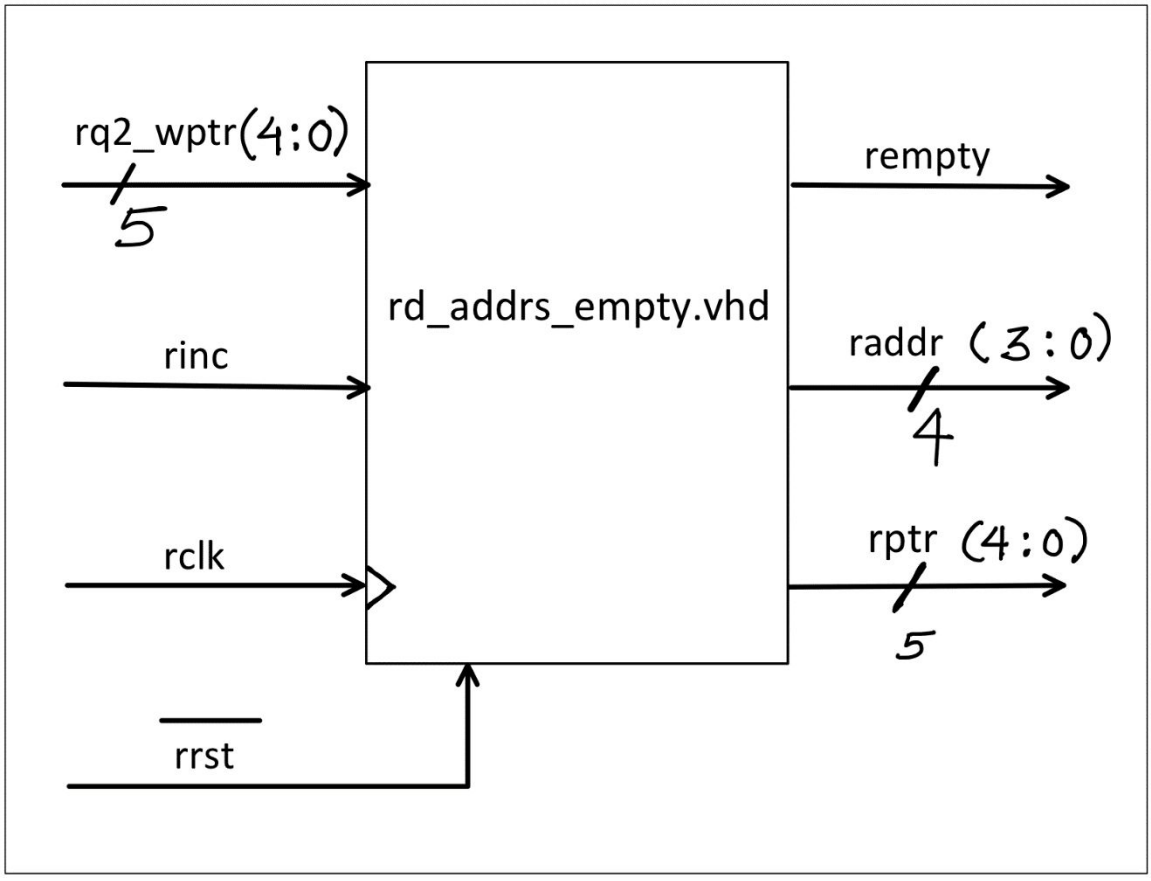
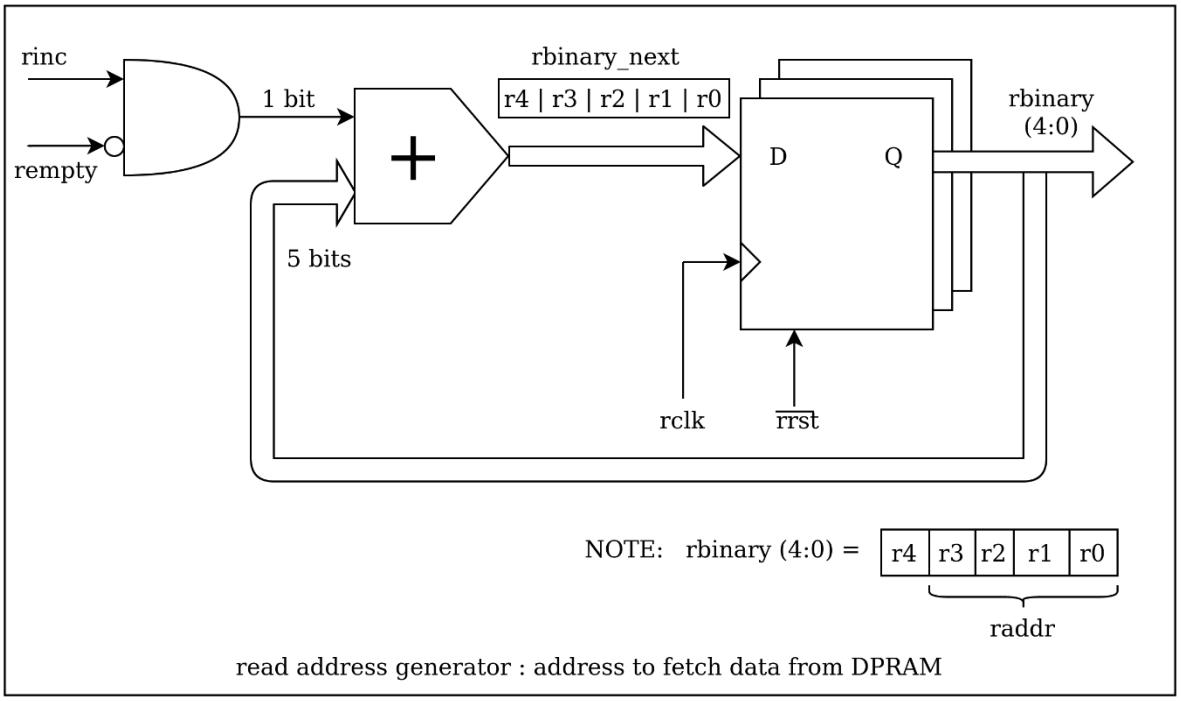
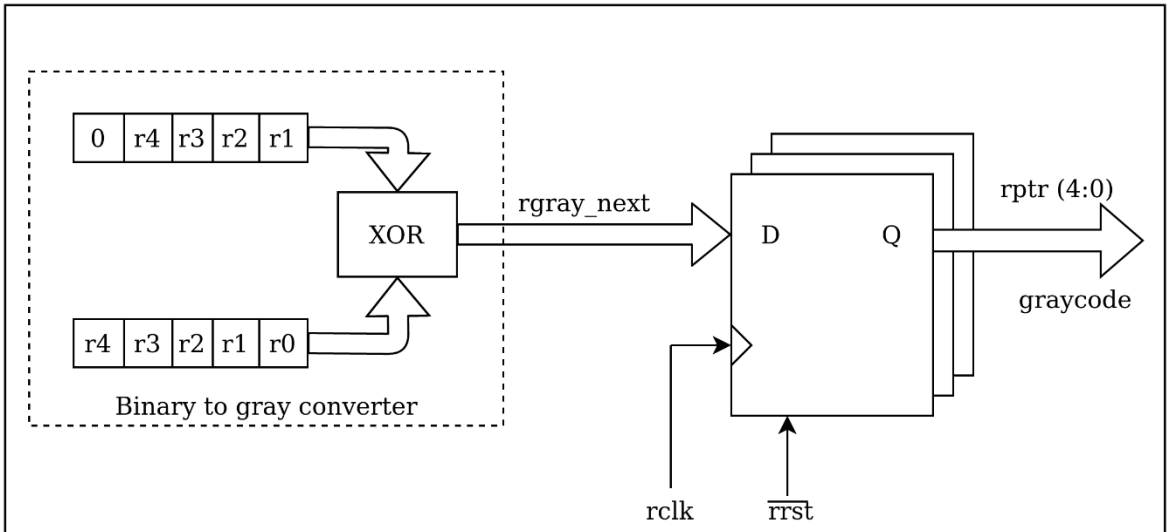


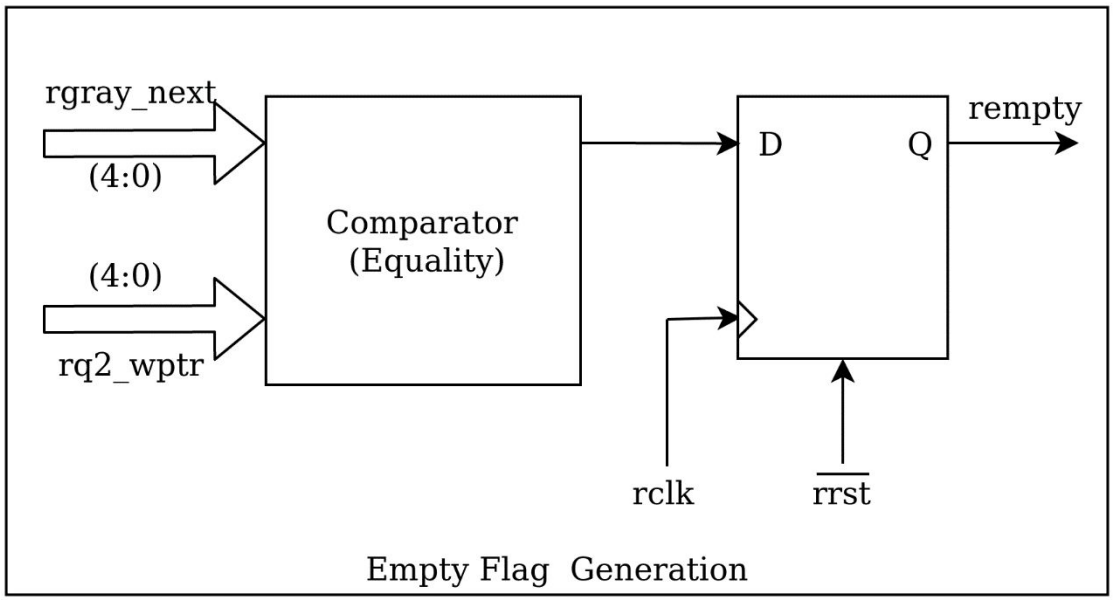
Fig: Read Address and Read Empty Flag Generator



read address generator : address to fetch data from DPRAM



read pointer generator : the pointer that will be sent to write clock domain



3. Synchronizer Read to Write Address :-

This module is used to synchronize the gray-coded read address with the write clock (i.e wclk). This is a 2-Stage Synchronizer module.

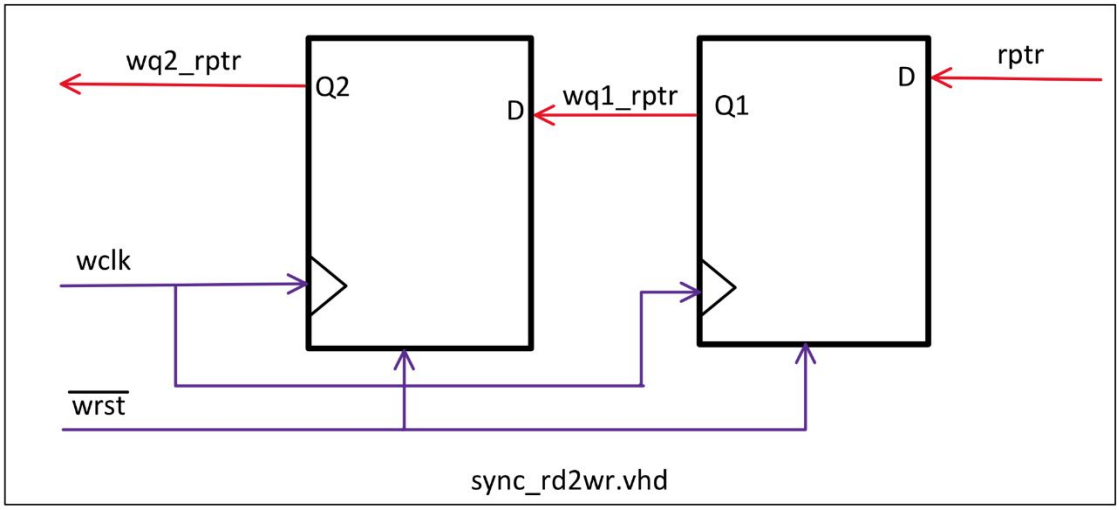
- *Signals in Synchronizer Read to Write Address Module:-*

wclk:- - Input write clock to write the data in DPRAM.

wrst :- It's a 1-bit I/p Active low reset to reset the write address module.

rptr :- It's a 5-bit I/p signal. It's the gray-coded value of read address coming from Read Address and Read Empty Flag Generator.

wq2_rptr :- It's a 5-bit O/p signal, generated by synchronizing rptr signal with wclk using 2-stage synchronizer.



4. Synchronizer Write to Read Address :-

This module is used to synchronize the gray-coded write address with the read clock (i.e rclk). This is a 2-Stage Synchronizer module.

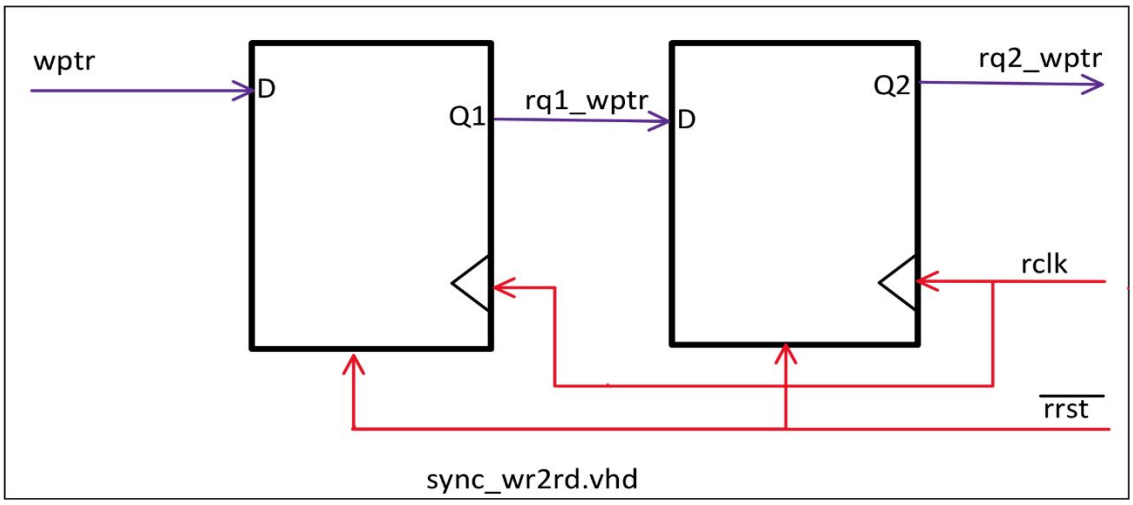
- *Signals in Synchronizer Write to Read Address Module:-*

rclk:- - Input read clock to read the data from DPRAM.

rrst :- It's a 1-bit I/p Active low reset to reset the read address module.

wptr :- It's a 5-bit I/p signal. It's the gray-coded value of write address coming from Write Address and Write Full Flag Generator.

rq2_wptr :- It's a 5-bit O/p signal, generated by synchronizing wptr signal with rclk using 2-stage synchronizer.



5. Dual-Port RAM (DPRAM) :-

We have instantiated a Dual port ram IP with the DEPTH=16 and WIDTH=8. In order to use this IP, 4-bit signals are required for write and read address location and 8-bit signals are required as write and read data throw the RAM.

SIMULATION RESULTS

We have designed this dual port FIFO for different as well as same port width. First of all we are going to simulate FIFO with same port width.

The module that we have designed here have **PORT WIDTH: 8** and **DEPTH: 16**. Here are the simulations results for this configuration. We are simulating 2 settings and will further refer these 2 settings as 1. EMPTY flag generation 2. Complete case(both FULL and EMPTY flag Generation):

1. Empty Flag Generation (We will be writing to 5 address of FIFO and then reading until empty flag gets ON). See the simulation from time 250 ns to 650 ns. Here first we will make the winc = 1 at about 250 ns to write data into the FIFO. Then we write to 5 address into FIFO . After that from about 440ns we make rinc = 1 to read the data from FIFO until EMPTY Flag is triggered at 640 ns.

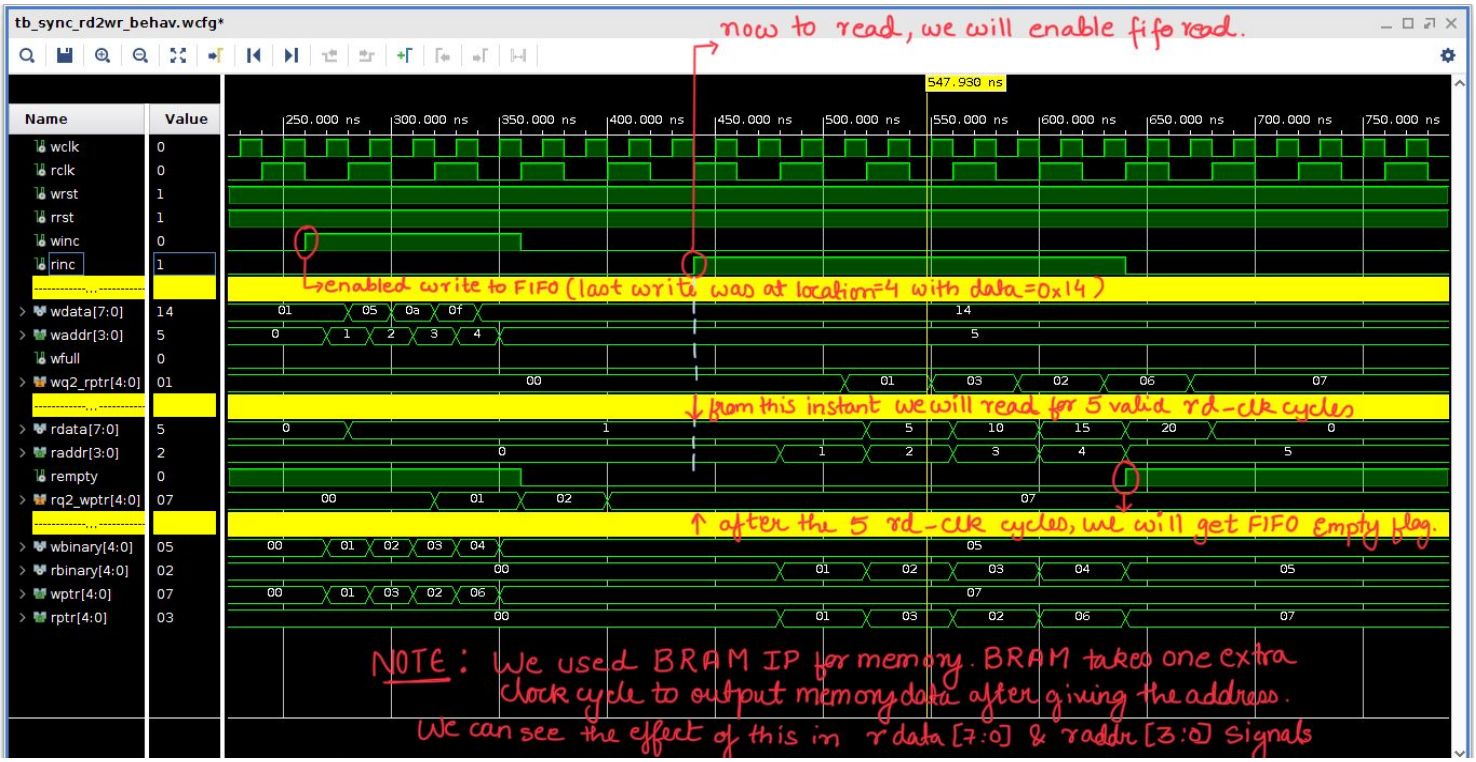


Fig: Empty Flag Generation

2. This is the complete case. Here we will first write to 16 addresses to trigger the FULL flag and then read from the 16 addresses to trigger the EMPTY flag. This case is shown with two screenshots.(**Complete case part 1** and **Complete case part 2**). So initially at 880 ns we will make the winc = 1 to write to 16 addresses which will trigger the full flag at 1190 ns. Then we make rinc = 1 at 1280 ns to read the data from the FIFO until the EMPTY flag is triggered at 1920 ns telling us that we have read all the data that was in FIFO.

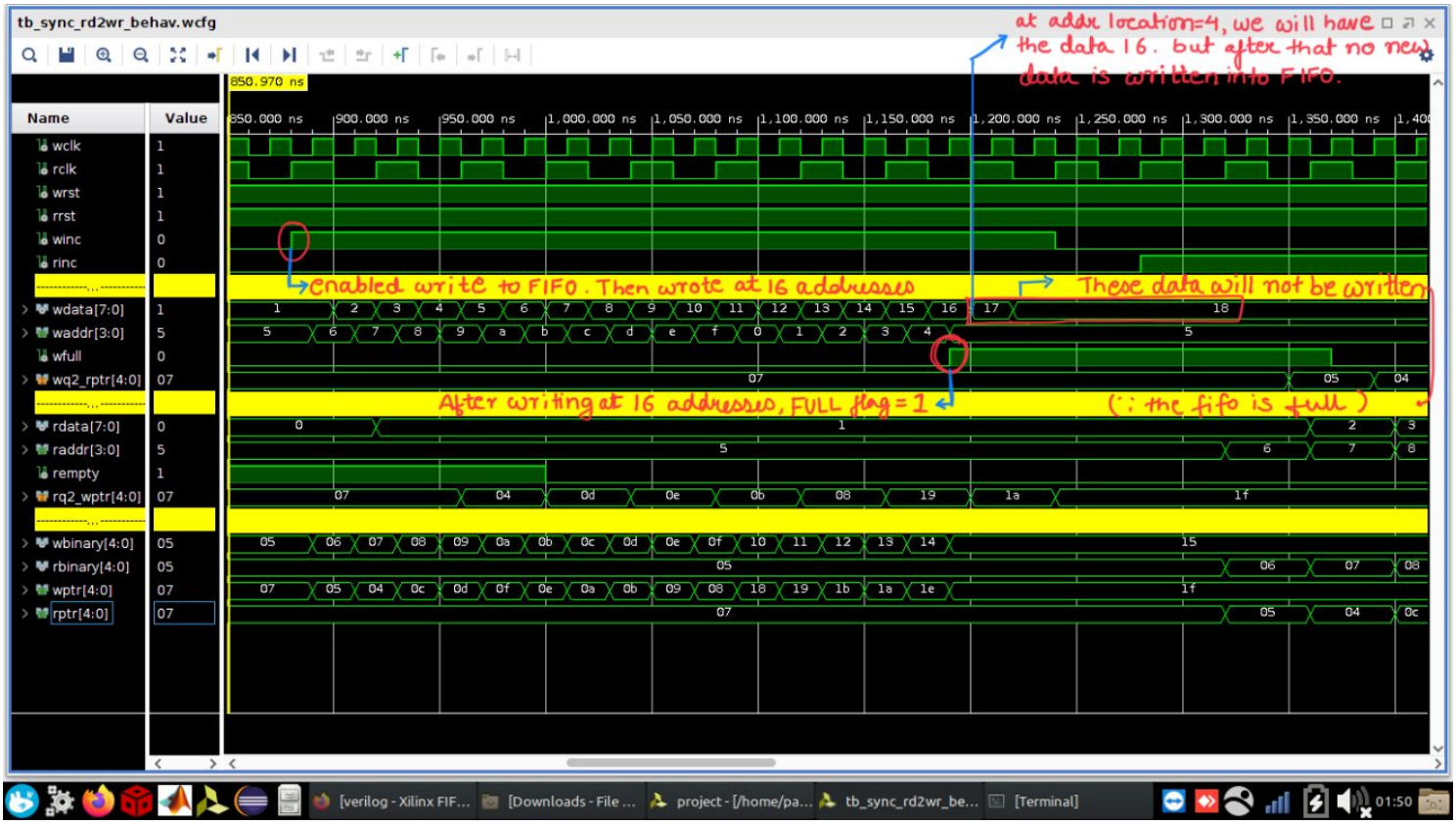


Fig: Complete Case part-1

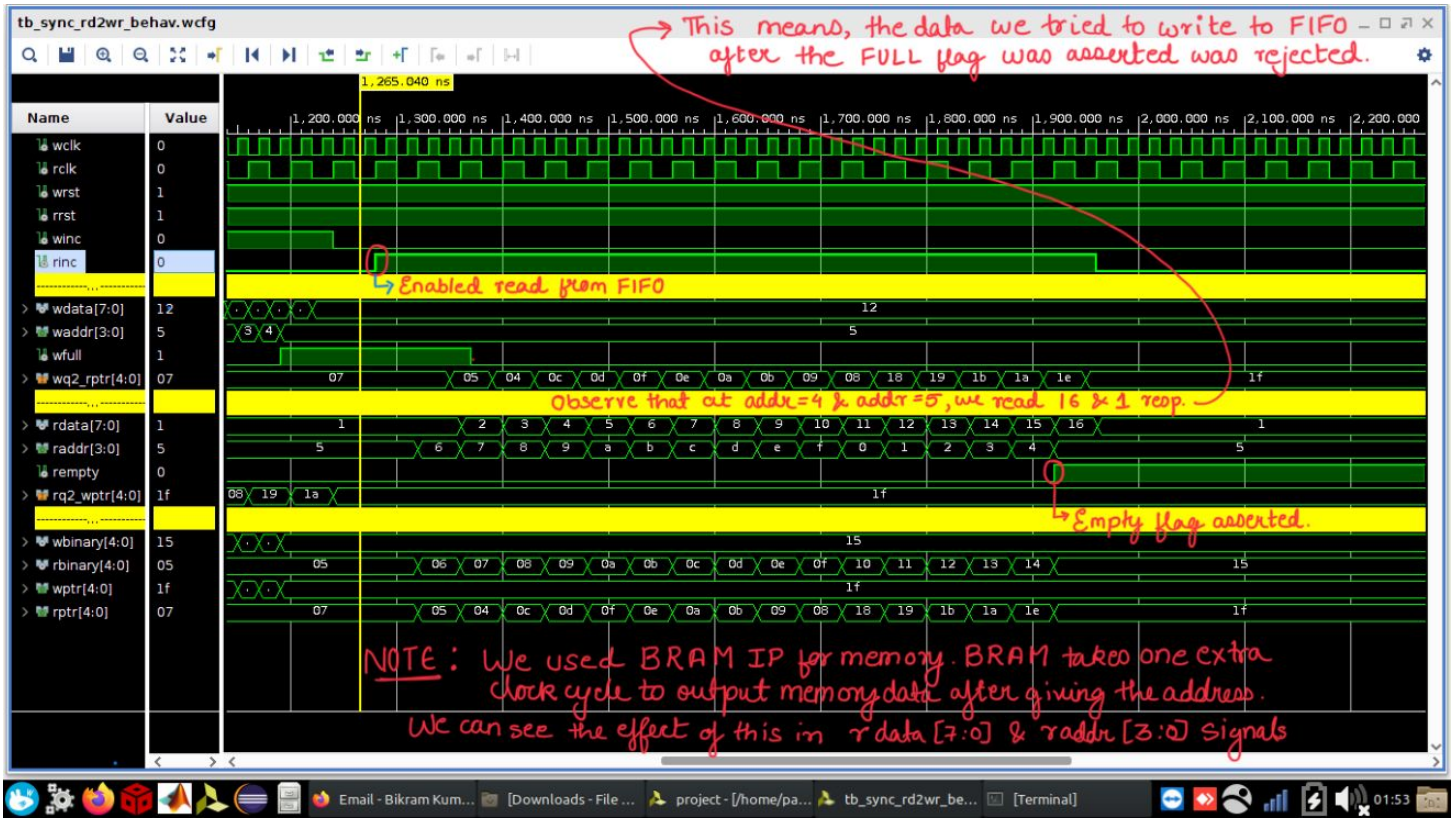


Fig: Complete Case part-2

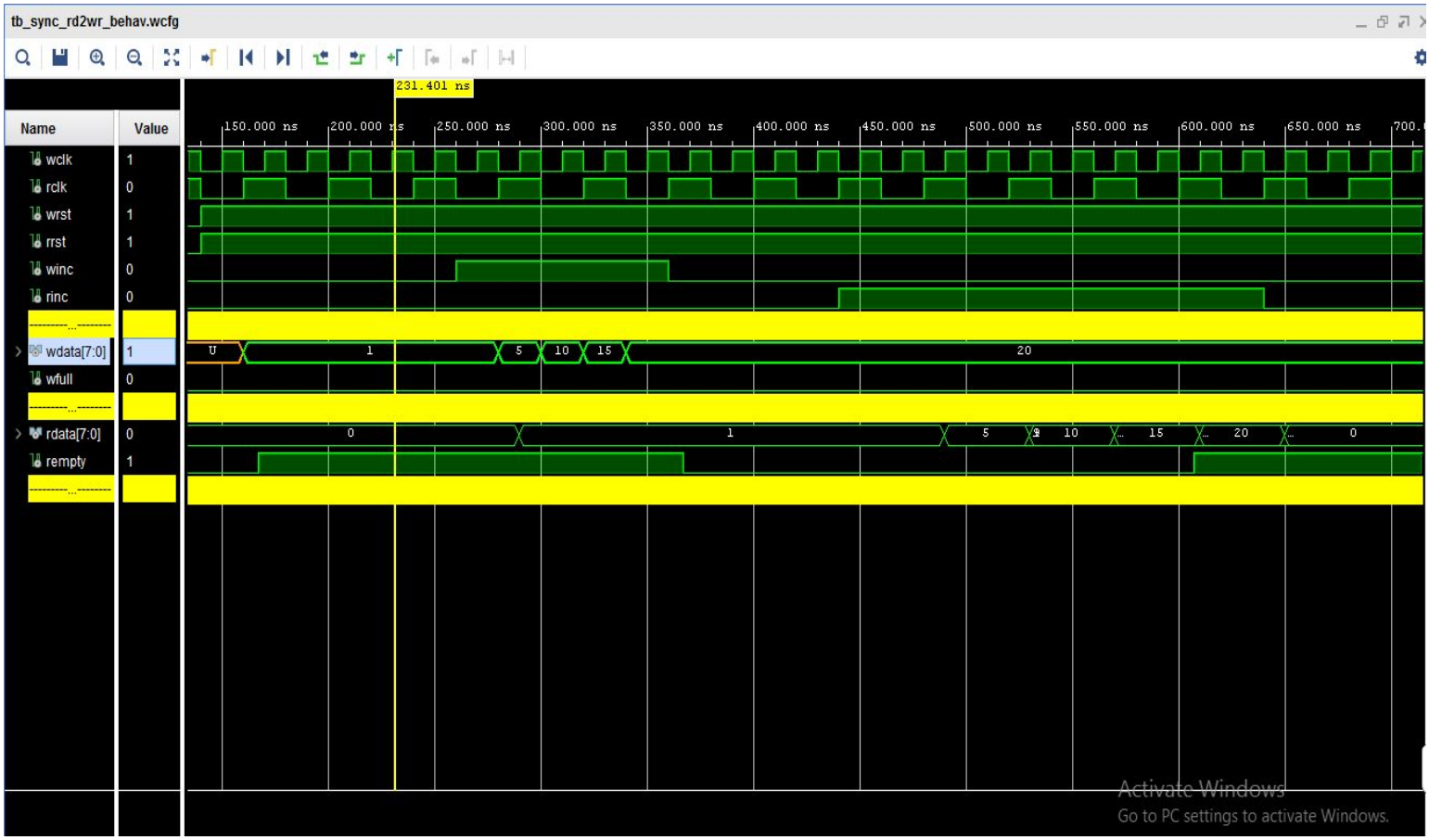


Fig: Post-Implementation Empty Flag Generation

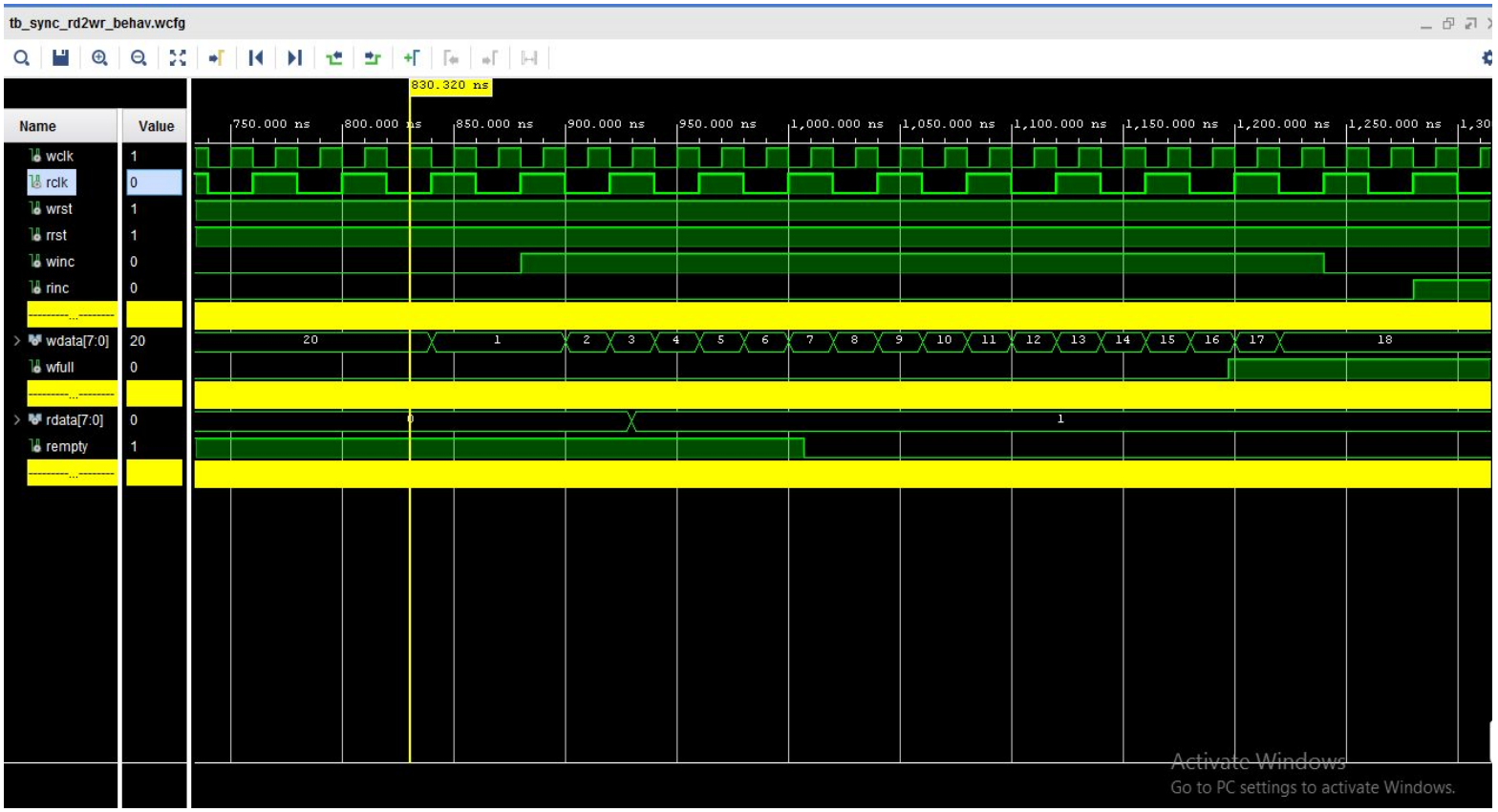


Fig: Post-Implementation Complete Case part-1

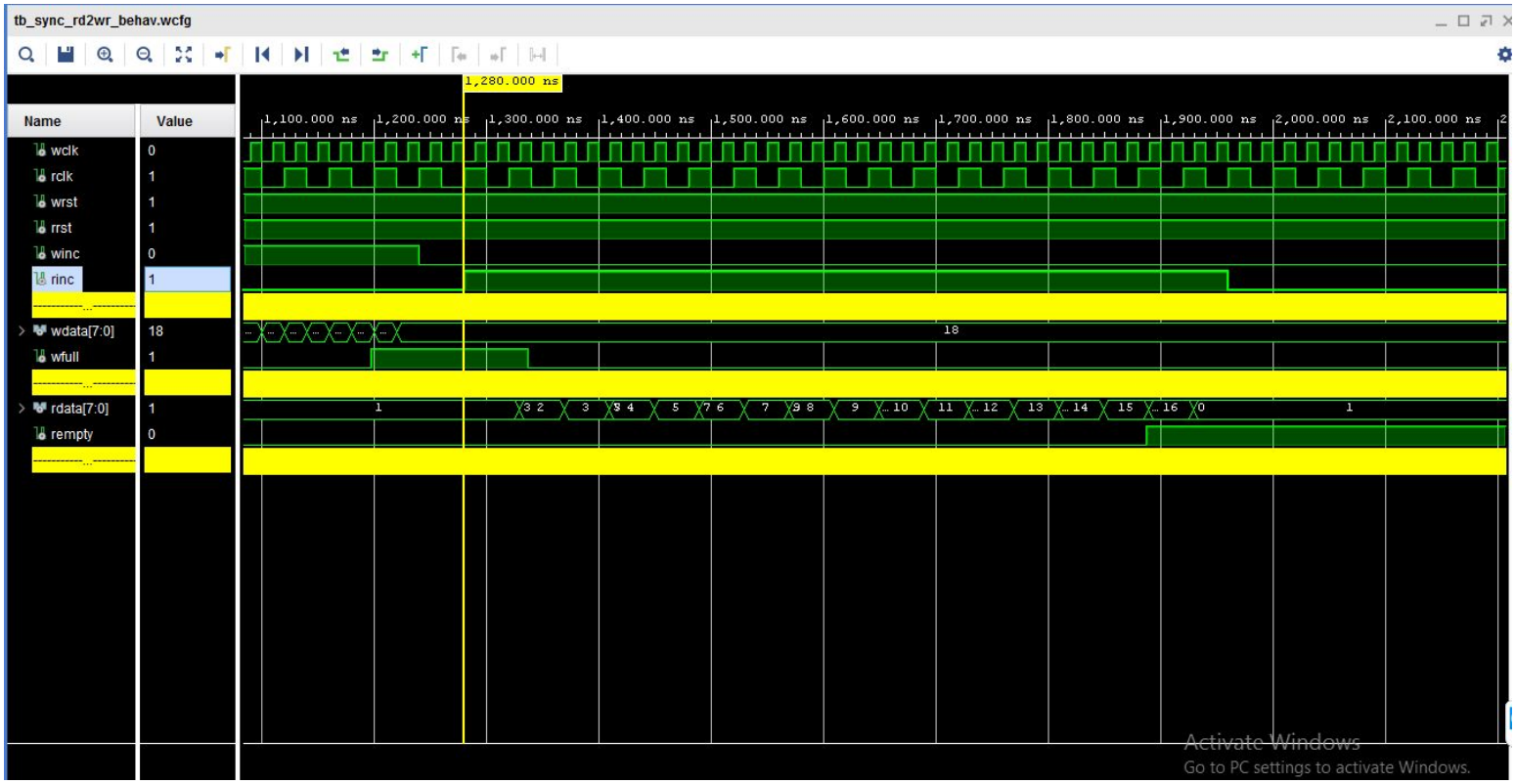


Fig: Post-Implementation Complete Case part-2

FIFO IMPLEMENTATION WITH DIFFERENT PORT SIZE

After Implementing Dual-Clock FiFo with the same Port Size we have done some changes in our design to Implement this for different Port sizes. Here, In this project we have designed different port size FIFO with these configuration:-

Write side PORT WIDTH: 8 ----- PORT-A

Read side PORT WIDTH: 4 ----- PORT-B

DEPTH: 16

What Changes have we done?

In order to bring our design to work for different port-width, some changes have been done. We are still using the same DPRAM with the same width as before (width=8), But now after reading from one memory location from the memory we are just reading only 4

bits at a time on a single read clock pulse. Next 4-bit of the same location is sent to the output port of the module in the 2nd read clock pulse. This can be further understood as this:-

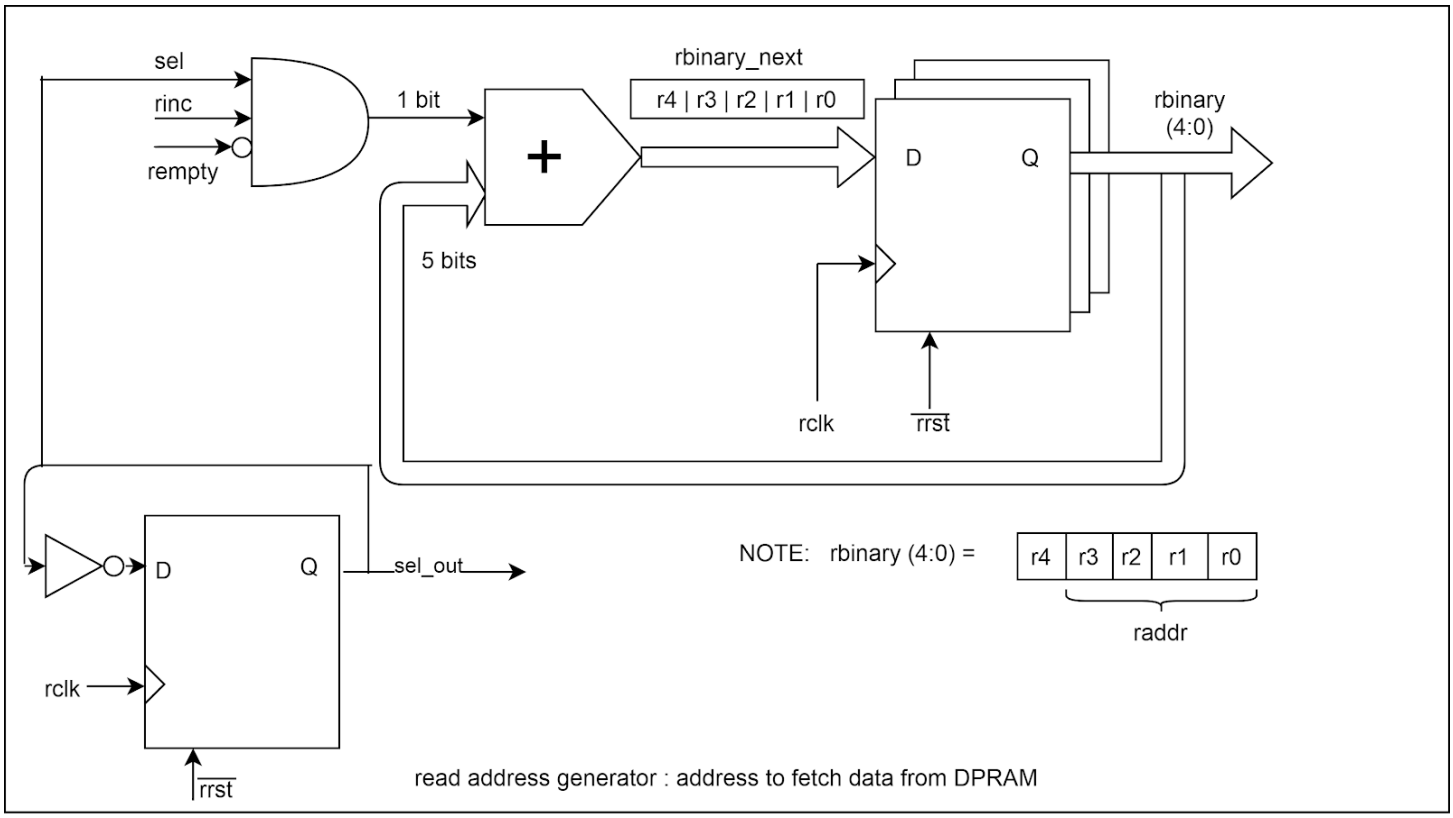
1. When a write clock pulse is triggered in the module, 8-bit input data(wdata) has been saved to the current memory address of DPRAM and address is incremented on every write data clock pulse.
2. When a read clock pulse is triggered in the module, lower 4-bits of current read address location has been sent to the output port of the module but the read address will not increment on this single read clock pulse for respective location. When the 2nd read clock pulse gets encountered, higher 4-bits of that current read location will be further sent to the output read port and then the read counter will increment and point to the next read location.

   In this way we are taking 8-bit at each write clock pulse and throwing only 4-bit at every read clock pulse at the output read port of the module.

Changes In Module-Hardware:-

In order to bring the above explained logic in hardware design we have modified our VHDL code. The main changes that we have done are:-

1. Changed the output read data port size to 4 bits. (i.e rdata[3:0]).
2. All the sub-modules are working exactly the same as before except there are some changes in Read Address and Read Empty Flag Generator. New output signal has been added to this module named **sel_out.** This signal then goes into the DPRAM module. It will work as a select line of mux that is sending lower nibble and higher nibble to the output read port. This sel_out has been toggled at every read clock pulse. This signal will hold the single read address for two read clock pulses as shown below.
3. When this select signal goes into DPRAM module, 8-bit data will then be read from the DPRAM at each read clock pulse and lower and higher nibble of this data is sent to 2x1 MUX whose select line is this sel_out signal, coming from the read address generator module.



SIMULATION RESULTS

As in earlier cases, we have simulated this updated module for the same testbench. Here, we are explaining the behavioral simulation for those 3 cases that as we did explain for the same port size FIFO module. (Emply, Complete case-1, Complete case-2)

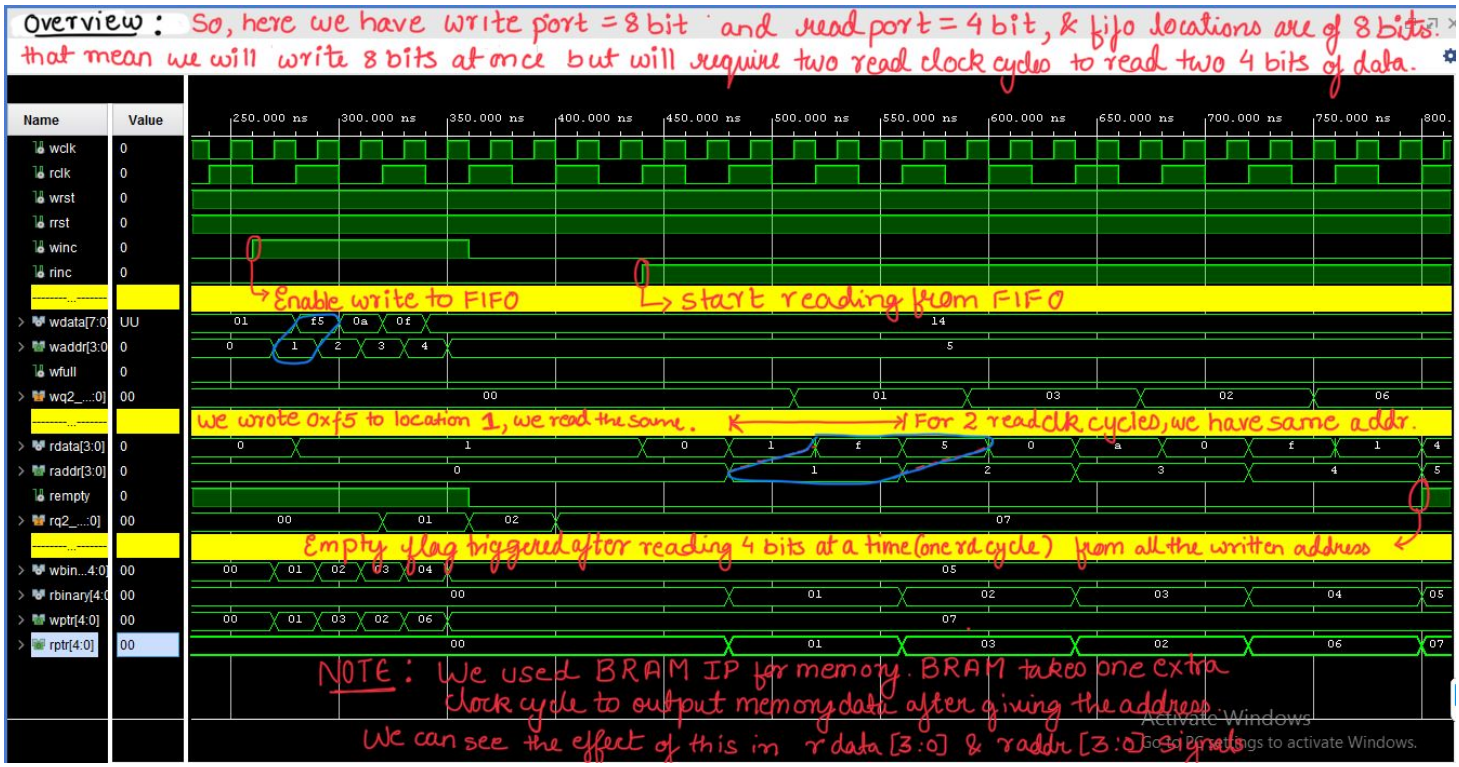


Fig: Empty Flag Generation for Different Port Size FIFO

Fig: Complete Case part-1 for Different Port Size FIFO

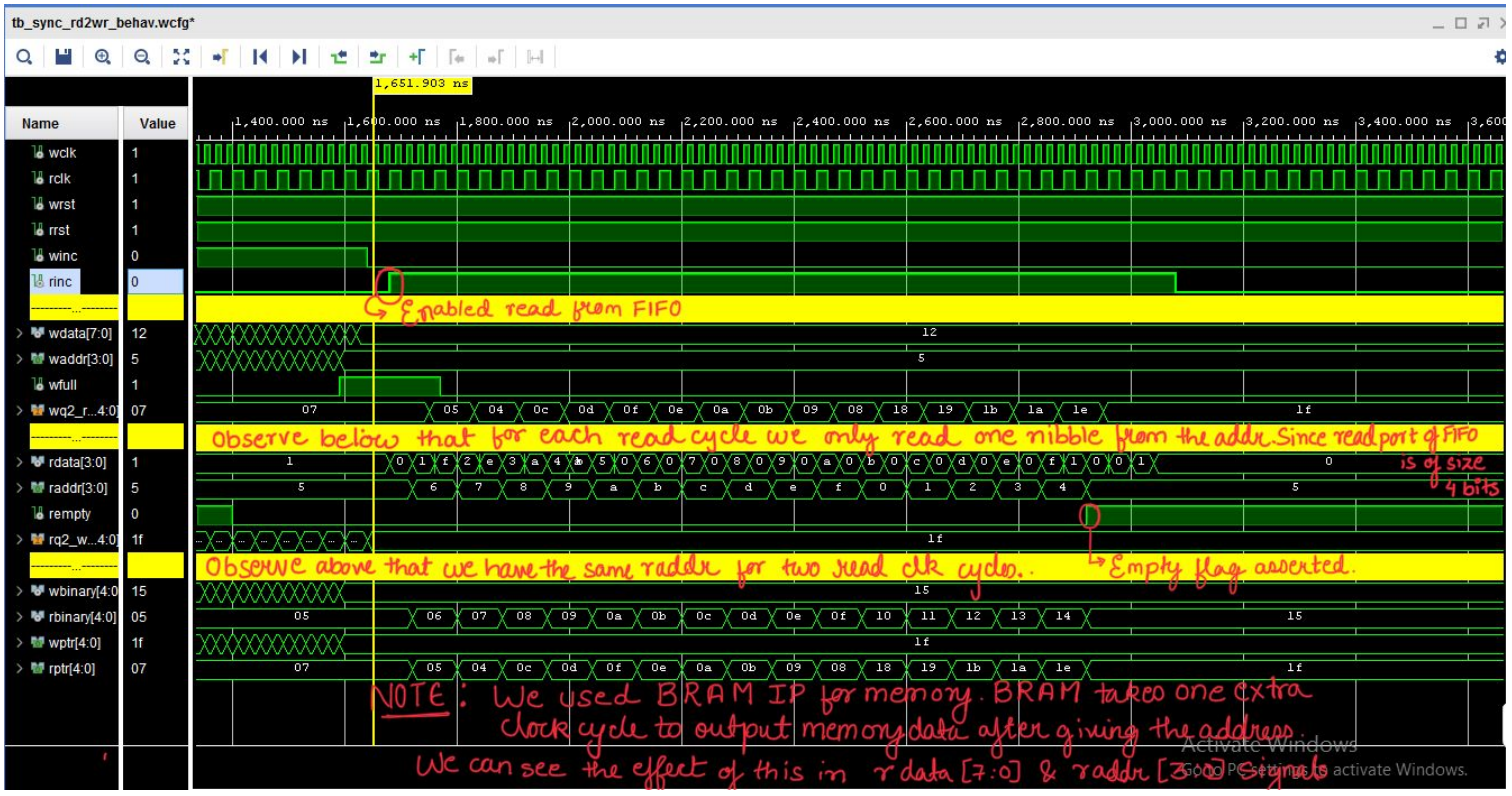


Fig: Complete Case part-2 for Different Port Size FIFO

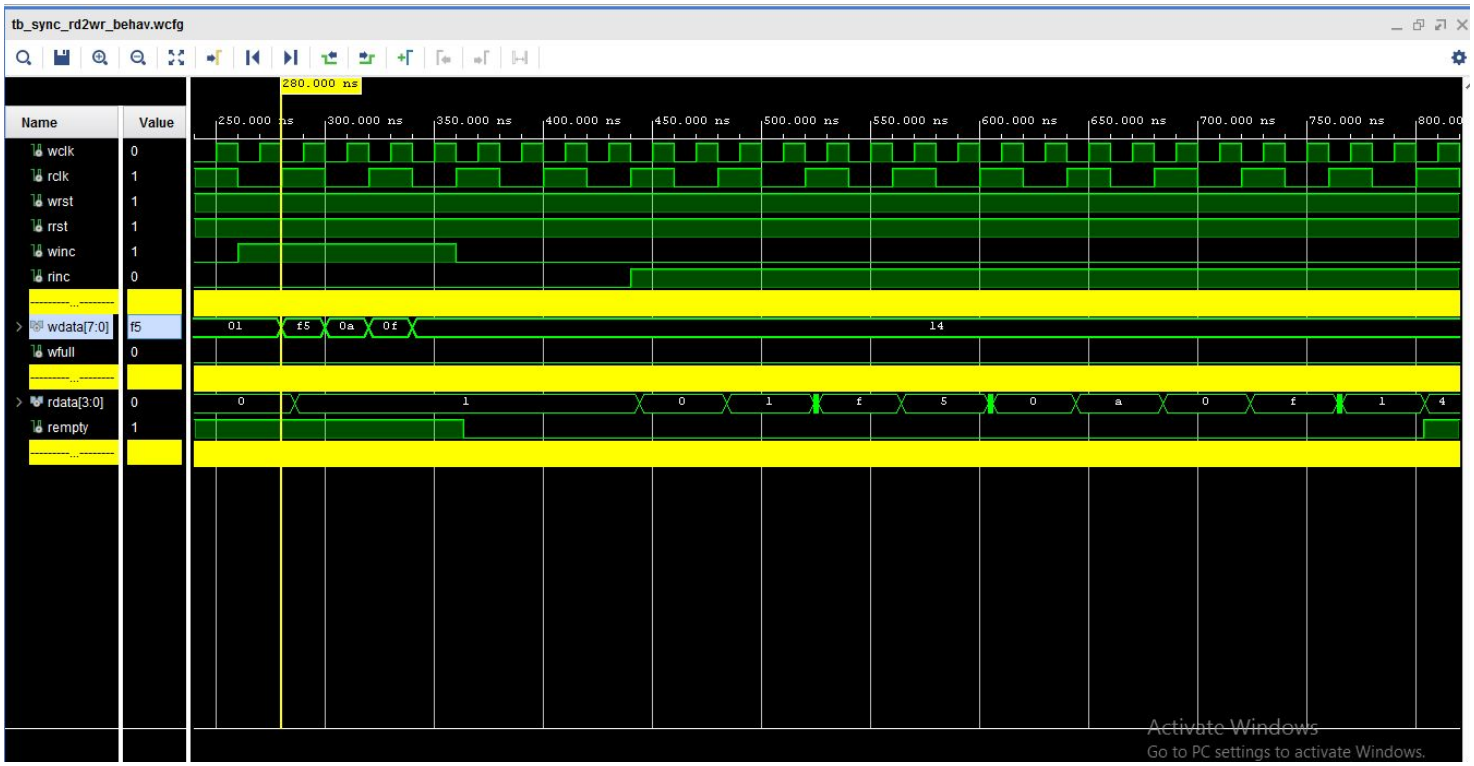


Fig: Post-synthesis simulation for Read Empty Flag

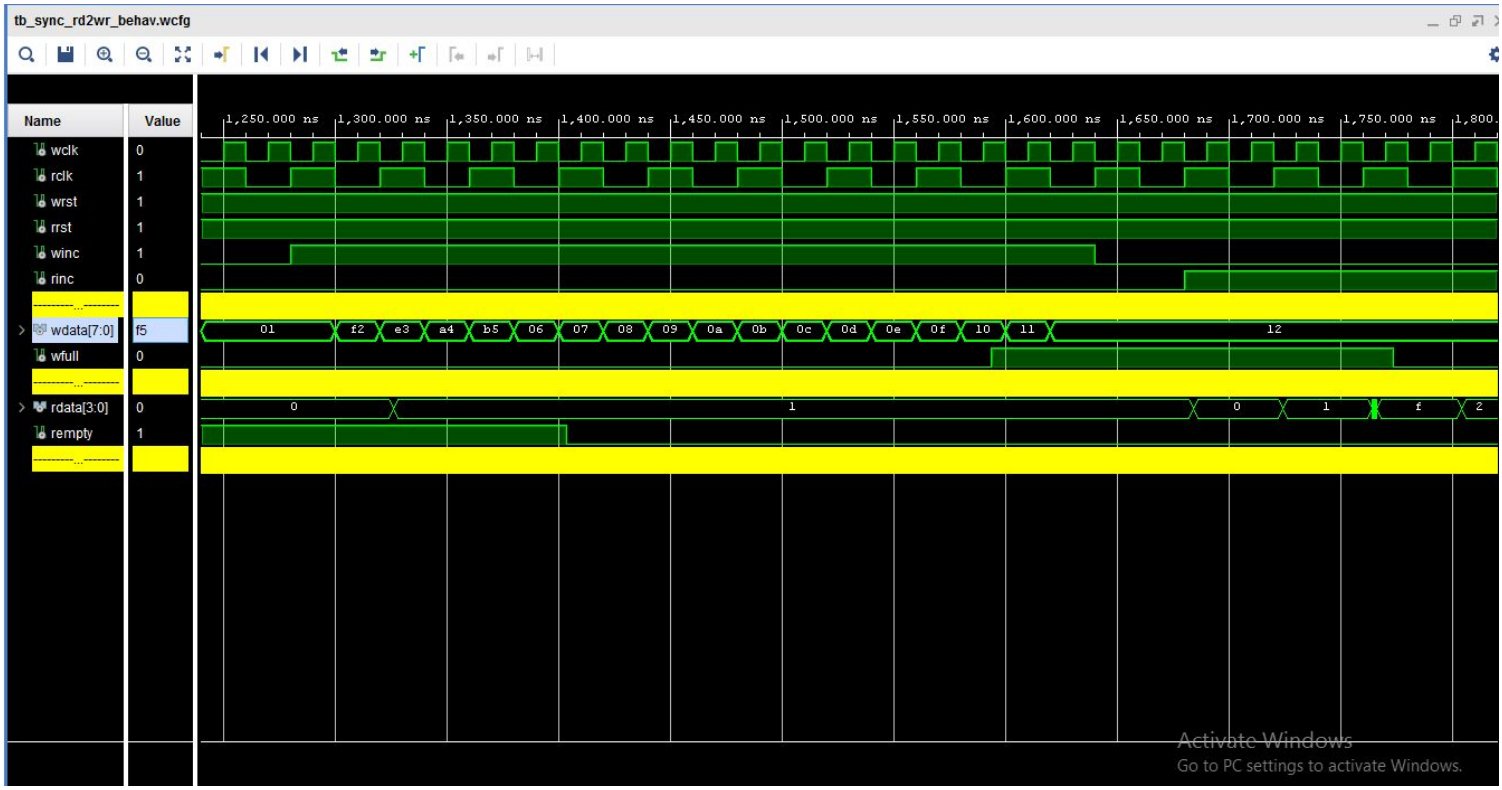


Fig: Post-synthesis simulation for Complete Case part-1

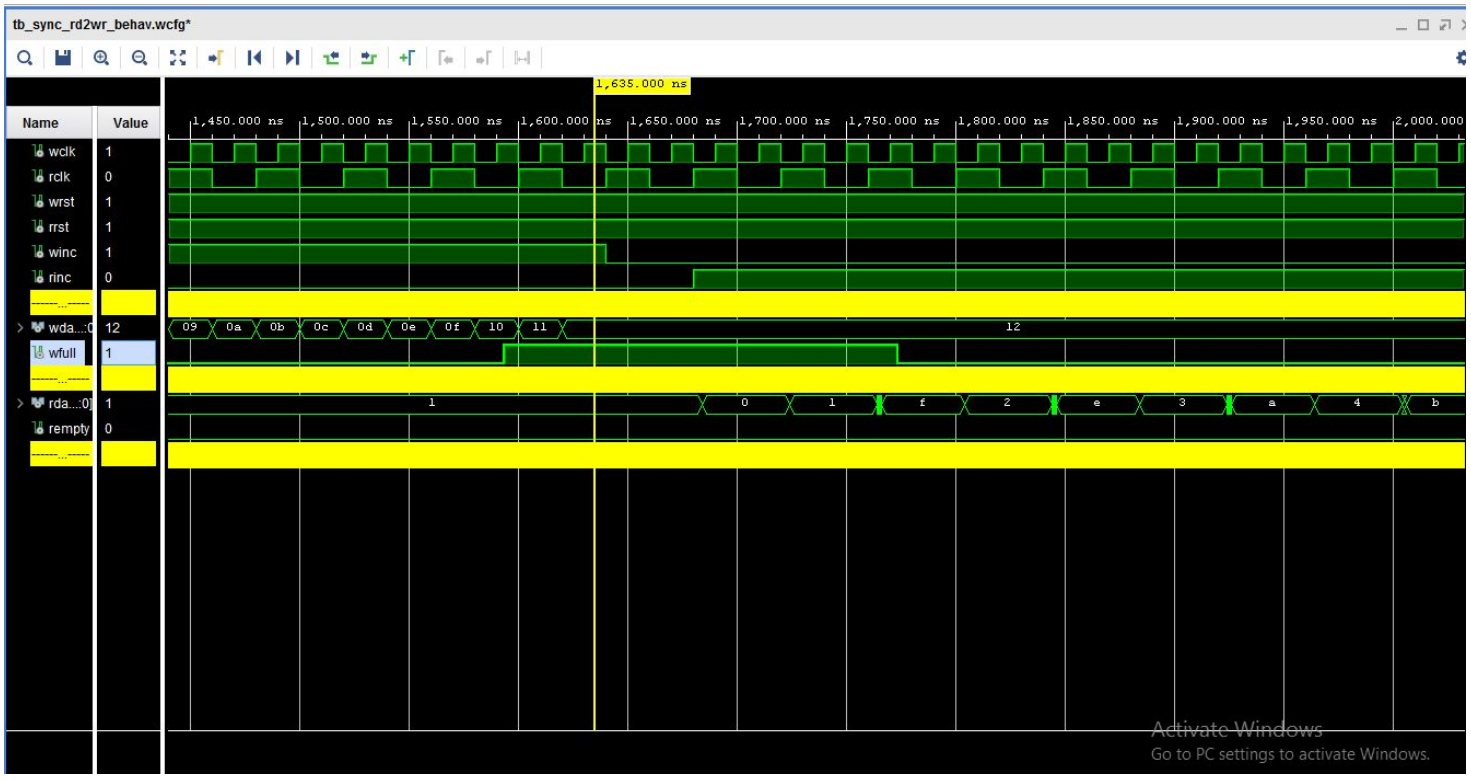


Fig: Post-synthesis simulation for Complete Case part-2

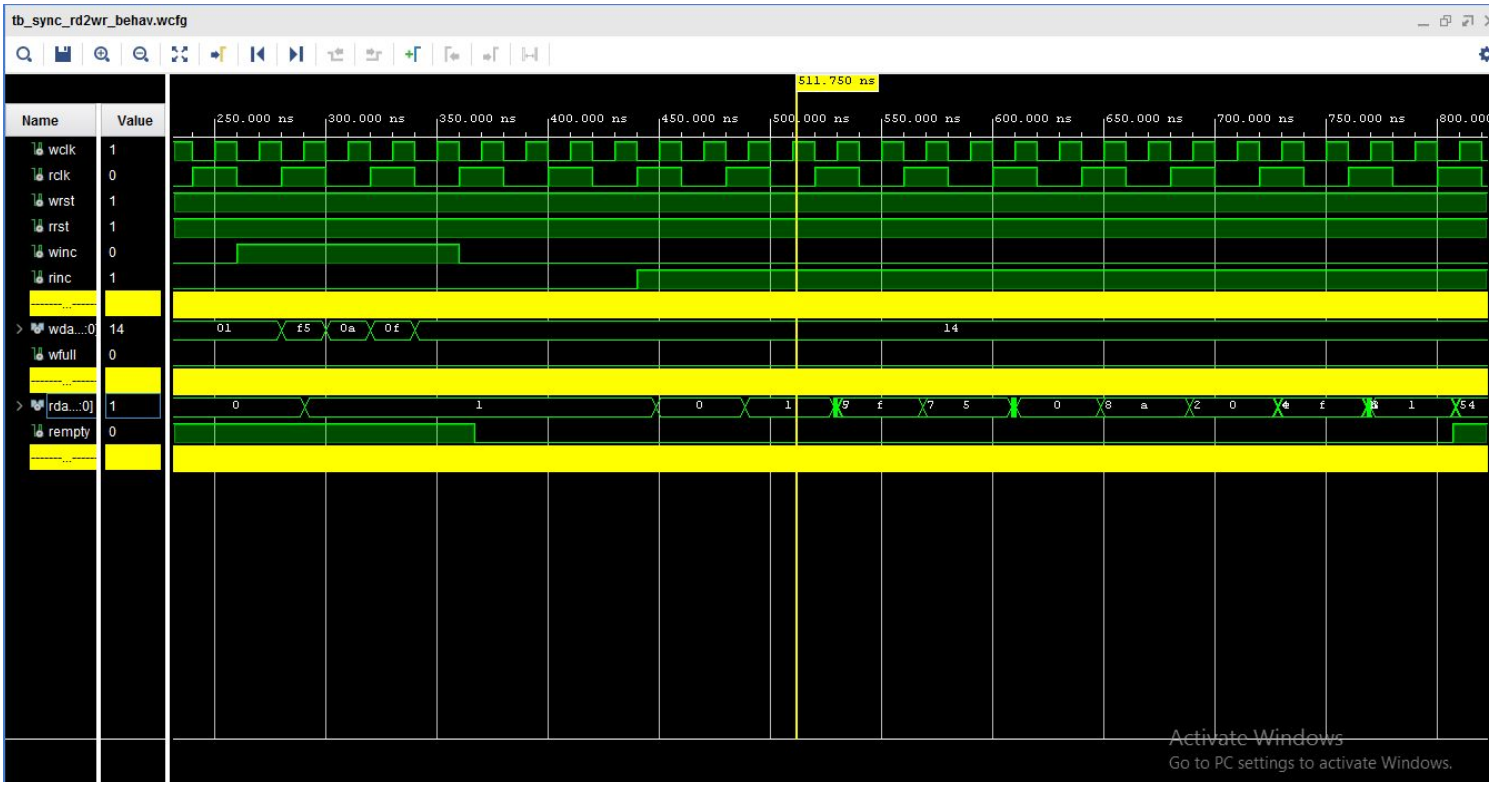


Fig: Post-Implementation simulation for Read Empty Flag

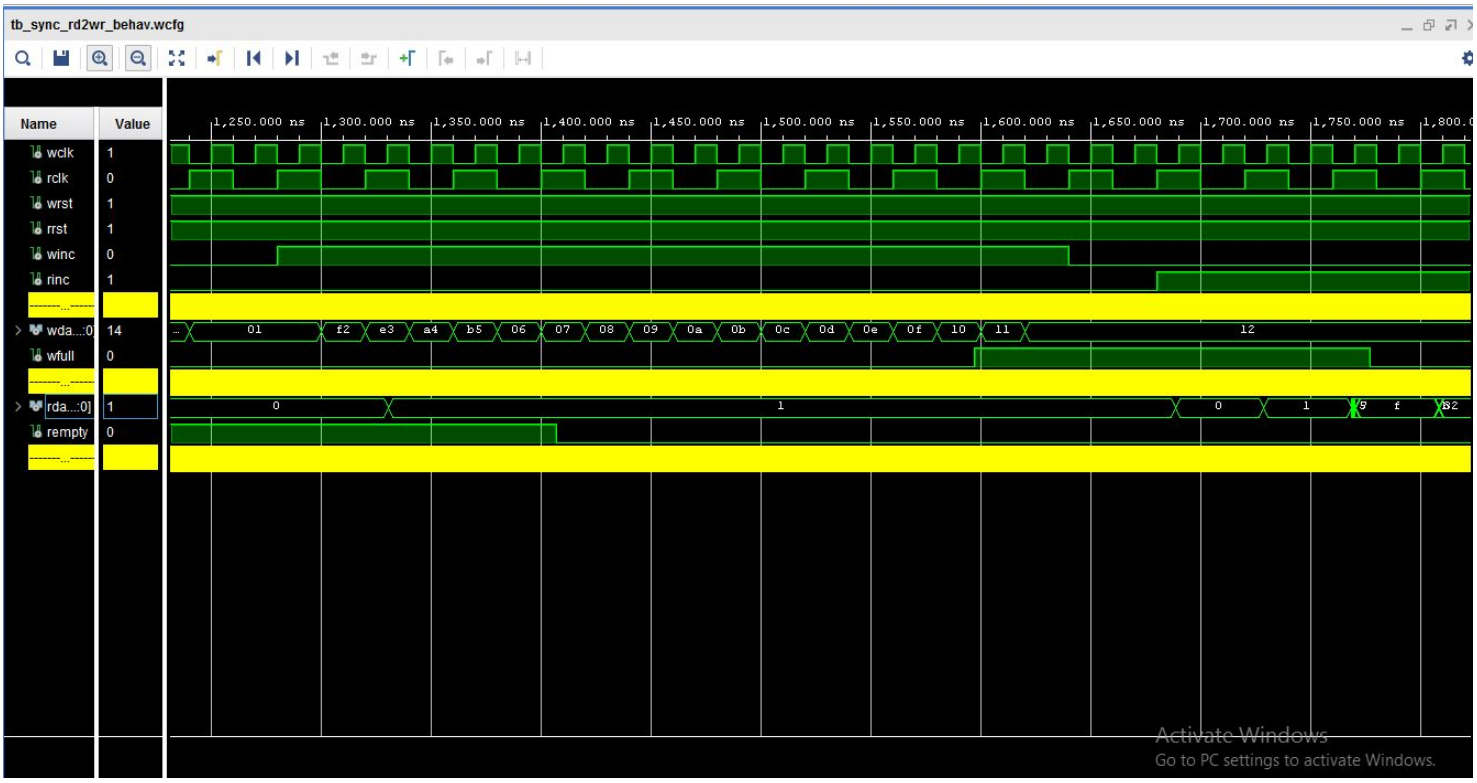


Fig: Post-Implementation simulation for Complete Case part-1

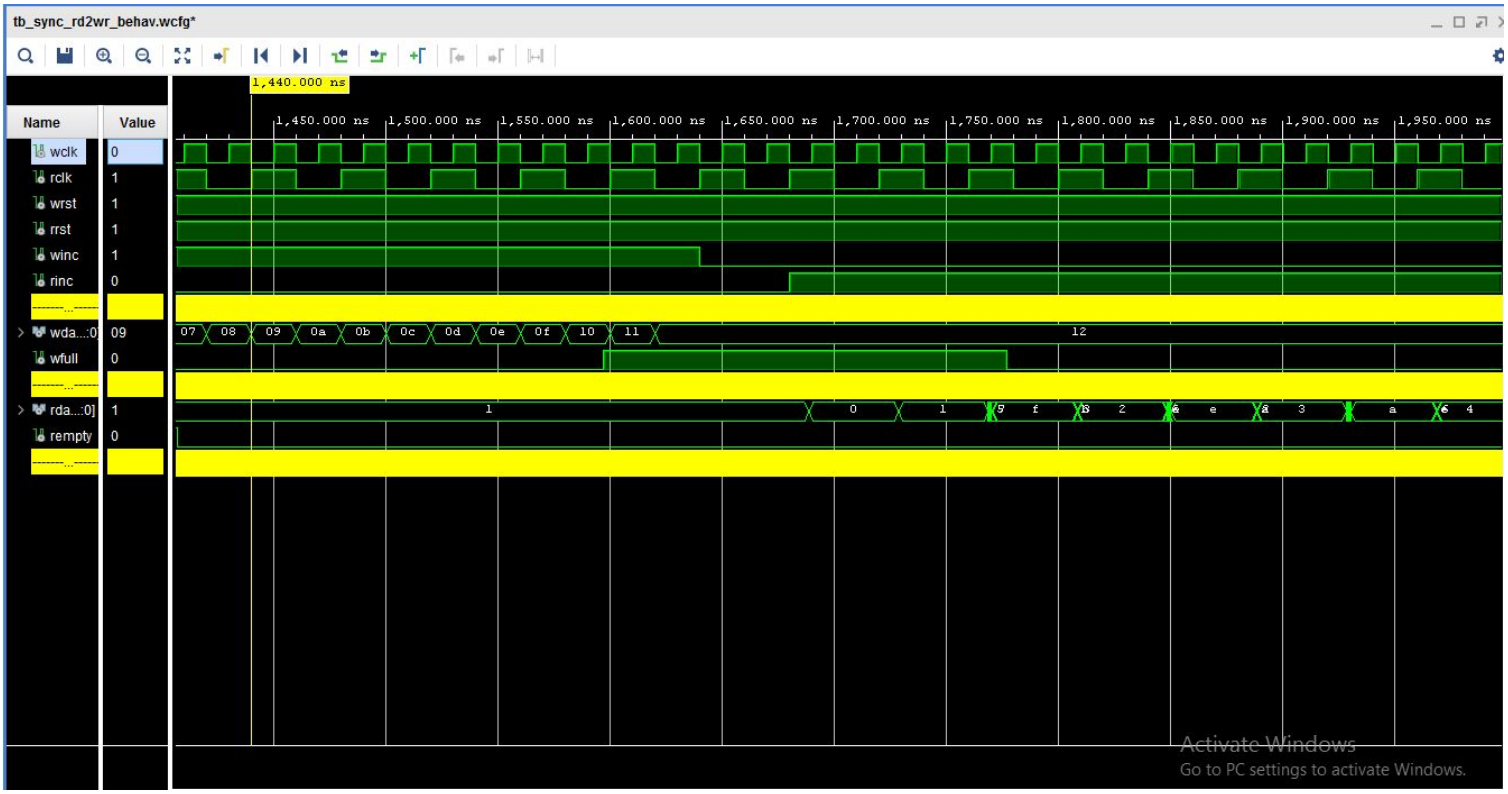


Fig: Post-Implementation simulation for Complete Case part-2

Utilization Report And Schematics

Resource	Utilization	Available	Utilization %
LUT	18	20800	0.09
FF	40	41600	0.10
BRAM	0.50	50	1.00
IO	24	106	22.64

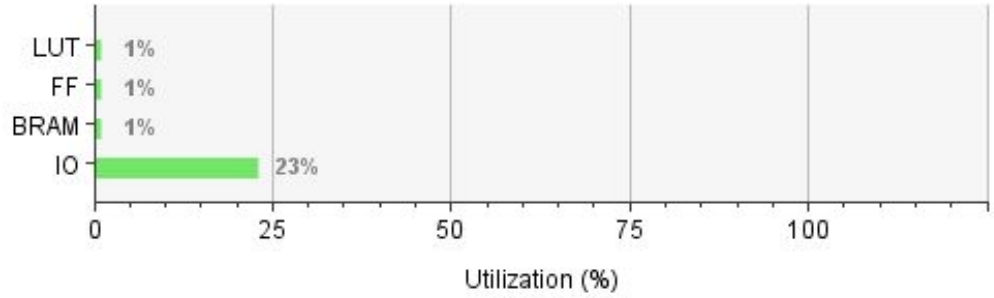


Fig: Utilization Report for Same Port Size FIFO

Resource	Utilization	Available	Utilization %
LUT	22	20800	0.11
FF	41	41600	0.10
BRAM	0.50	50	1.00
IO	20	106	18.87

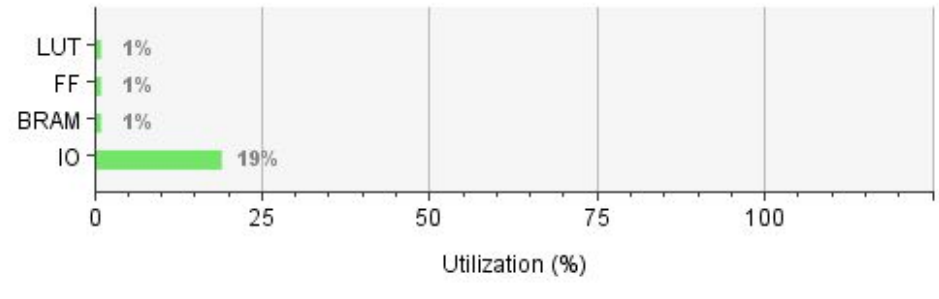


Fig: Utilization Report for Different Port Size FIFO

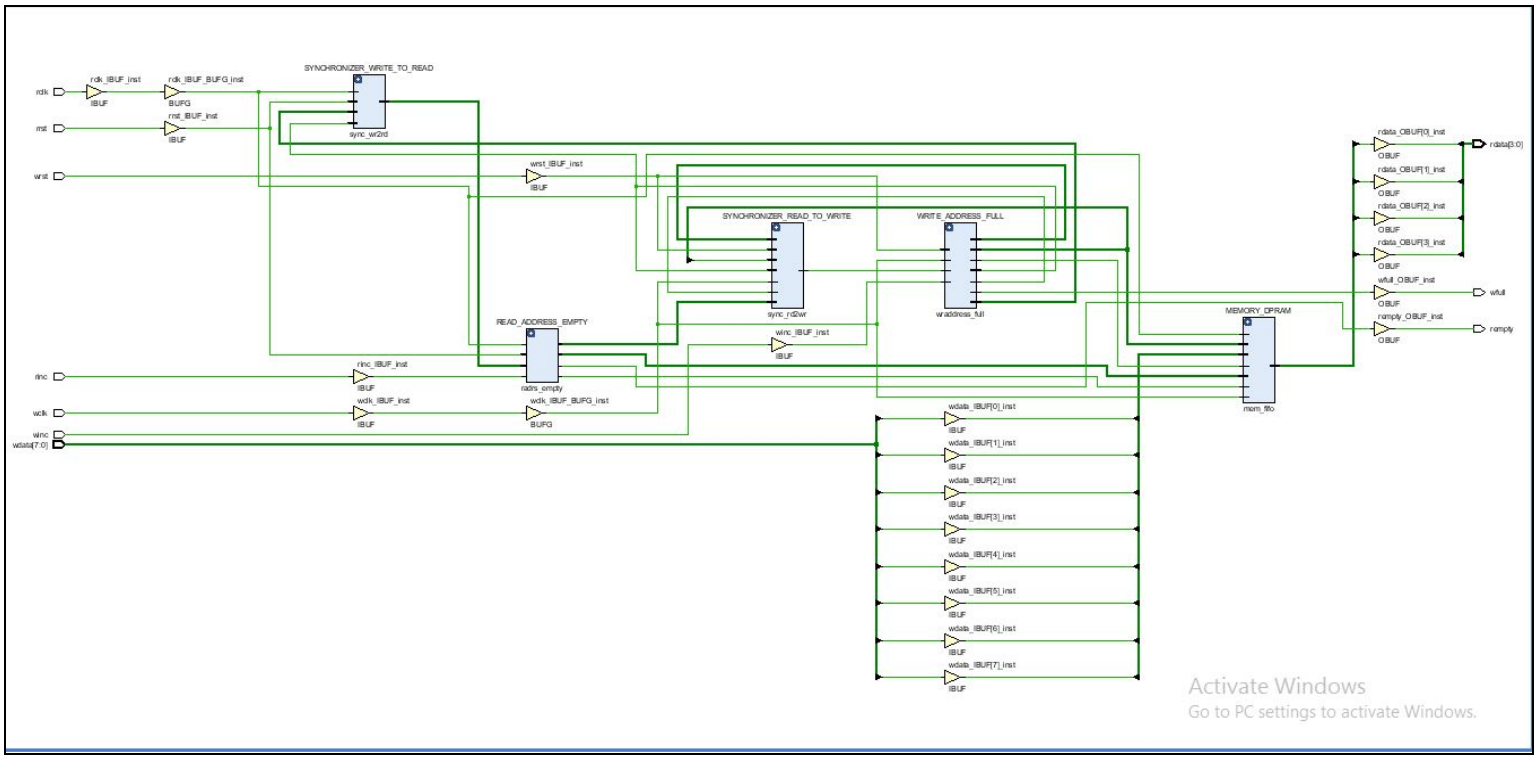


Fig: Schematic Generated (High resolution Image is also shared with report for both cases.)

TIMING ANALYSIS REPORT

(timing report for both cases is included in the project folder)

The Incorrect Analysis:-

Initially when we did the timing analysis, we got Total negative HOLD slack, to counter this , we inserted buffers(instantiated IBUFs to increase the delay and thus reduce the HOLD slack).

This way we tried removing the total hold slack in the synthesis process but it was of no use since the IBUFs we instantiated got removed due to optimization in the implementation process giving us back our HOLD timing violation.

The correct Analysis:-

As we are doing Clock Domain Crossing, we have to mark the CDC paths as false paths, thus effectively not evaluating the timing constraints on those path.So we did that and could successfully implement the FIFO design without any implementation ERROR.

CONCLUSION

We implemented FIFO using two configurations.

1. **Same Write and Read port size.**
2. **Different Write and Read port size.**

We also learnt how to send data across two different clock domains reliably.

REFERENCES:

Clifford E. Cummings, "Simulation and Synthesis Techniques for Asynchronous FIFO Design,"

SNUG 2002 - www.sunburst-design.com/papers/CummingsSNUG2002SJ_FIFO1.pdf