NAME-Chhandak Roy ROLL-244102404 MTECH VLSI

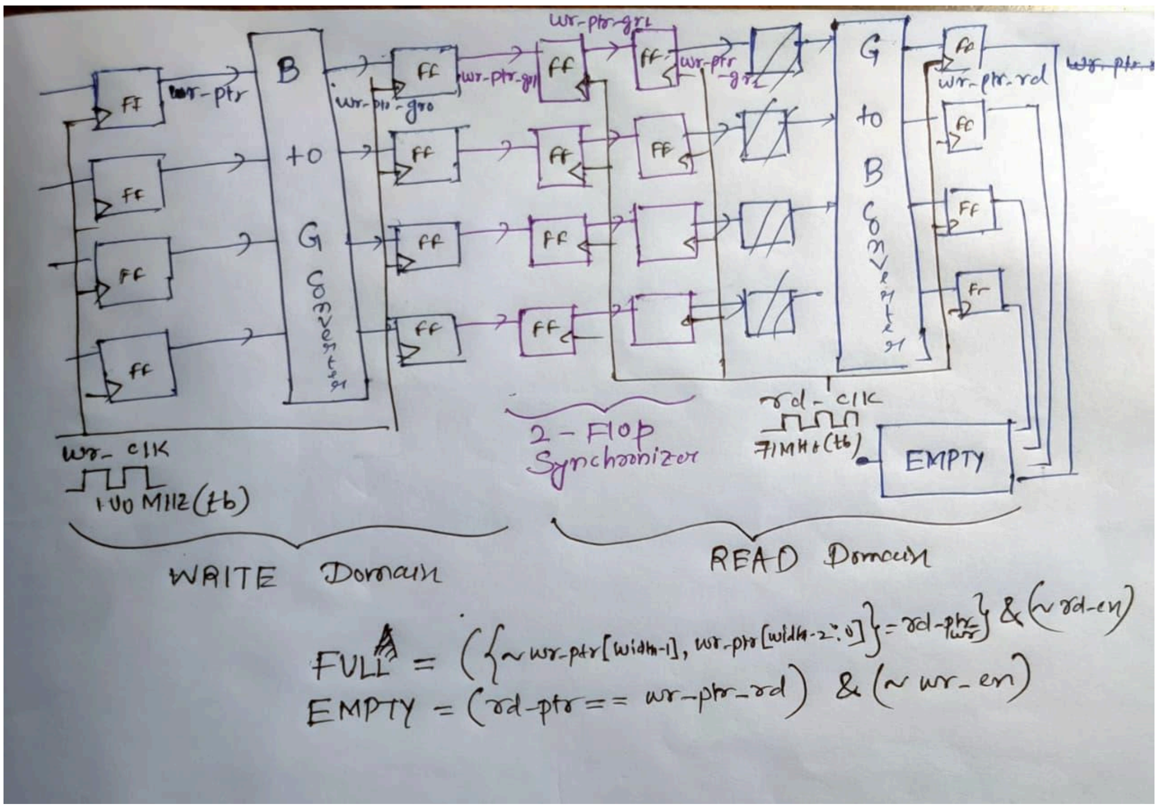
Asynchronous FIFO

- ## Block Diagram & Overview

The FIFO has been parameterized in DEPTH and WIDTH dimension.
The Asynchronous FIFO does Read and Write ops in 2 different clock cycles, rd_clk and wr_clk, because of this Clock Domain crossing techniques use to update FULL and EMPTY condition, here we have employed the following parameters to avoid metastability and corrupt result due to domain crossing:

1. *Binary to Gray converter .*
2. *2-Flop synchronizer.*

We have transferred wr_ptr to read domain to generate Empty and rd_ptr to write domain to generate Full condition.



Wr_ptr to read domain via 2 flop synchronizer

The entire Verilog Code follows this Architecture or Data Flow Model, similarly the rd_ptr is transferred to write domain as shown in the above figure.

NOTE:

When a Read operation is done, the Fifo replaced the address from where the data has been read to 'x' just for mere ease in visual representation in simulation.

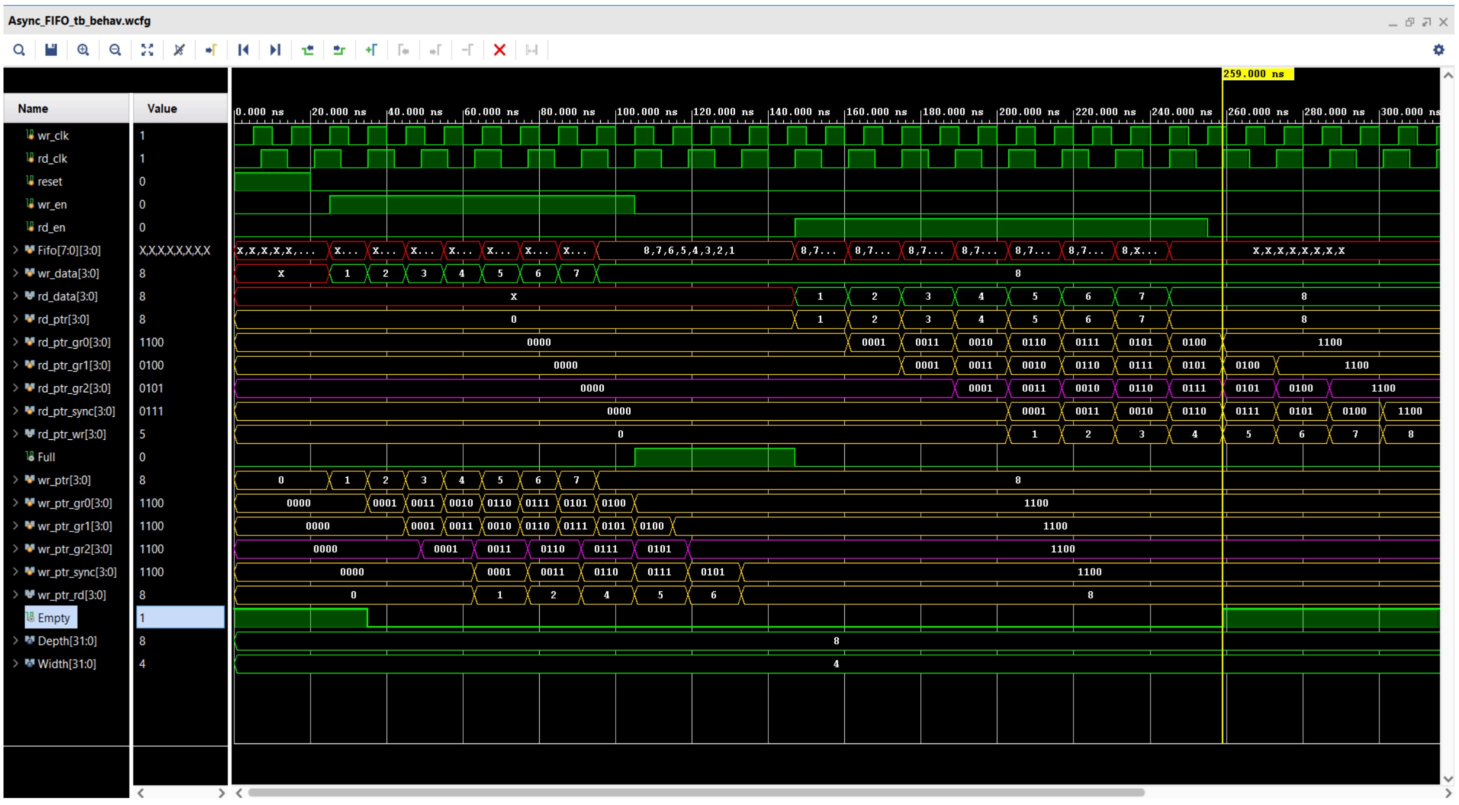
- ## **Results and Observation**

The Simulation results has been generated by the following test bench:

```
module Async_FIFO_tb;

 // Parameters
 parameter Depth = 8;
 parameter Width = 4;

 // Testbench signals
 reg wr_clk, rd_clk;
 reg reset;
 reg wr_en, rd_en;
 reg [Width-1:0] wr_data;
 wire [Width-1:0] rd_data;
 wire Full, Empty;

 // DUT instantiation
 Async_FIFO #(Depth, Width) dut (
  .wr_en(wr_en),
  .rd_en(rd_en),
  .wr_data(wr_data),
  .wr_clk(wr_clk),
  .rd_clk(rd_clk),
  .reset(reset),
  .Full(Full),
  .Empty(Empty),
  .rd_data(rd_data)
 );

 // Clock generation
 initial begin
  wr_clk = 0;
  forever #5 wr_clk = ~wr_clk;   // write clock = 100 MHz (10ns period)
 end

 initial begin
  rd_clk = 0;
  forever #7 rd_clk = ~rd_clk;   // read clock = ~71 MHz (14ns period)
 end


 initial begin
  // Initialize signals
  reset   = 1;
  wr_en   = 0;
  rd_en   = 0;


  // Release reset
  #20 reset = 0;
```

```
// Write data into FIFO
@(posedge wr_clk);
wr_en   = 1;
wr_data = 1;

@(posedge wr_clk);
wr_data = 2;

@(posedge wr_clk);
wr_data = 3;


@(posedge wr_clk);
wr_data = 4;


@(posedge wr_clk);
wr_en   = 1;
wr_data =5;

@(posedge wr_clk);
wr_data = 6;


@(posedge wr_clk);
wr_data = 7;

@(posedge wr_clk);
wr_data = 8;

@(posedge wr_clk);
wr_en   = 0; // stop writing

// Wait before reading
#30;

  @(posedge rd_clk);
rd_en = 1;

@(posedge rd_clk);
rd_en = 1;

 @(posedge rd_clk);
rd_en = 1;

@(posedge rd_clk);
rd_en=1;

 @(posedge rd_clk);
rd_en = 1;

@(posedge rd_clk);
rd_en=1;
```

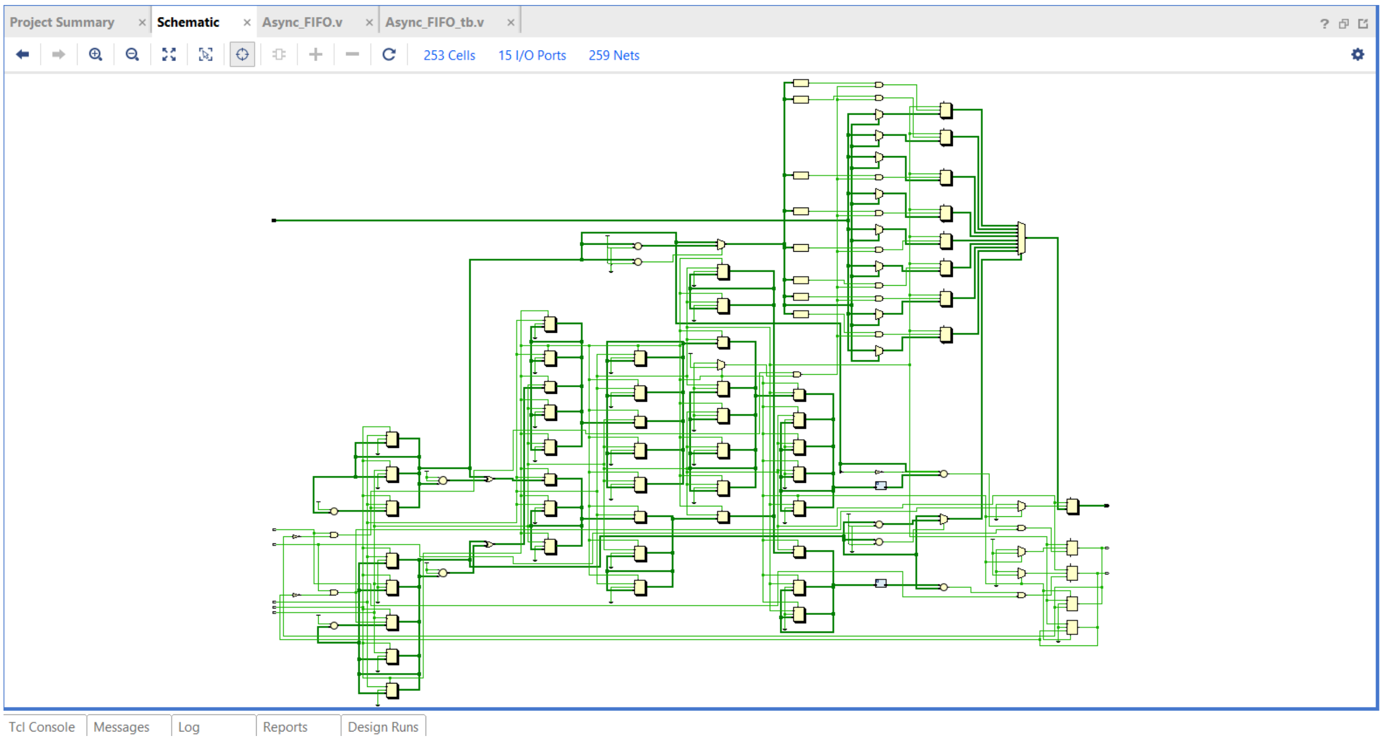
```
   @(posedge rd_clk);
  rd_en = 1;

  @(posedge rd_clk);        // FIFO should be empty here finally
  rd_en = 1;


  #10
  rd_en = 0;

  #100
  $finish;

 end

endmodule
```



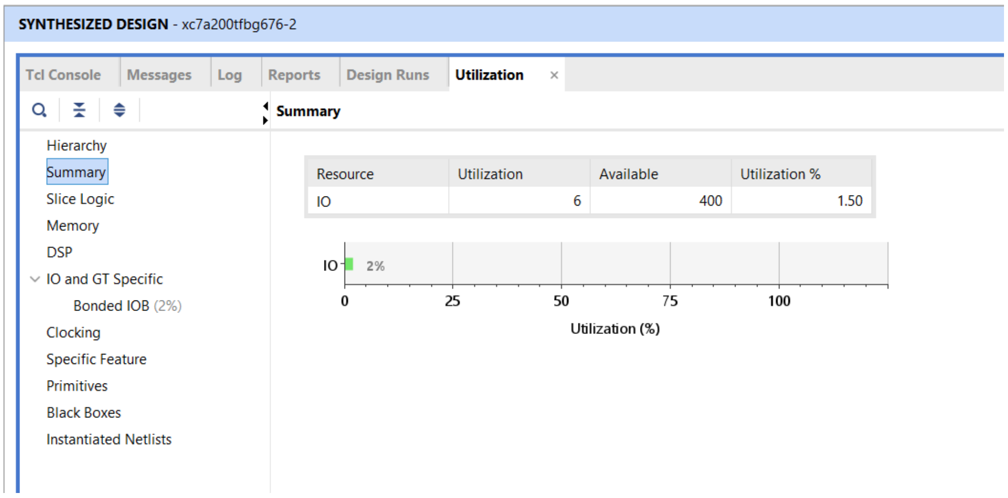
Name	Value
wr_clk	1
rd_clk	1
reset	0
wr_en	0
rd_en	0
Fifo[7:0][3:0]	X,X,X,X,X,X,X,X
wr_data[3:0]	8
rd_data[3:0]	8
rd_ptr[3:0]	8
rd_ptr_gr0[3:0]	1100
rd_ptr_gr1[3:0]	0100
rd_ptr_gr2[3:0]	0101
rd_ptr_sync[3:0]	0111
rd_ptr_wr[3:0]	5
Full	0
wr_ptr[3:0]	8
wr_ptr_gr0[3:0]	1100
wr_ptr_gr1[3:0]	1100
wr_ptr_gr2[3:0]	1100
wr_ptr_sync[3:0]	1100
wr_ptr_rd[3:0]	8
Empty	1
Depth[31:0]	8
Width[31:0]	4

NOTE:

The data is getting stored in the FIFO register as shown:
It is “RED” because we have *replaced the read address by ‘x’ intentionally for visual understanding* and are not errors, as already mentioned.
For Successful synthesis, we have commented out “ Fifo[rd_ptr]<={Width{1'dx}};” in the code.



RTL Schematic



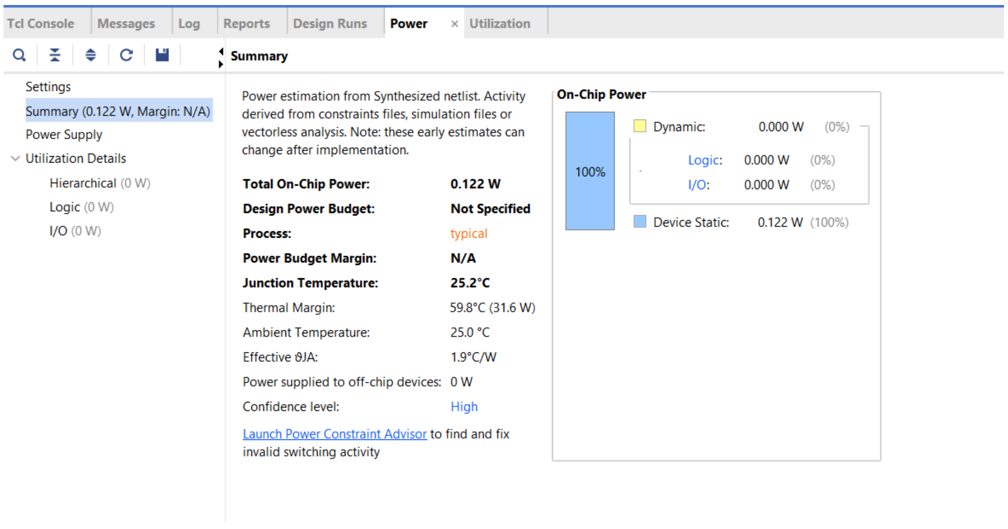
SYNTHESIZED DESIGN - xc7a200tfbg676-2

Tcl Console | Messages | Log | Reports | Design Runs | **Utilization**

Summary

- Hierarchy
- Summary
- Slice Logic
- Memory
- DSP
- IO and GT Specific
  - Bonded IOB (2%)
- Clocking
- Specific Feature
- Primitives
- Black Boxes
- Instantiated Netlists

Resource	Utilization	Available	Utilization %
IO	6	400	1.50



Tcl Console | Messages | Log | Reports | Design Runs | **Power** | Utilization

Summary

- Settings
- Summary (0.122 W, Margin: N/A)
- Power Supply
- Utilization Details
  - Hierarchical (0 W)
  - Logic (0 W)
  - I/O (0 W)

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**0.122 W**
Design Power Budget:	**Not Specified**
Process:	typical
Power Budget Margin:	**N/A**
Junction Temperature:	**25.2°C**
Thermal Margin:	59.8°C (31.6 W)
Ambient Temperature:	25.0 °C
Effective ϑJA:	1.9°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	High

Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

Dynamic:	0.000 W	(0%)
Logic:	0.000 W	(0%)
I/O:	0.000 W	(0%)
Device Static:	0.122 W	(100%)

Hardware Utilization and Power