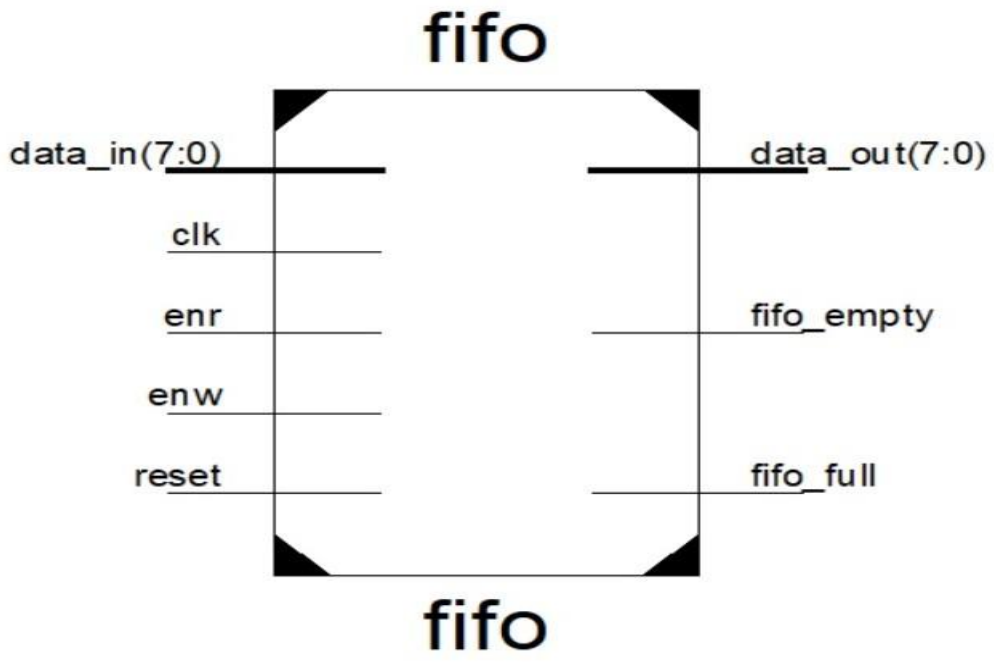
Class	:	
Batch	:	
ABC ID	:	
Roll. No	:	
Assignment No.	:	A.4
Assignment Name	:	FIFO (32-bit , organized as 4 X 8 , BYTE Addressable)
Date Of Performance	:	

BLOCK DIAGRAM



FUNCTION TABLE

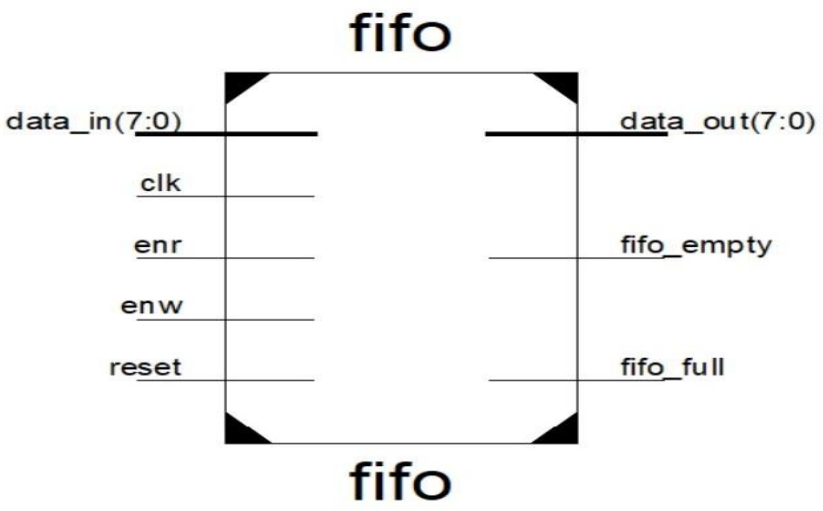
reset	clk	enw	enr	data_out	fifo_empty	fifo_full
1	x	x	x	$(00)_{16}$	1	0
0	↓	1	0	NA	0	0
0	↓	1	0	NA	0	0
0	↓	1	0	NA	0	0
0	↓	1	0	NA	0	1
0	↓	0	1	mem_0	0	0
0	↓	0	1	mem_1	0	0
0	↓	0	1	mem_2	0	0
0	↓	0	1	mem_3	0	0

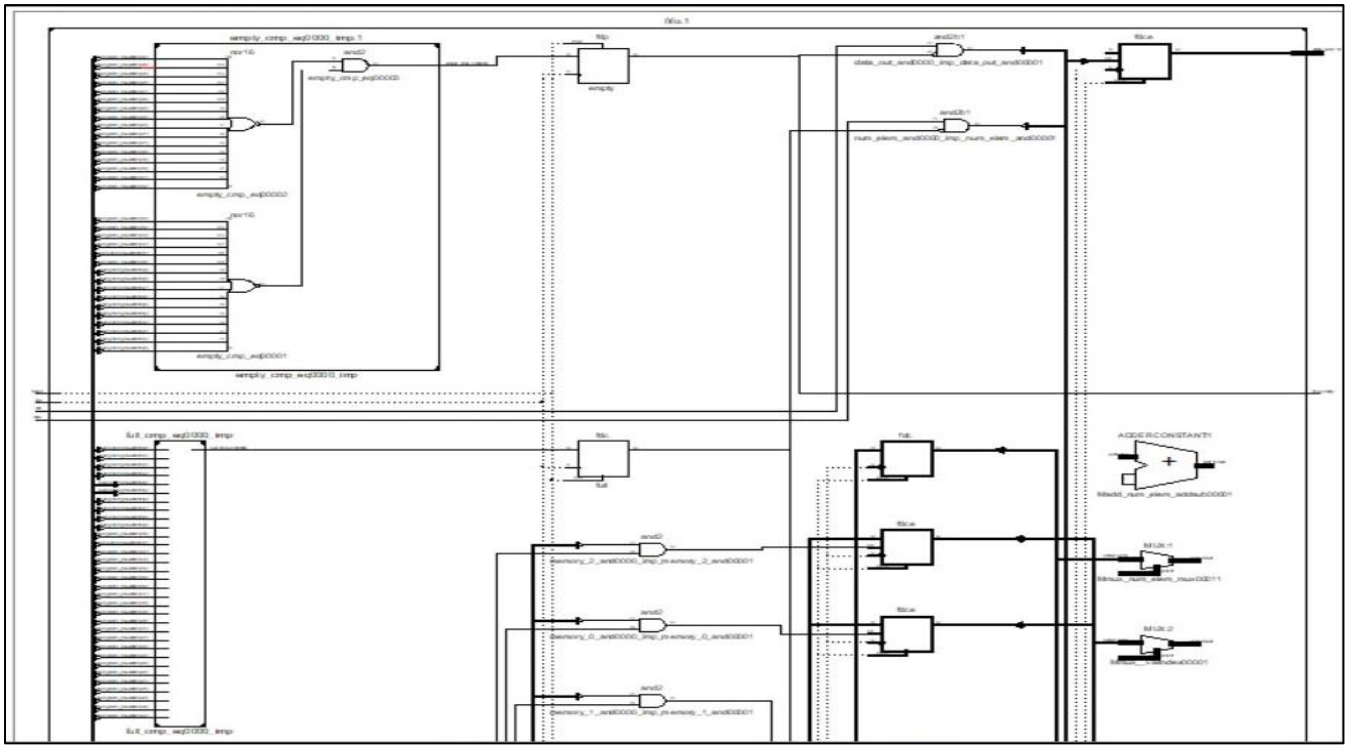
MAIN VHDL MODEL (MVM)

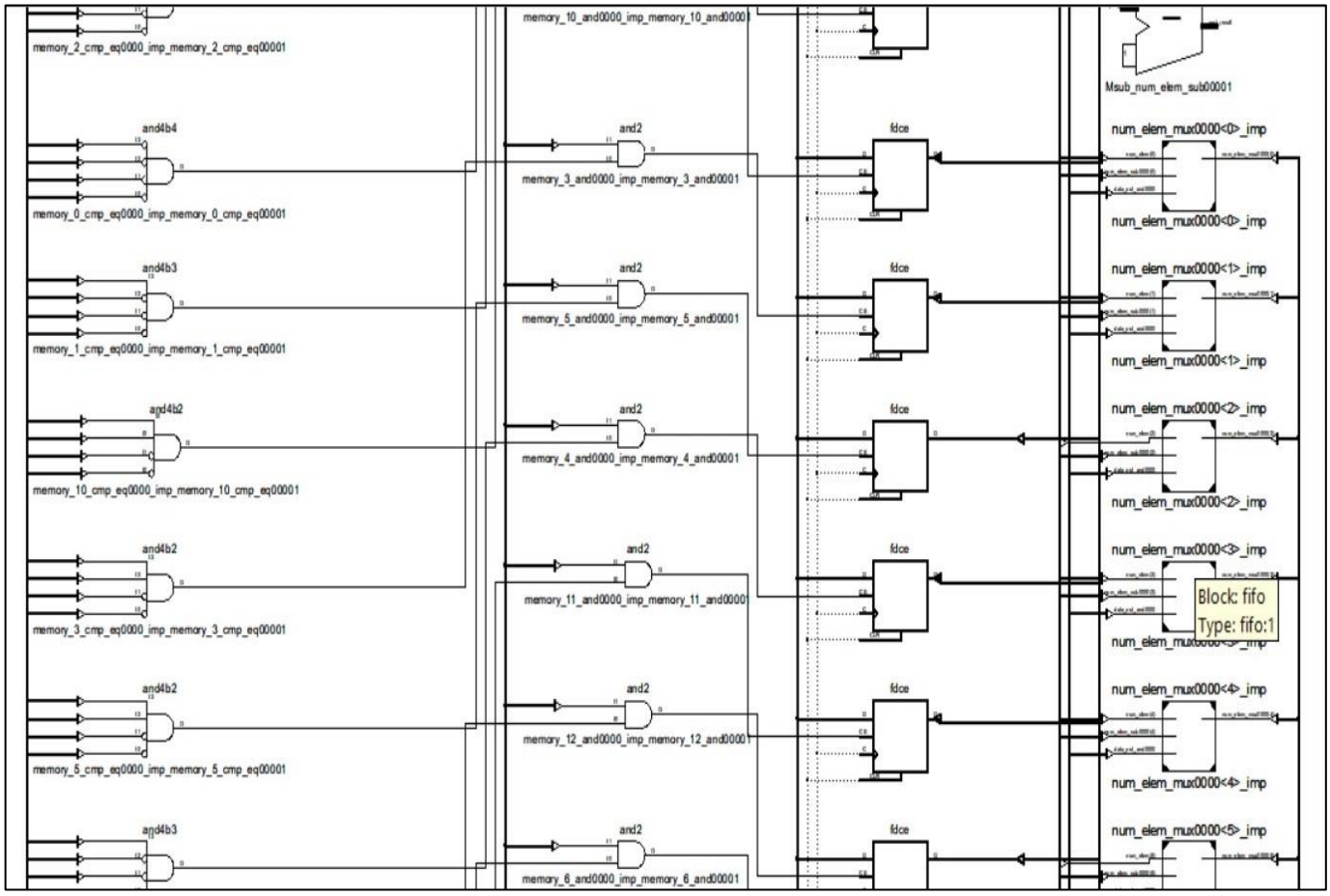
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity fifo is
generic (depth : integer := 16);  --depth of fifo
port (    clk : in std_logic;
          reset : in std_logic;
          enr : in std_logic;   --enable read,should be '0' when not in use.
          enw : in std_logic;    --enable write,should be '0' when not in use.
          data_in : in std_logic_vector (7 downto 0);     --input data
          data_out : out std_logic_vector(7 downto 0);    --output data
          fifo_empty : out std_logic;     --set as '1' when the queue is empty
          fifo_full : out std_logic     --set as '1' when the queue is full
         );
end fifo;

architecture fifo_arch of fifo is

type memory_type is array (0 to depth-1) of std_logic_vector(7 downto 0);
signal memory : memory_type :=(others => (others => '0'));   --memory for queue.
signal readptr,writeptr : integer := 0;  --read and write pointers.
signal empty,full : std_logic := '0';

begin

fifo_empty <= empty;
fifo_full <= full;

process(Clk,reset)
--this is the number of elements stored in fifo at a time.
--this variable is used to decide whether the fifo is empty or full.
variable num_elem : integer := 0;
begin
if(reset = '1') then

--    for i in 0 to depth-1 loop
--
--              memory(i)<=(others=>'0');
--            end loop;
 memory <= (others => (others=> '0'));

    data_out <= (others => '0');
             empty <= '1';
    full <= '0';
    readptr <= 0;
    writeptr <= 0;
    num_elem := 0;
elsif(rising_edge(Clk)) then
    if(enr = '1' and empty = '0') then  --read
```

```
      data_out <= memory(readptr);
      readptr <= readptr + 1;
      num_elem := num_elem-1;
    end if;
    if(enw ='1' and full = '0') then    --write
      memory(writeptr) <= data_in;
      writeptr <= writeptr + 1;
      num_elem := num_elem+1;
    end if;
    --rolling over of the indices.
    if(readptr = depth-1) then      --resetting read pointer.
      readptr <= 0;
    end if;
    if(writeptr = depth-1) then       --resetting write pointer.
      writeptr <= 0;
    end if;
    --setting empty and full flags.
    if(num_elem = 0) then
      empty <= '1';
    else
      empty <= '0';
    end if;
    if(num_elem = depth) then
      full <= '1';
    else
      full <= '0';
    end if;
  end if;
  end process;

  end fifo_arch;
```

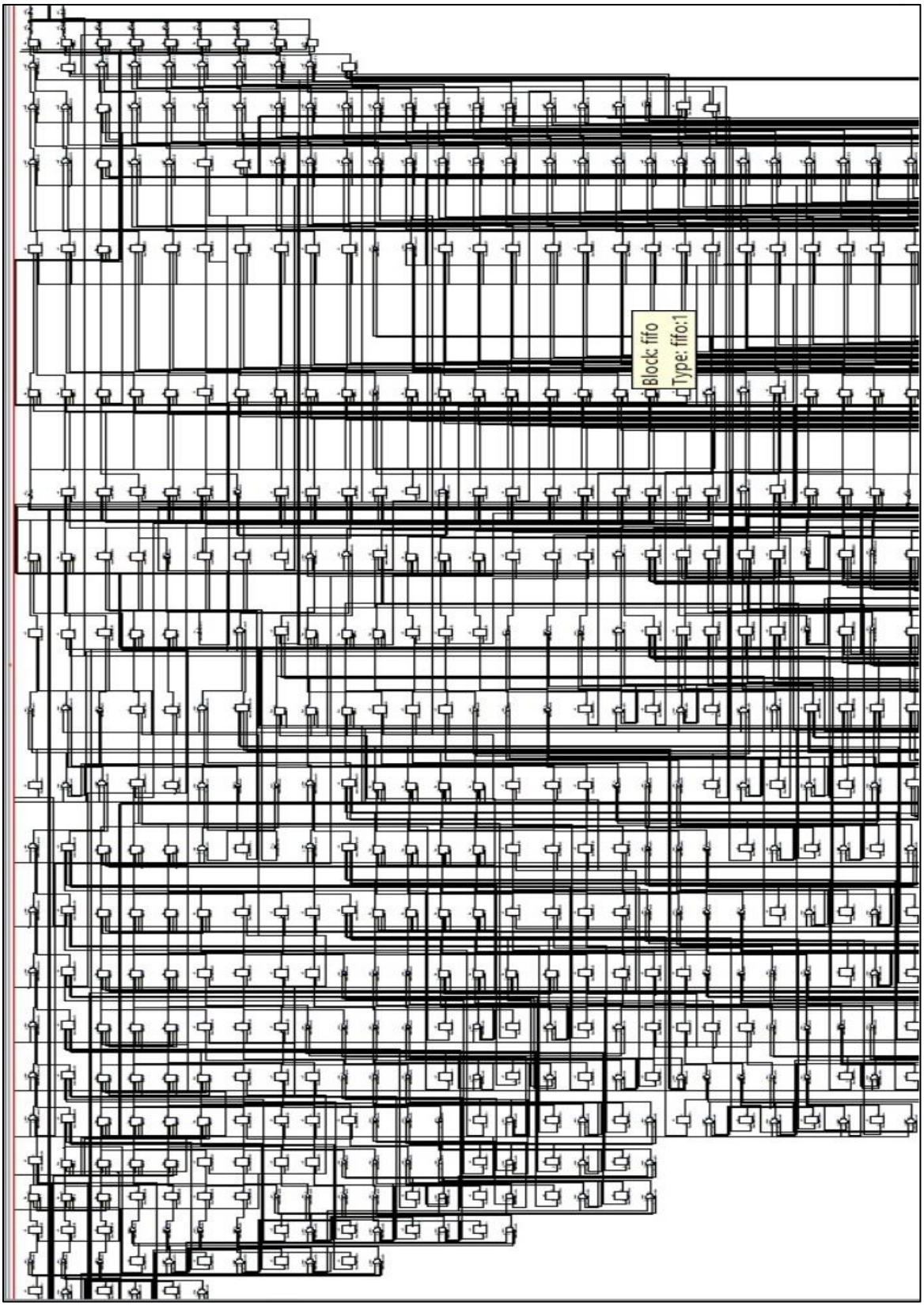
RTL SCHEMATIC:






memory_2_cmp_eq0000_imp_memory_2_cmp_eq00001
memory_10_and0000_imp_memory_10_and0000
Msub_num_elem_sub00001
and4b4
memory_0_cmp_eq0000_imp_memory_0_cmp_eq00001
and2
memory_3_and0000_imp_memory_3_and00001
fdce
num_elem_mux0000<0>_imp
and4b3
memory_1_cmp_eq0000_imp_memory_1_cmp_eq00001
and2
memory_5_and0000_imp_memory_5_and00001
fdce
num_elem_mux0000<1>_imp
and4b2
memory_10_cmp_eq0000_imp_memory_10_cmp_eq00001
and2
memory_4_and0000_imp_memory_4_and00001
fdce
num_elem_mux0000<2>_imp
and4b2
memory_3_cmp_eq0000_imp_memory_3_cmp_eq00001
and2
memory_11_and0000_imp_memory_11_and0000
fdce
num_elem_mux0000<3>_imp
Block: fifo
Type: fifo:1
and4b2
memory_5_cmp_eq0000_imp_memory_5_cmp_eq00001
and2
memory_12_and0000_imp_memory_12_and0000
fdce
num_elem_mux0000<4>_imp
and4b3
and2
memory_6_and0000_imp_memory_6_and00001
fdce
num_elem_mux0000<5>_imp

TECHNOLOGY SCHEMATIC



SYNTHESIS REPORT

a) Device Utilization Summary:

```
=========================================================================
*                      Final Report                     *
=========================================================================
Final Results
RTL Top Level Output File Name     : FIFO_4x8.ngr
Top Level Output File Name         : FIFO_4x8
Output Format                      : NGC
Optimization Goal                  : Speed
Keep Hierarchy                     : No

Design Statistics
# IOs                              : 23

Cell Usage :
# BELS                             : 32
#      GND                         : 1
#      INV                         : 2
#      LUT3                        : 21
#      MUXF5                       : 8
# FlipFlops/Latches                : 42
#      FDC_1                       : 1
#      FDCE_1                      : 40
#      FDP_1                       : 1
# Clock Buffers                    : 1
#      BUFGP                       : 1
# IO Buffers                       : 22
#      IBUF                        : 12
#      OBUF                        : 10
=========================================================================

Device utilization summary:
---------------------------

Selected Device : 3s250epq208-5

 Number of Slices:                 26  out of   2448     1%
 Number of Slice Flip Flops:       40  out of   4896     0%
 Number of 4 input LUTs:           23  out of   4896     0%
 Number of IOs:                    23
 Number of bonded IOBs:            23  out of    158    14%
   IOB Flip Flops:                 2
 Number of GCLKs:                   1  out of     24     4%
```

b) TIMING REPORT:

NOTE: THESE TIMING NUMBERS ARE ONLY A SYNTHESIS ESTIMATE.
FOR ACCURATE TIMING INFORMATION PLEASE REFER TO THE TRACE REPORT

GENERATED AFTER PLACE-and-ROUTE.

Clock Information:

Clock Signal	Clock buffer(FF name)	Load
clk	BUFGP	42

Asynchronous Control Signals Information:

Control Signal	Buffer(FF name)	Load
rst_inv(rst_inv1_INV_0:O)	NONE(d_out_0)	42

Timing Summary:

Speed Grade: -5

Minimum period: 2.098ns (Maximum Frequency: 476.644MHz)
Minimum input arrival time before clock: 3.955ns
Maximum output required time after clock: 4.040ns
Maximum combinational path delay: No path found

Timing Detail:

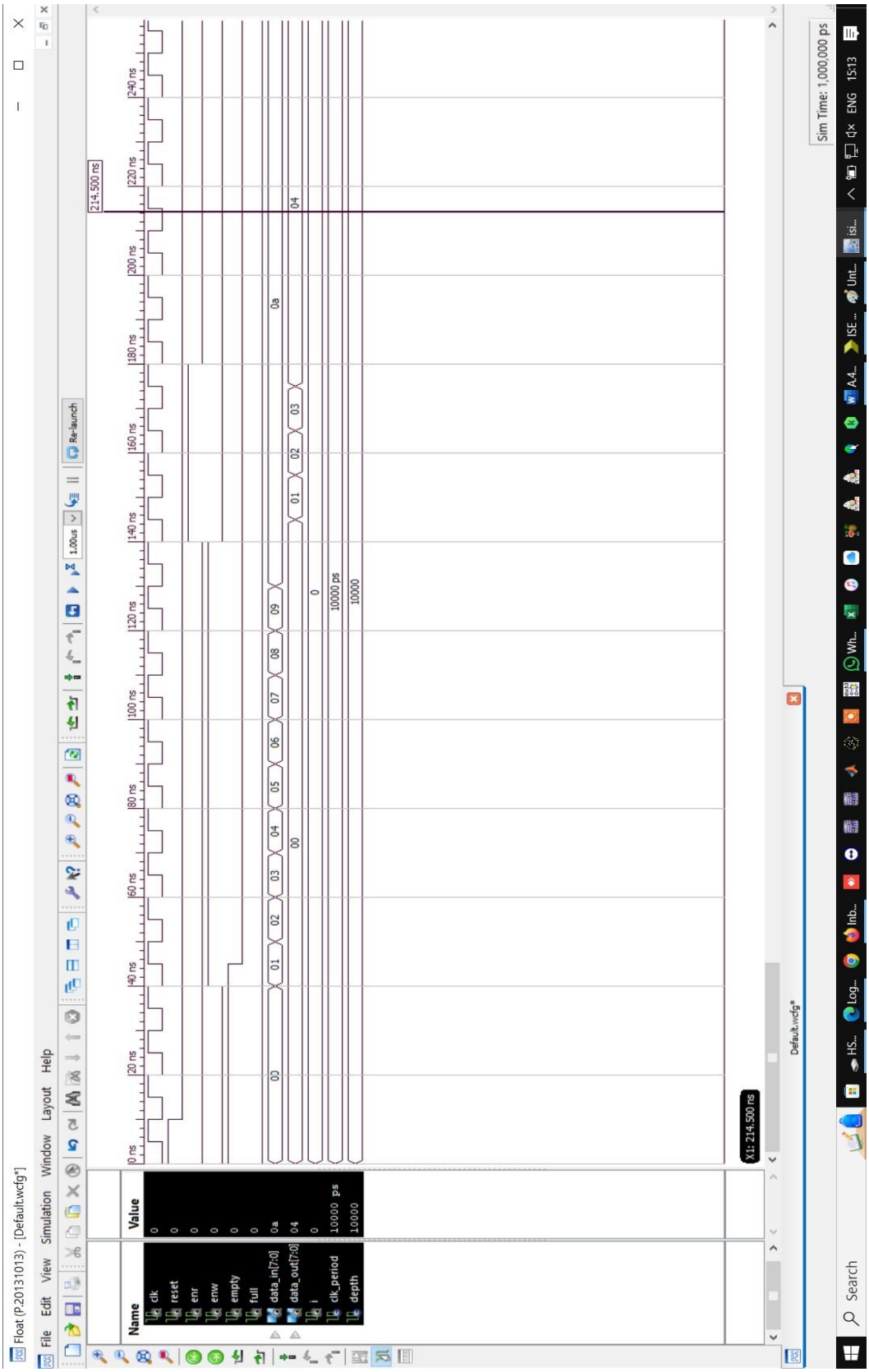
All values displayed in nanoseconds (ns)

TESTBENCH VHDL MODEL (TVM)

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_arith.ALL;

ENTITY fifo_tb IS
END fifo_tb;

ARCHITECTURE behavior OF fifo_tb IS
  --Inputs and outputs
  signal Clk,reset,enr,enw,empty,full : std_logic := '0';
  signal data_in,data_out : std_logic_vector(7 downto 0) := (others => '0');
   --temporary signals
   signal i : integer := 0;
  -- Clock period definitions
  constant Clk_period : time := 10 ns;
   constant depth : integer := 16;  --specify depth of fifo here.

BEGIN

   -- Instantiate the Unit Under Test (UUT)
  uut: entity work.fifo generic map(depth => depth) PORT MAP
(clk,reset,enr,enw,data_in,data_out,empty,full);

  -- Clock process definitions
  Clk_process :process
  begin
      Clk <= '0';
      wait for Clk_period/2;
      Clk <= '1';
      wait for Clk_period/2;
  end process;

  -- Stimulus process
  stim_proc: process
  begin
      reset <= '1';  --apply reset for one clock cycle.
      wait for clk_period;
      reset <= '0';
      wait for clk_period*3;  --wait for 3 clock periods(simply)
      enw <= '1';     enr <= '0';        --write 10 values to fifo.
    for i in 1 to 10 loop
         Data_In <= conv_std_logic_vector(i,8);
         wait for clk_period;
    end loop;
      enw <= '0';     enr <= '1';        --read 4 values from fifo.
    wait for clk_period*4;
      enw <= '0';     enr <= '0';
      wait for clk_period*10;  --wait for some clock cycles.
```

```
    enw <= '1';    enr <= '0';       --write 10 values to fifo.
   for i in 11 to 20 loop
       Data_In <= conv_std_logic_vector(i,8);
       wait for clk_period;
   end loop;
    enw <= '0';    enr <= '0';
    wait for clk_period*10;  --wait for some clock cycles.
    enw <= '0';    enr <= '1';       --read 4 values from fifo.
   wait for clk_period*4;
    enw <= '0';    enr <= '0';
    wait for clk_period;
    enw <= '0';    enr <= '1';       --read 4 values from fifo.
   wait for clk_period*8;
    enw <= '0';    enr <= '0';
    wait for clk_period;
    enw <= '0';    enr <= '1';       --read 8 values from fifo.
   wait for clk_period*4;
    enw <= '0';    enr <= '0';
    wait for clk_period;
    enw <= '0';    enr <= '1';       --read 4 values from fifo.
   wait for clk_period*4;
    enw <= '0';    enr <= '0';
    wait;
  end process;

END;
```

ISIM WAVEFORMS



PIN-LOCKING REPORT

```
# PlanAhead Generated physical constraints

NET "data_in[7]" LOC = P165;    #sw4-0
NET "data_in[6]" LOC = P167;    #sw4-1
NET "data_in[5]" LOC = P163;    #sw4-2
NET "data_in[4]" LOC = P164;
NET "data_in[3]" LOC = P161;
NET "data_in[2]" LOC = P162;
NET "data_in[1]" LOC = P160;
NET "data_in[0]" LOC = P153;    #sw4-7
NET "data_out[7]" LOC = P179;  #sw3-0
NET "data_out[6]" LOC = P180;  #sw3-1
NET "data_out[5]" LOC = P177;
NET "data_out[4]" LOC = P178;
NET "data_out[3]" LOC = P152;
NET "data_out[2]" LOC = P168;
NET "data_out[1]" LOC = P171;
NET "data_out[0]" LOC = P172;  #sw3-7
NET "clk" LOC = P132;
NET "reset" LOC = P204;         #k0
NET "enr" LOC = P184;           #sw2-6
NET "enw" LOC = P194;           #sw2-7
NET "fifo_empty" LOC = P199;#sw1-6
NET "fifo_full" LOC = P196;     #sw1-7
```

CONCLUSION

Thus, we have:

1) Modeled a 4x8 FIFO using Behavioral Modeling Style.
2) Observed following Schematics: **RTL** & **Technology Schematics** generated **Post-Synthesis**.
3) Interpreted **Device Utilization Summary** in terms of LUTs, SLICES, IOBs, Multiplexers &D FFs used out of the available device resources.
4) Interpreted the TIMING Report in terms of Maximum combinational delay as indicative ofthe Maximum Operating Frequency.
5) Written a TESTBENCH to verify the functionality of 4x8 FIFO & verified the functionality asper the FUNCTION-TABLE, by observing ISIM Waveforms.
6) Used PlanAhead Editor for pin-locking.
7) Prototyped the FPGA **XC3S250EPQ208-5** to realize 4x8 FIFO & verified its operation by giving suitable input combinations.