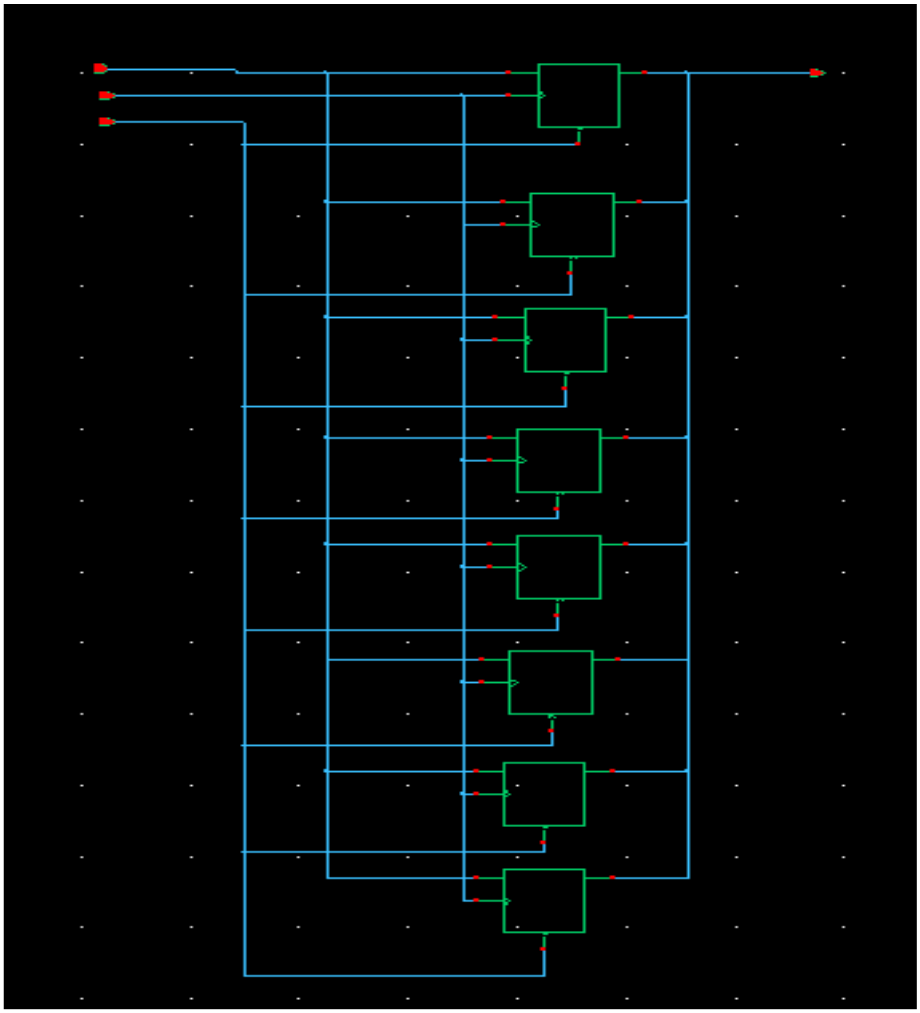
A brief account of any design choices you made and reasons why.

1. Data Buffer: The required buffer is positive edge with active low reset. Hence a D -flip-flop with D input, Q output, edge clock in and low reset is been chose from the library. i.e. DFFSRLQ (Positive-edge triggered D-flipflop with synchronous reset (active low) and Q-output only)

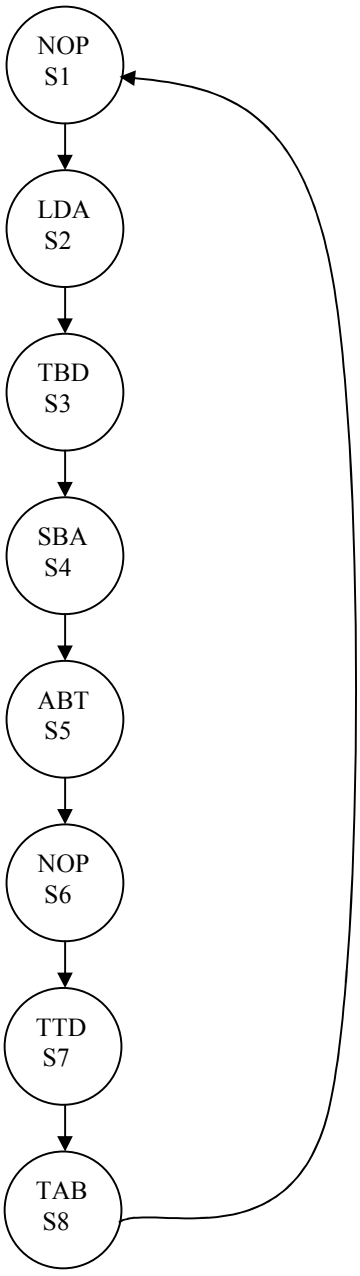


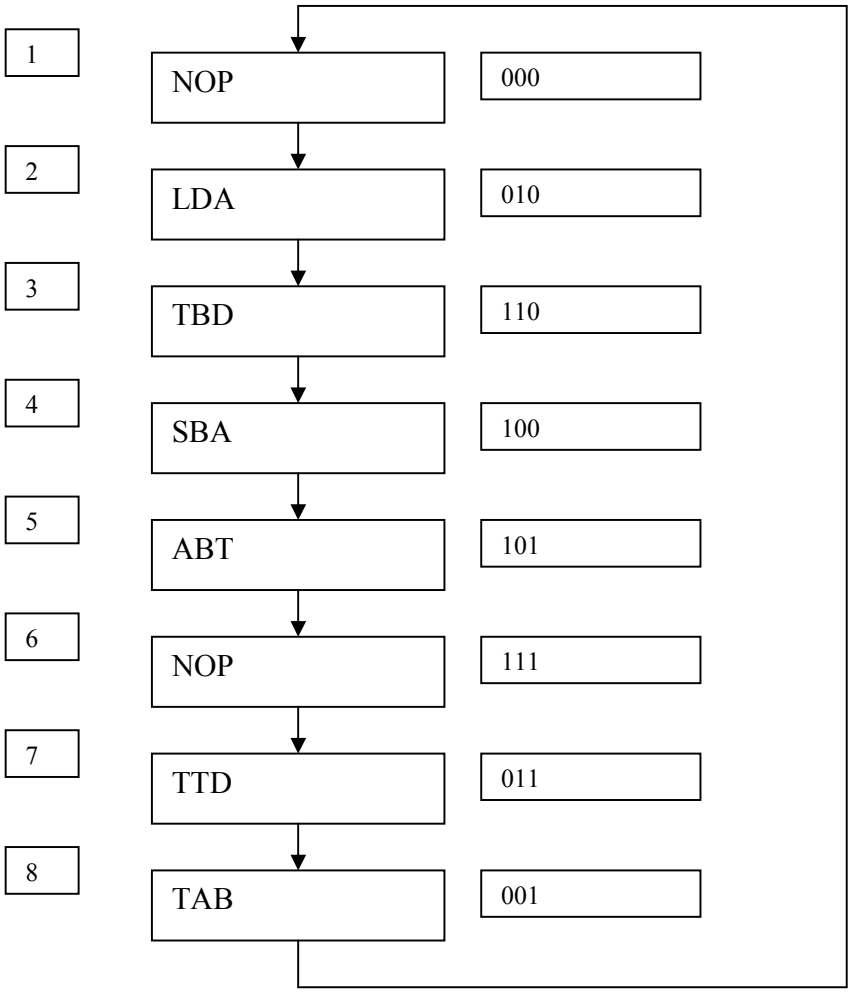
2. Control Unit: A moore machine with clocked output is implemented for control unit of the LDI86. It has Eight state with seven different signals.

3. Multiplexor: This unit is full custom cell. First, using PMOS and NMOS a 1-bit multiplexor is made. Then this cell is used for 8-bit as an 8-bit multiplexor.

An ASM chart for the control unit.

STATE	NOP	LDA	TBD	SBA	ABT	NOP	TTD	TAB
SIGNAL								
M(2:0)	110	100	011	000	001	110	101	010
ADDSHIFT(1:0)	01	01	01	10	01	01	01	01
AEN		1	0	0	0		0	0
BEN		0	0	0	0		0	1
TEN		0	0	1	1		0	0
DEN		0	1	0	0		1	0
DATALOAD	0	1	1	1	1	1	1	1






```
library IEEE; use IEEE.std_logic_1164.all;

entity C_UNIT is
port (CLOCK, RESET : in std_logic;
        AEN,BEN,TEN,DEN,DATALOAD :out std_logic;
        M: out std_logic_vector(2 downto 0);
        ADDSHIFT: out std_logic_vector(1 downto 0));
end C_UNIT;
architecture BEHAVIOUR of C_UNIT is

type STATE_TYPE is (ZERO,ONE,TWO,THREE,FOUR,FIVE,SIX,SEVEN,EIGHT);

signal STATE: STATE_TYPE;

begin
FSM : process (CLOCK, RESET)
begin
if RESET = '0' then STATE <= ZERO;
elsif CLOCK'event and CLOCK ='1' then
case STATE is
    when ZERO =>
```

```
STATE <= ONE;
            when ONE =>
STATE <= TWO;
            when TWO =>
STATE <= THREE;
            when THREE =>
STATE <= FOUR;
            when FOUR =>
STATE <= FIVE;
            when FIVE =>
STATE <= SIX;
            when SIX =>
STATE <= SEVEN;
            when SEVEN=>
STATE <= EIGHT;
            when EIGHT =>
STATE <= ONE;
end case;
end if;
end process FSM;

M<=         "110" when (STATE = ONE) else
            "100" when (STATE = TWO) else
            "011" when (STATE = THREE) else
            "001" when (STATE = FIVE) else
            "110" when (STATE = SIX) else
            "101" when (STATE = SEVEN) else
            "010" when (STATE = EIGHT) else
            "000";

ADDSHIFT <=     "10" when (STATE = FOUR) else
                "00" when (STATE = ZERO ) else
                "01";

AEN<=   '1' when (STATE = TWO) else
            '0';
BEN<=   '1' when (STATE = EIGHT) else
            '0';
TEN<=   '1' when (STATE = FOUR) else
            '1' when (STATE = FIVE) else
            '0';
DEN<=   '1' when (STATE = THREE) else
            '1' when (STATE = SEVEN) else
            '0';
DATALOAD<=      '0' when (STATE = ONE) else
                '1';
end BEHAVIOUR;
```

TESTBENCH FOR FSM CONTROL UNIT

```
library ieee;
Use ieee.std_logic_1164.ALL;

Entity tstbnch_cu is
end tstbnch_cu;

architecture test of tstbnch_cu is

component CTRL_UNIT
port (      reset,clock: in std_logic;
            aen,ben,ten,den,dataload: out std_logic;
            m: out std_logic_vector(2 downto 0);
```

```
            addshift: out std_logic_vector(1 downto 0)
            );
end component;

signal reset,clock: std_logic;
signal aen,ben,ten,den,dataload: std_logic;
signal m: std_logic_vector(2 downto 0);
signal addshift: std_logic_vector(1 downto 0);


begin
comp_ut: CTRL_UNIT
port map(reset,clock,aen,ben,ten,den,dataload,m,addshift);

    --generate a 352.8 kHz clock (8x44.1khz)
        clock_gen : process
        begin
        clock <= '0', '1' after 1400 us;
        wait for 2800 us;
        end process;

intialisation:process
begin
reset <= '0';
wait for 5600 us; reset <= '1';

    wait for 22400 us; wait;

end process; end test;
```

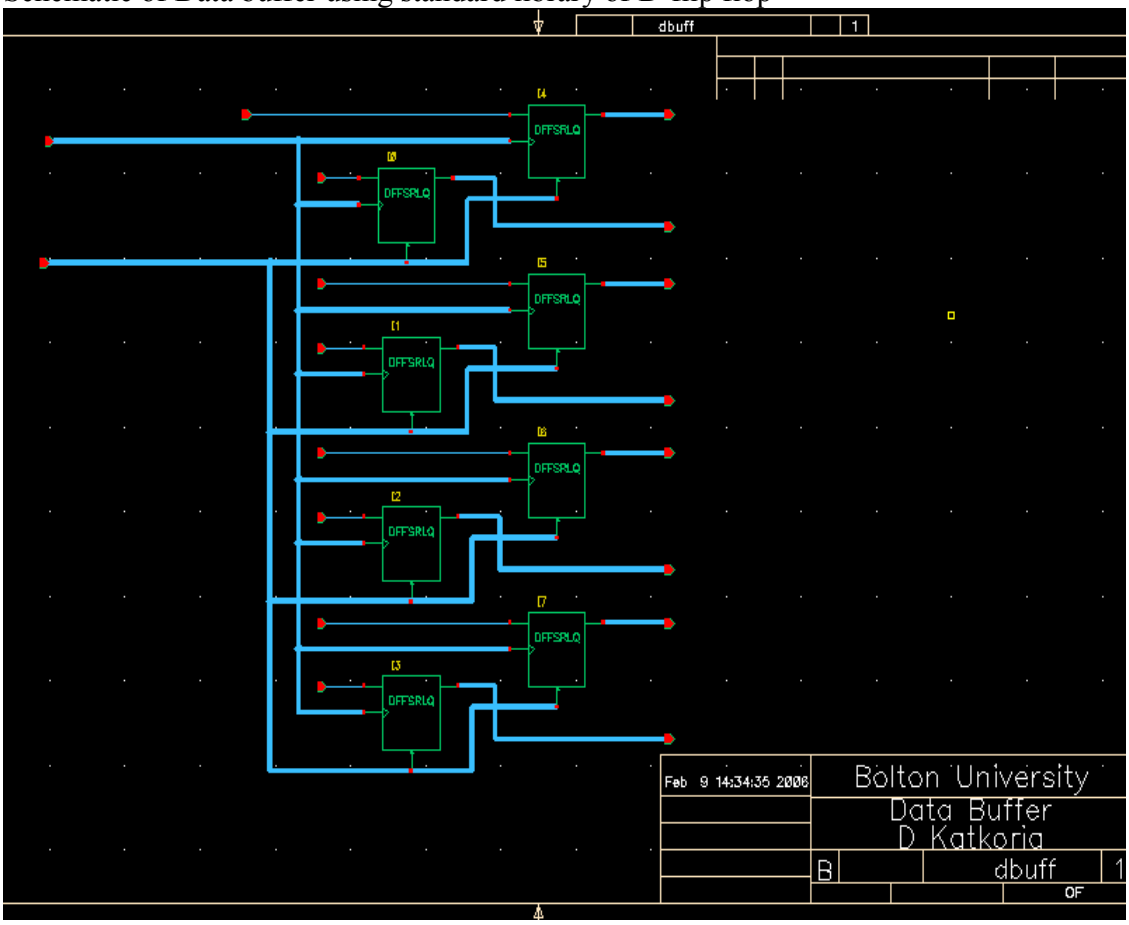
A brief analysis of the testability of the design and how this could be improved.

LDI86 is divided into following four major blocks.

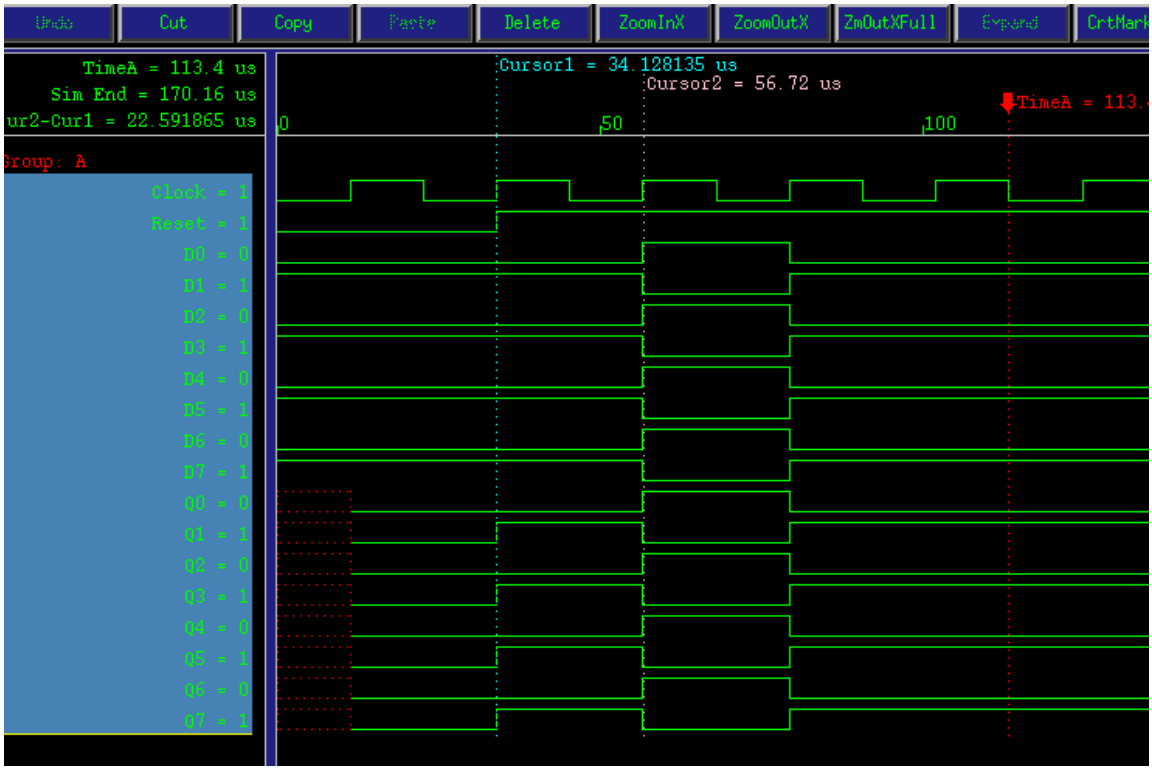
1. Data Buffer (made by using standard library)
2. Processor (IP core)
3. Control Unit (VHDL)
4. Multiplexor (full custom component)

1. Data Buffer

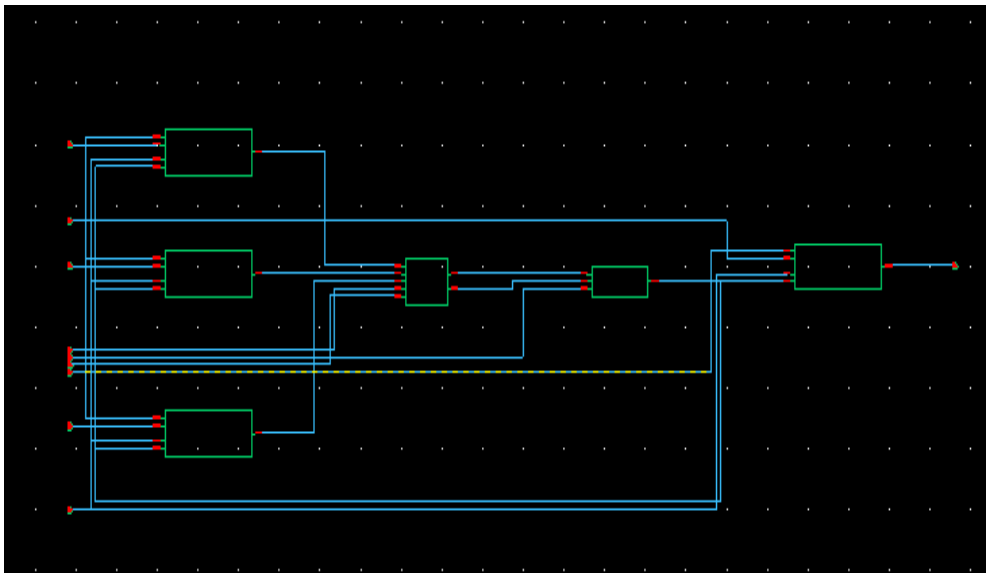
Schematic of Data buffer using standard library of D-flip flop



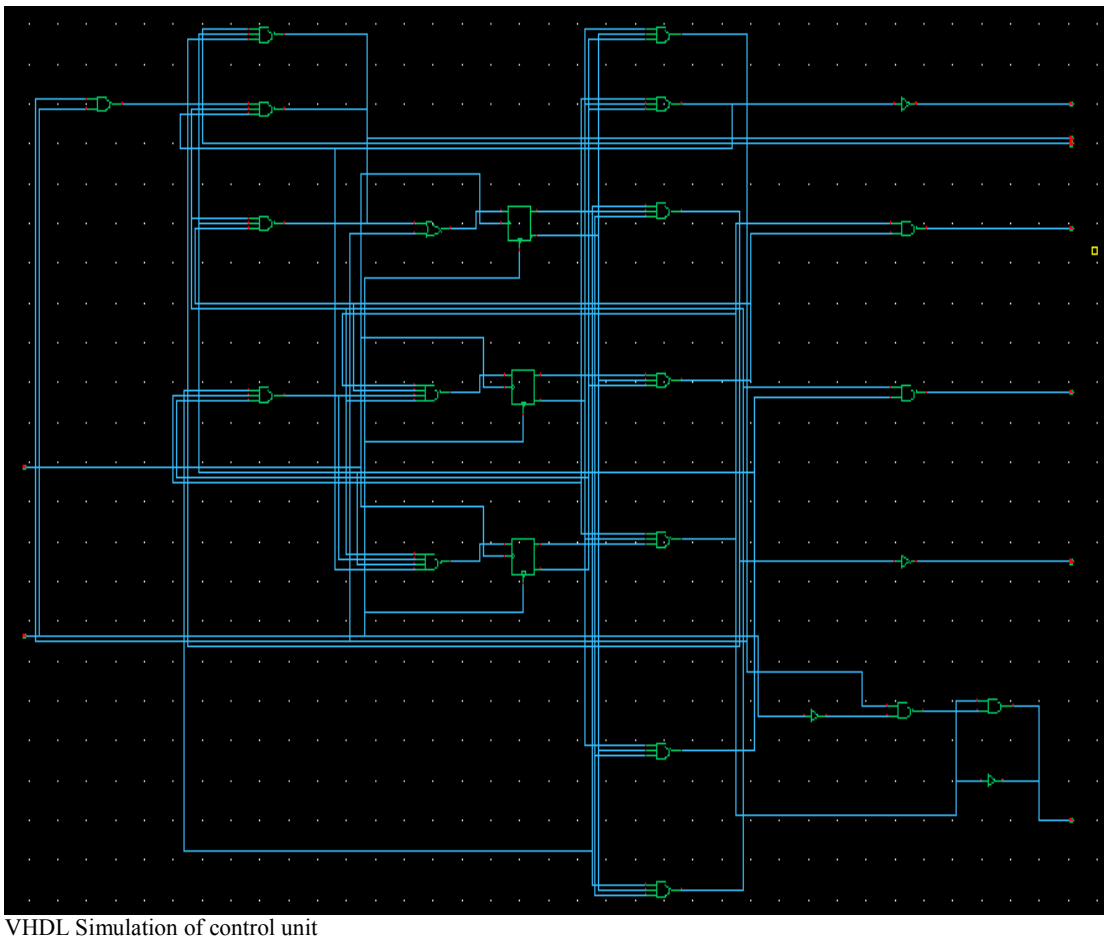
Simulation of Data buffer



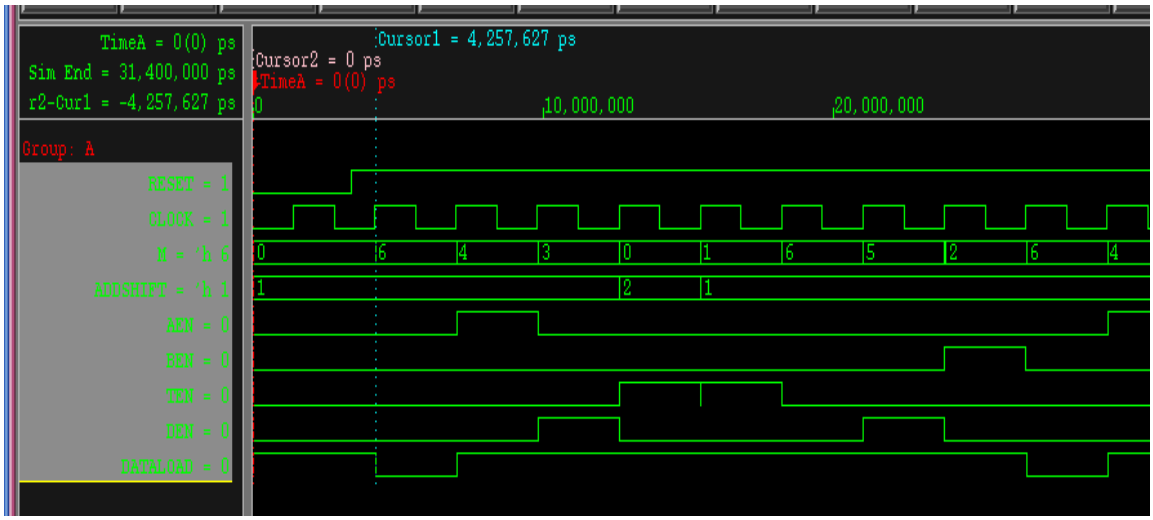
2. PROCESSOR
SYNTHESIS of IP core



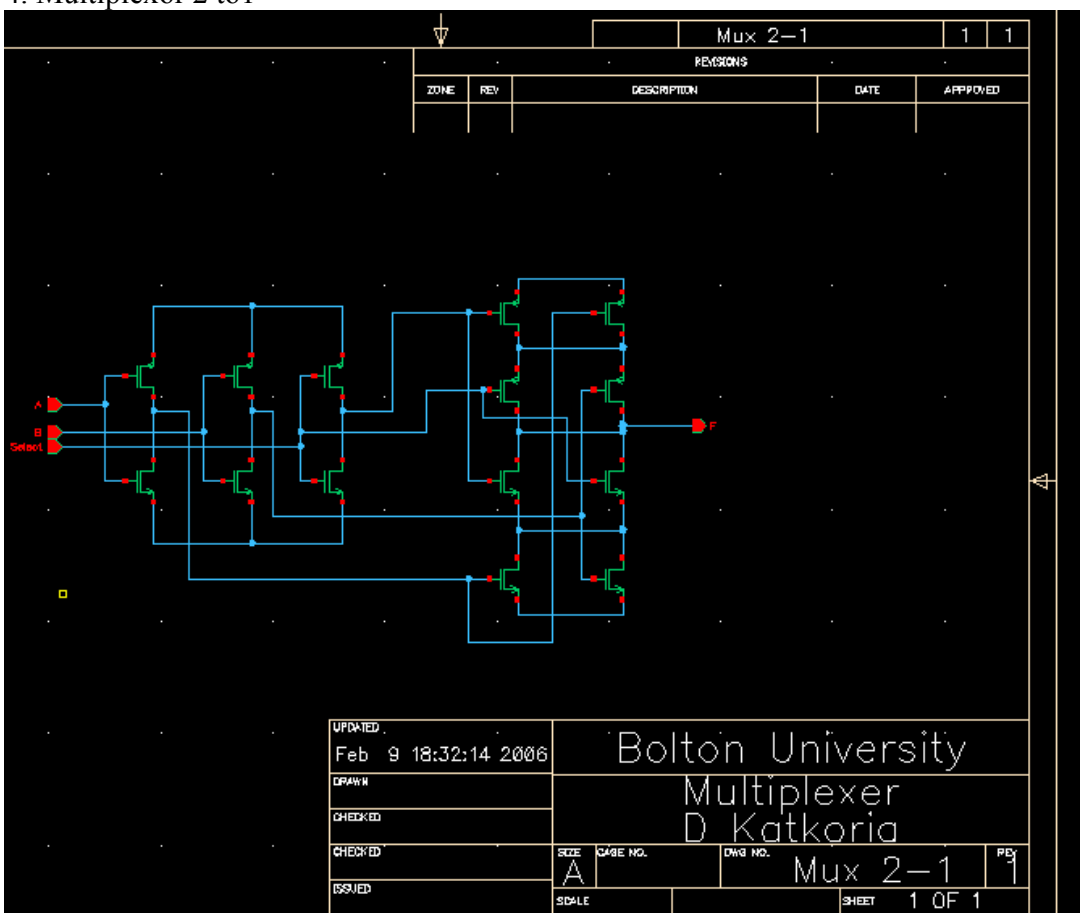
3.Control Unit
SYNTHESIS of VHDL code



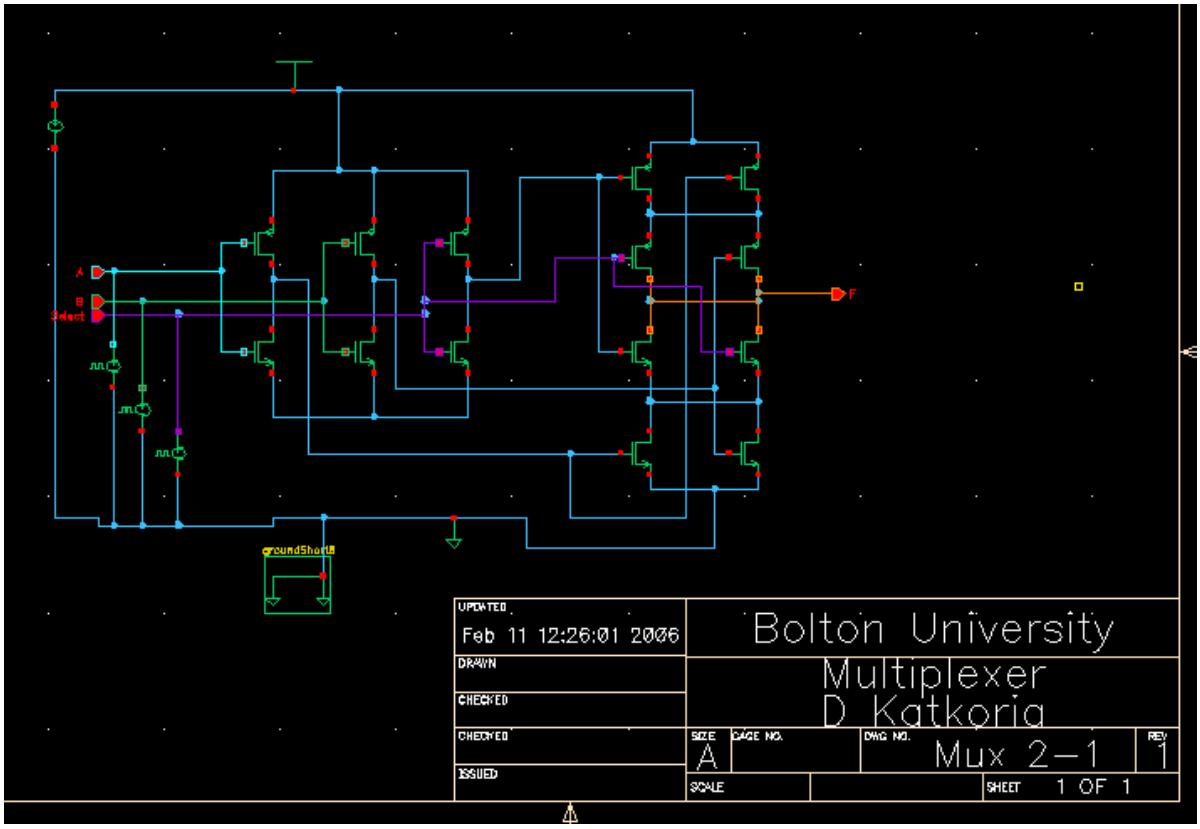
VHDL Simulation of control unit



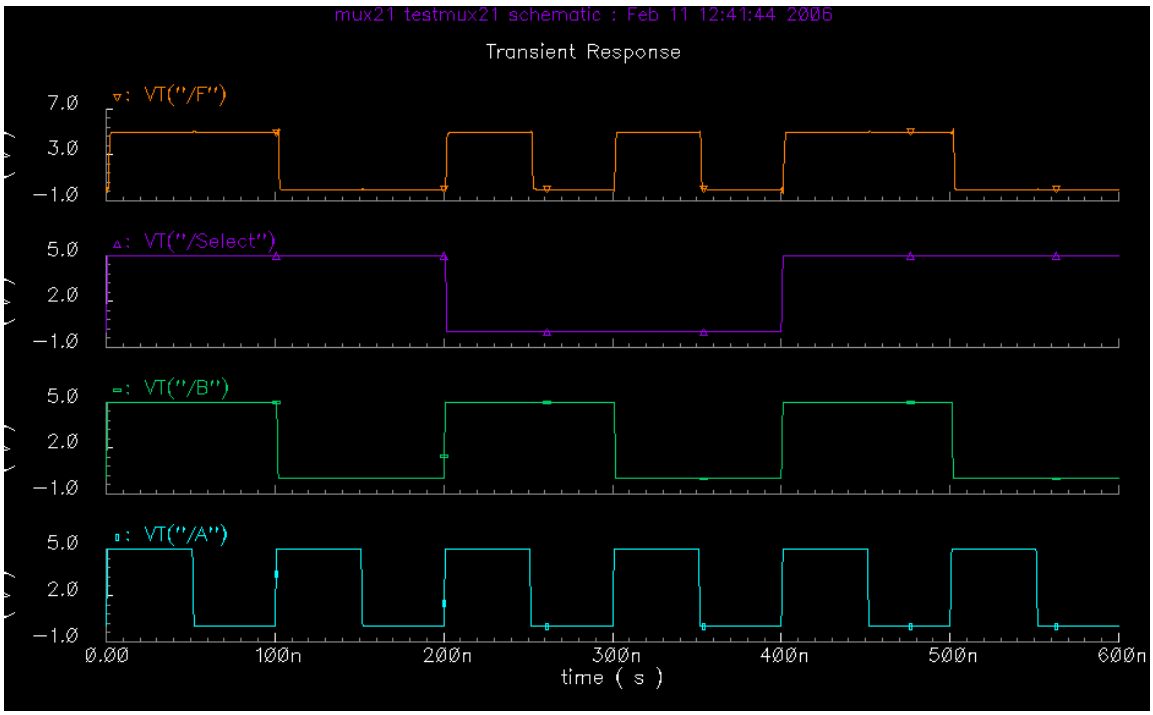
4. Multiplexor 2 to1



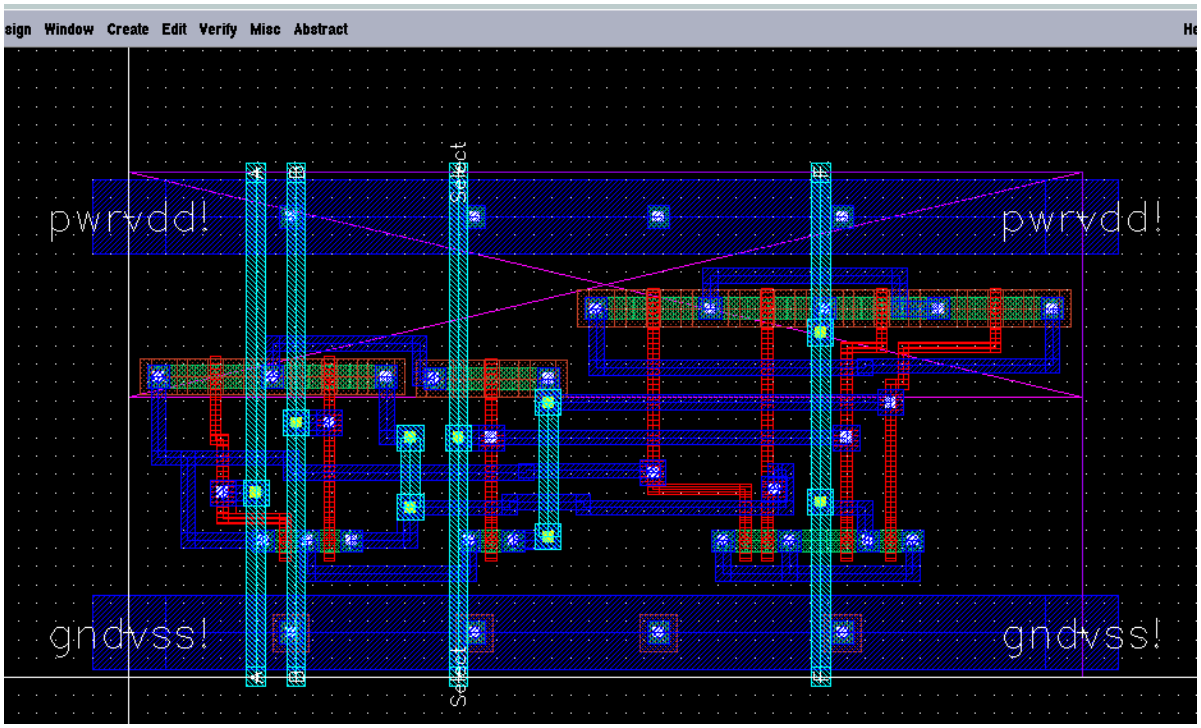
Test Schematic of 2 -1 full custom multiplexor cell:



Simulation waveform:



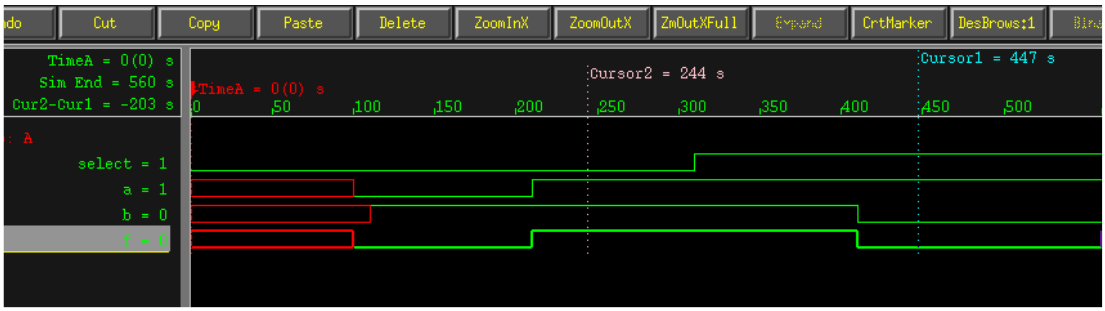
Layout and abstract of 2-1 full custom multiplexor cell:



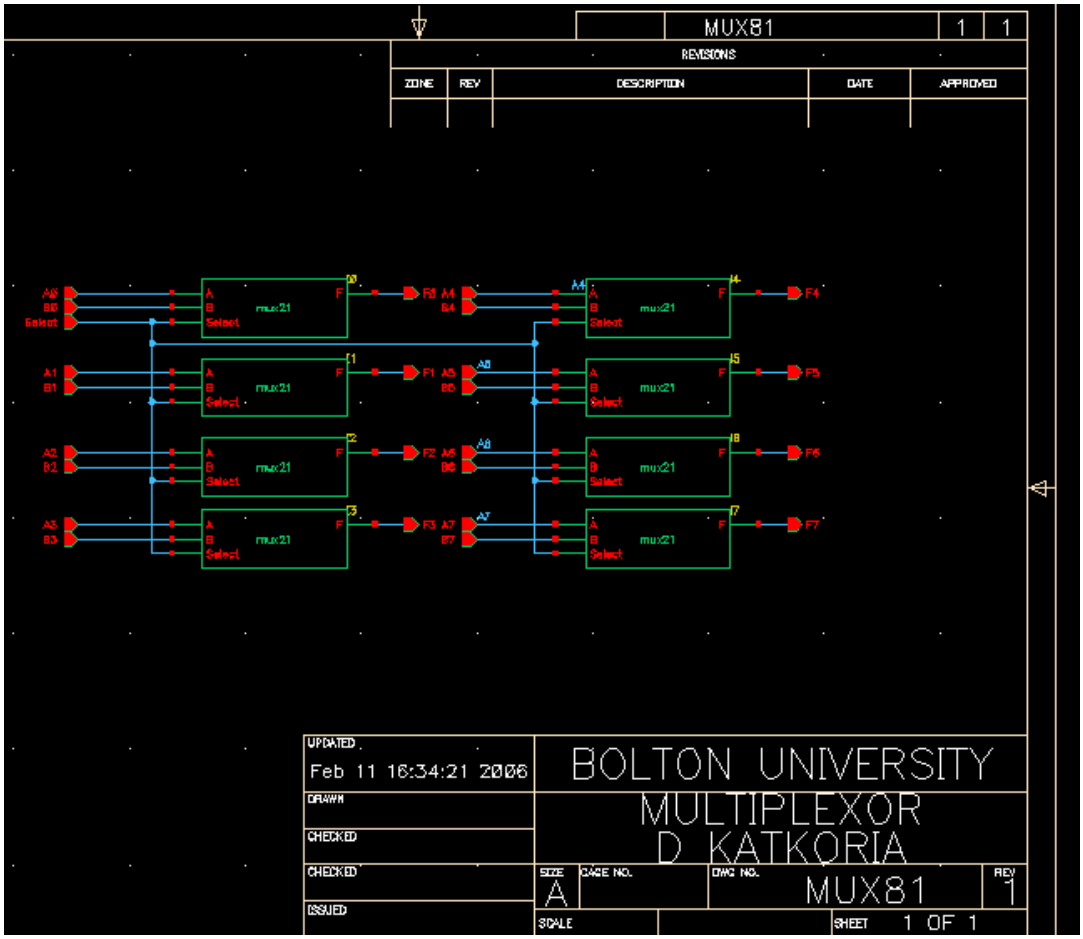
Verilog functional model:

```
module mux21 (F, A, B, Select);
    output F;
    input A;
    input B;
    input Select;
reg F;
always @(select or A or B)
F=Select ? B : A;
endmodule
```

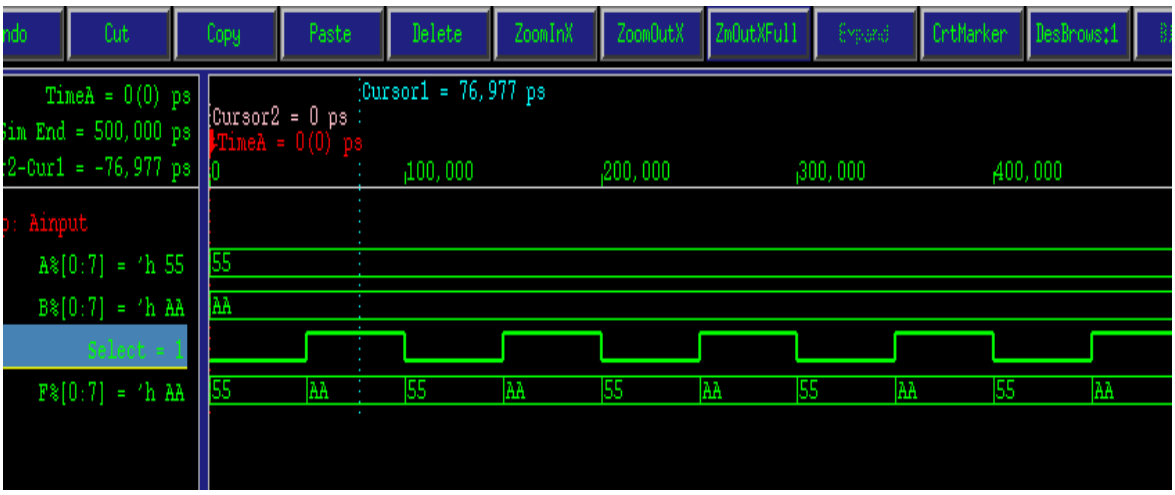
Verilog Simulation of 2-1 multiplexor



5. Multiplexor 8 to1 (combination of 2 -1 multiplexor to form 8-1 mux)
Schematic

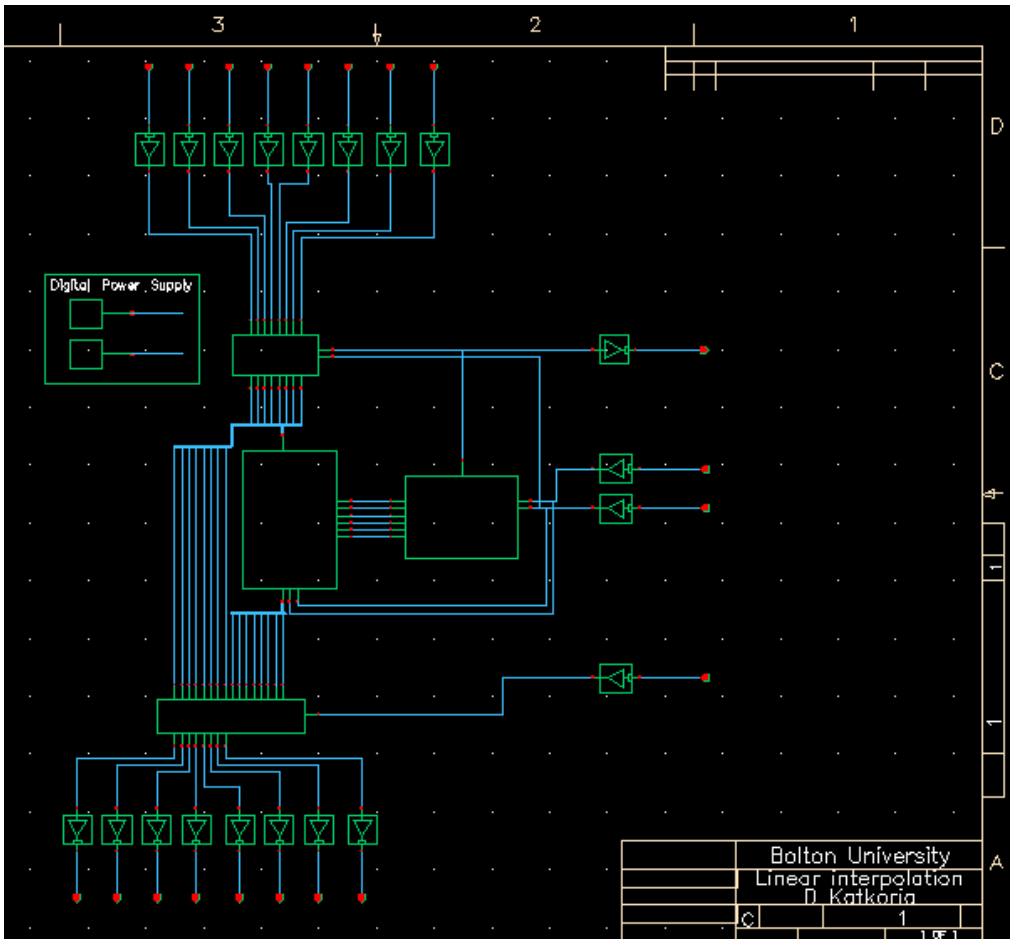


Verilog Simulation of 8-1 multiplexor

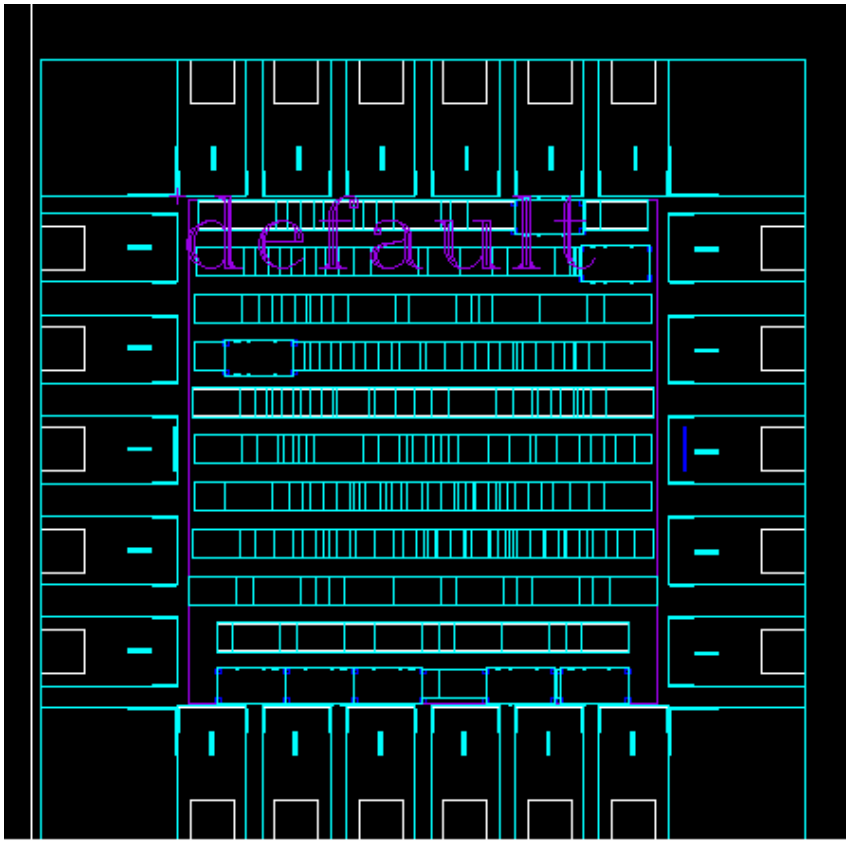


Chip:

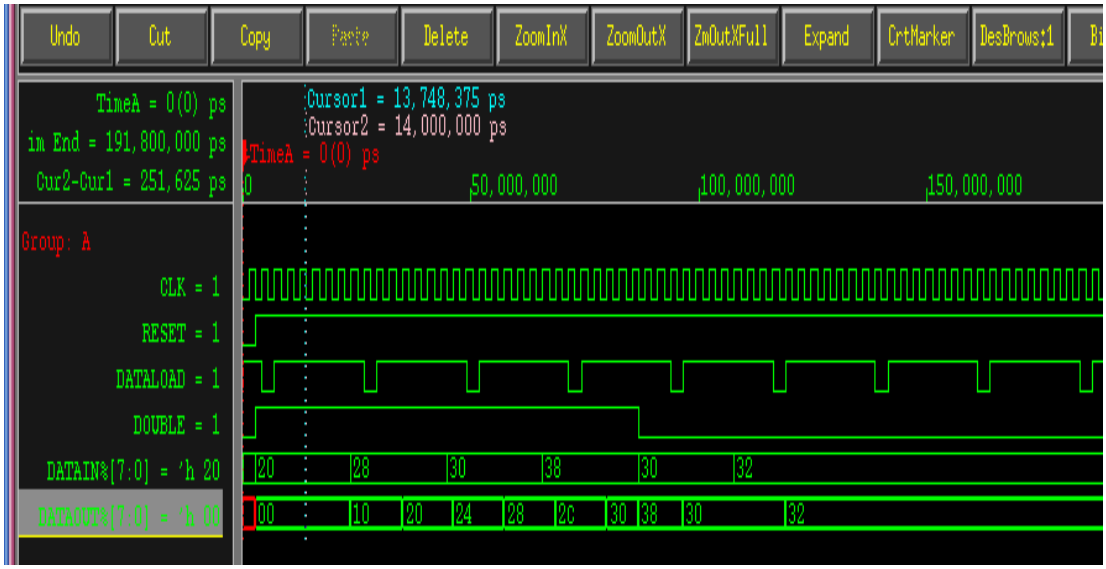
Synthesised Schematic of chip LDI86:



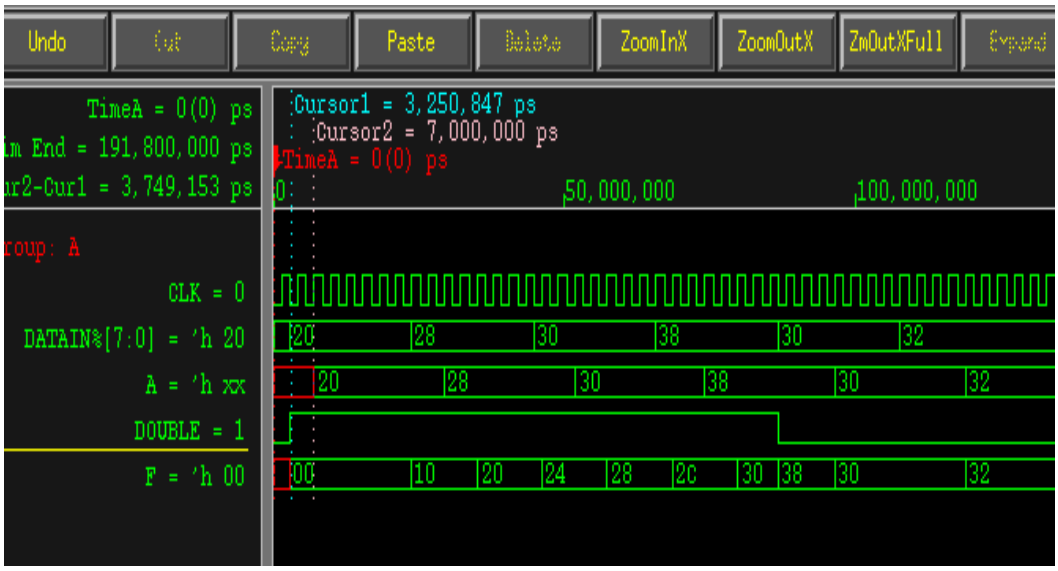
Placement/routing of chip LDI86:



Post layout simulation



DELAY OF DATA BUFFER= 3.7 us



Design file: /v1/students/dk1ect/AMI4407/stdcell
Library Name: linterpol
1. Cell: C_UNIT (for Control Unit)
2. Cell: mux81 (for Full Custom Multiplexor)
3. Cell: processor (for Processor)
4. Cell: dbuffer (for Data Buffer)
HDL File: /v1/students/dk1ect/AMI4407/stdcell/vhdl