LNMIIT
The LNM Institute of Information Technology

Digital System Design using FPGA

PROJECT REPORT

TOPIC: Low Power FIR Filter design using FPGA

Submitted By:
Tanmay Chaturvedi (Y12UC280)
Paras (Y12UC174)
Adil Siddiqui (Y12UC010)

Mentor:
Dr. Kusum Lata

Low Power FIR Filter design using VHDL

ABSTRACT

The motive of our project is to design and implement a low power FIR filter. We used VHDL (**V**ery high speed integrated circuit **H**ardware **D**escriptive **L**anguage) to support the development of filter on state-of-the art FPGA (**F**ield **P**rogrammable **G**ate **A**rray). The filter is implemented using two main components- D Flip Flop and adder. We carried comparison between 2-Taps Filter, 4-Taps Filter and 10-Taps Filter. The filter efficiency was improved by using higher order FIR filter but the power efficiency was reduced due to increased number of components. The lowest power we achieved was **55 mWatts** using 2-Tap FIR Filter based on adder and D Flip Flop in **25 MHZ** with 8 bit inputs. To our conclusion, we were able to decrease the power consumption to upto **33%**. We used **Xilinx Power Estimator** to calculate the power in Watts. We worked on Xilinx Family Spartan3E, device XC3S100E.

THEORY

Digital filters can be classified into 2 classes known as FIR (Finite Impulse response) and IIR (Infinite Impulse Response) filters. Advantage of FIR over IIR is that they are relatively stable. The output of an FIR filter Y(n) is given by the following equation:

$$Y(n) = \sum_{i=0}^{N-1} x(n-i)h_i$$

Where 'N' is the order of filter, x(n) is the input.

To find the optimum filter response, we used Window method. In this, we used Kaiser Window and extracted the filter coefficients for different values of N (order).

The MATLAB code for Kaiser window:

```
clc;
close all;
clear all;
d=fdesign.lowpass('n,fc',9,150,1000);
Hd= window(d,'window',@kaiser);
fvtool(Hd);
```

Filter coefficients from above window method.

N=2:

```
0.5
0.5
```

N=4:

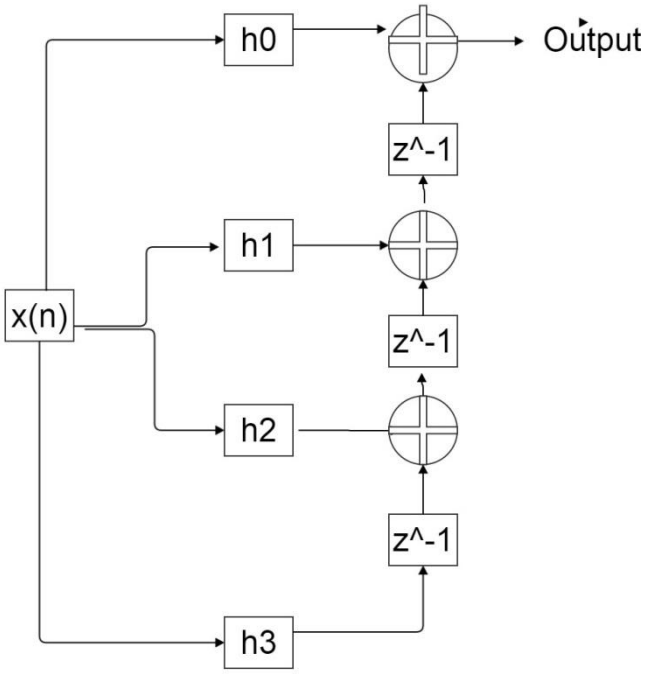
```
0.20353106055596351
0.29646893944403652
0.29646893944403652
0.20353106055596351
```

N=10:

```
-0.05783290413150511
-0.013379160772761783
 0.086222086210398979
 0.20315883358403275
 0.28183114510983509
 0.28183114510983509
 0.20315883358403275
 0.086222086210398979
-0.013379160772761783
-0.05783290413150511
```

IMPLEMENTATION

Implementation of 4-Tap FIR FILTER:



VHDL CODE

Code for D-Flip Flop Component

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

entity DFF is
  port(
    Q : out signed(15 downto 0);      --output connected to the adder
    Clk :in std_logic;      -- Clock input
    D :in  signed(15 downto 0)      -- Data input from the M block.
  );
end DFF;

architecture Behavioral of DFF is

signal qt : signed(15 downto 0) := (others => '0');

begin

Q <= qt;

process(Clk)
begin
 if ( rising_edge(Clk) ) then
   qt <= D;
 end if;
end process;
end Behavioral;
```

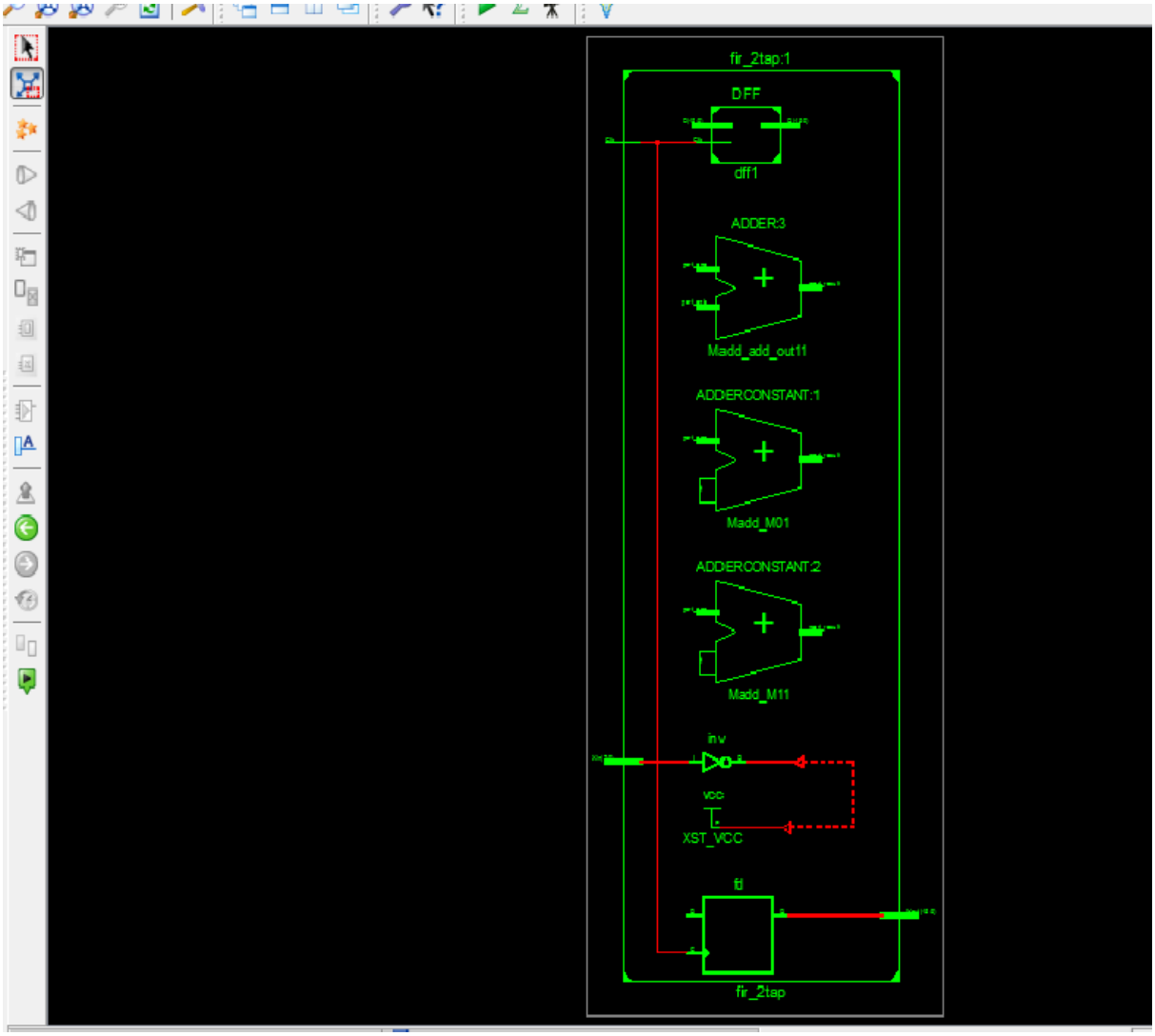
Main code for 2 Tap Filter:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL; arithmetic functions with Signed or Unsigned
values
entity fir_2tap is
```

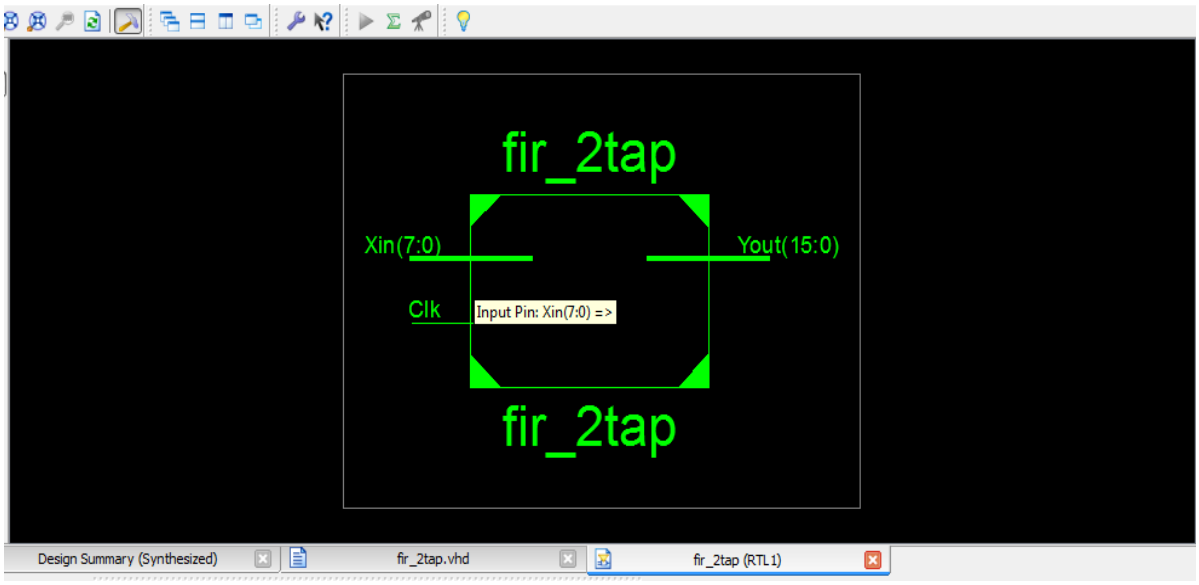
```
port(   Clk : in std_logic; --clock signal
        Xin : in signed(7 downto 0); --input signal
        Yout : out signed(15 downto 0)  --filter output
        );
end fir_2tap;

architecture Behavioral of fir_2tap is
component DFF is
  port(
    Q : out signed(15 downto 0);      --output connected to the adder
    Clk :in std_logic;      -- Clock input
    D :in  signed(15 downto 0)      -- Data input from the MCM block.
  );
end component;

signal H0,H1 : signed(7 downto 0) := (others => '0');
signal M0,M1,add_out1 : signed(15 downto 0) := (others => '0');
signal Q1 : signed(15 downto 0) := (others => '0');

begin

--filter coefficient initializations.
--H = [50 50].
H0 <= to_signed(50,8);
H1 <= to_signed(50,8);

-- Multiplications of H(i) with X(in).
M1 <= H1*Xin;
M0 <= H0*Xin;

--adders
--using only 1 adder for 2 tap filter
add_out1 <= Q1 + M0;
--flipflops(for introducing a delay).

dff1 : DFF port map(Q1,Clk,M1);

--an output produced at every positive edge of clock cycle.
process(Clk)
begin
```

```
    if(rising_edge(Clk)) then
        Yout <= add_out1;
    end if;
end process;
end Behavioral;
```

Block Implementation



RTL Schematic:

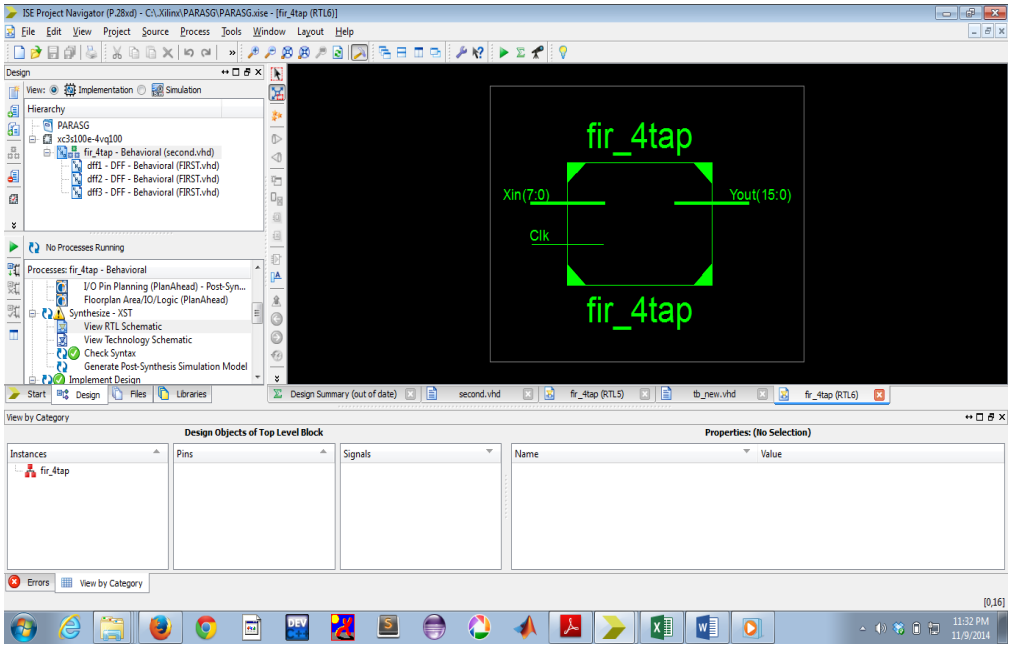


Main code for 4 Tap Filter:

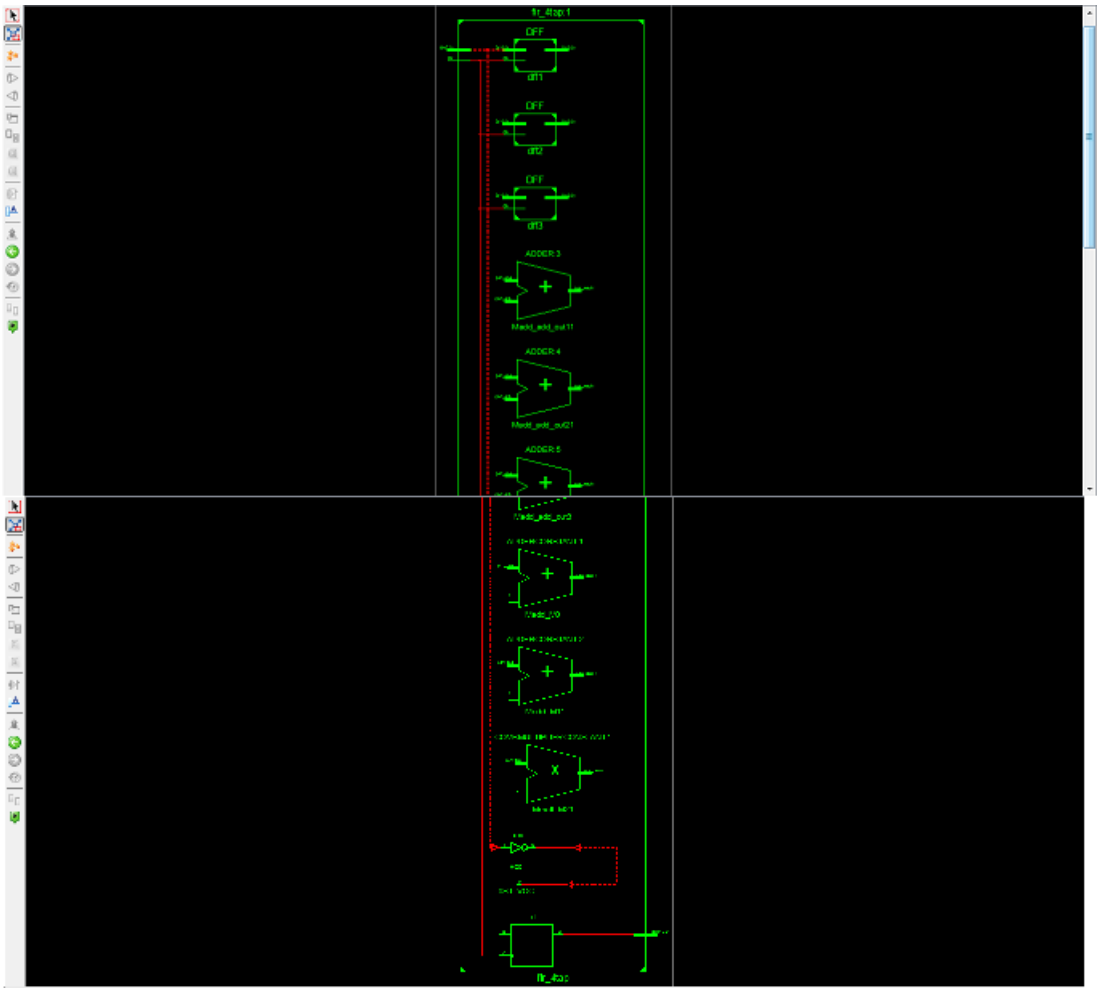
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity fir_4tap is
port(   Clk : in std_logic; --clock signal
        Xin : in signed(7 downto 0); --input signal
        Yout : out signed(15 downto 0)  --filter output
        );
end fir_4tap;

architecture Behavioral of fir_4tap is

component DFF is
  port(
    Q : out signed(15 downto 0);      --output connected to the adder
    Clk :in std_logic;      -- Clock input
    D :in  signed(15 downto 0)      -- Data input from the MCM block.
  );
end component;

signal H0,H1,H2,H3 : signed(7 downto 0) := (others => '0');
```

```
signal M0,M1,M2,M3,add_out1,add_out2,add_out3 : signed(15 downto 0) :=
(others => '0');
signal Q1,Q2,Q3 : signed(15 downto 0) := (others => '0');

begin

--filter coefficient initializations.
--H = [20 29 29 20].
H0 <= to_signed(20,8);
H1 <= to_signed(29,8);
H2 <= to_signed(29,8);
H3 <= to_signed(20,8);

-- multiplications.
M3 <= H3*Xin;
M2 <= H2*Xin;
M1 <= H1*Xin;
M0 <= H0*Xin;

--adders
add_out1 <= Q1 + M2;
add_out2 <= Q2 + M1;
add_out3 <= Q3 + M0;


--flipflops(for introducing a delay).
dff1 : DFF port map(Q1,Clk,M3);
dff2 : DFF port map(Q2,Clk,add_out1);
dff3 : DFF port map(Q3,Clk,add_out2);

--an output produced at every positive edge of clock cycle.
process(Clk)
begin
   if(rising_edge(Clk)) then
      Yout <= add_out3;
   end if;
end process;
end Behavioral;
```

RTL Schematic:



Block Implementation



Main code for 10 Tap Filter:

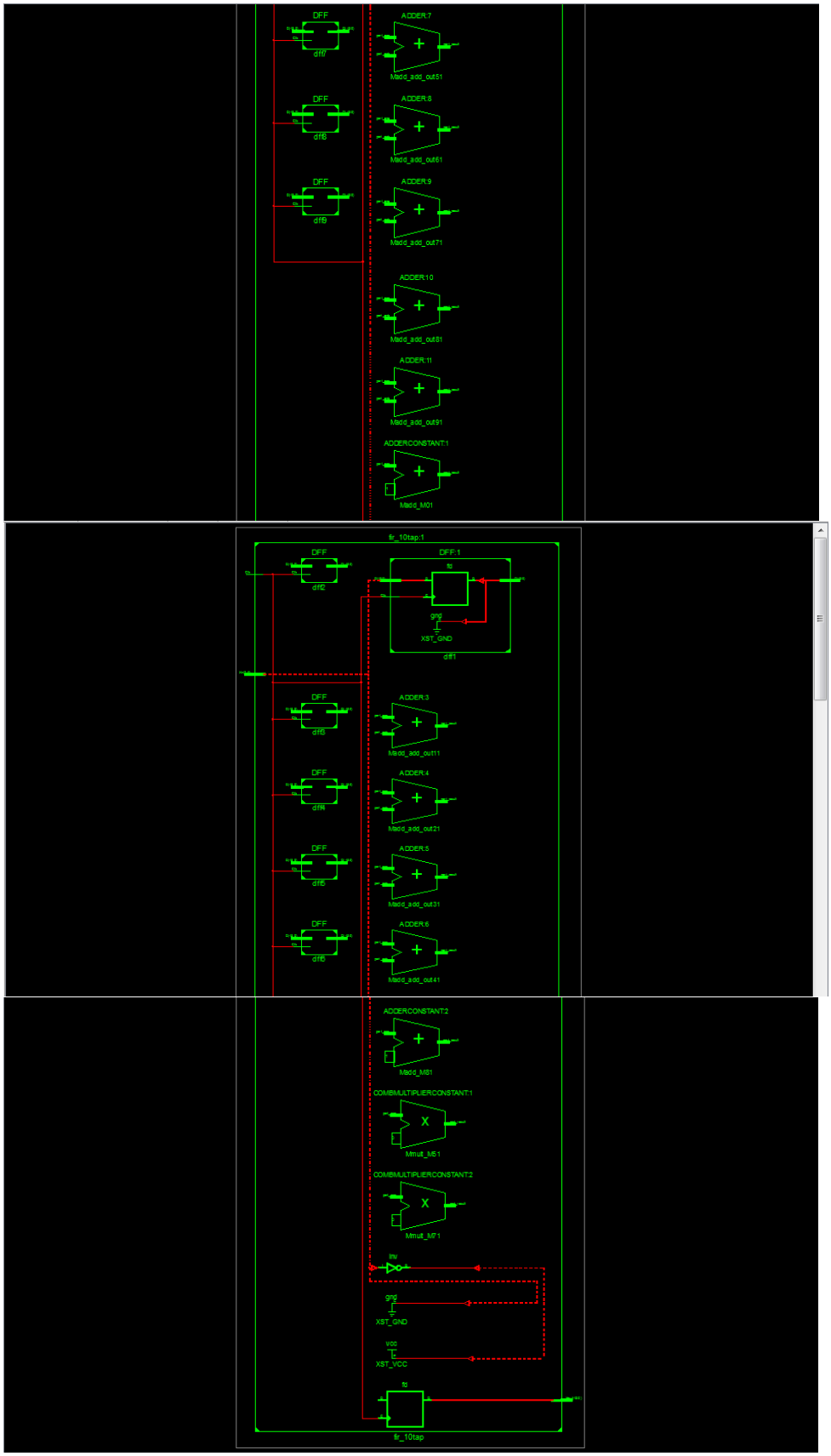
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
entity fir_10tap is
port(   Clk : in std_logic; --clock signal
        Xin : in signed(7 downto 0); --input signal
        Yout : out signed(15 downto 0)  --filter output
        );
end fir_10tap;

architecture Behavioral of fir_10tap is

component DFF is
  port(
    Q : out signed(15 downto 0);      --output connected to the adder
    Clk :in std_logic;      -- Clock input
    D :in  signed(15 downto 0)      -- Data input from the MCM block.
  );
end component;

signal H0,H1,H2,H3,H4,H5,H6,H7,H8,H9 : signed(7 downto 0) := (others => '0');
signal
M0,M1,M2,M3,M4,M5,M6,M7,M8,M9,add_out1,add_out2,add_out3,add_out
4,add_out5,add_out6,add_out7,add_out8,add_out9 : signed(15 downto 0) :=
(others => '0');
signal Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8,Q9 : signed(15 downto 0) := (others => '0');
begin
--filter coefficient initializations.
--H = [5 1 8 20 28 28 20 8 1 5].
H0 <= to_signed(5,8);
H1 <= to_signed(1,8);
H2 <= to_signed(8,8);
H3 <= to_signed(20,8);
H4 <= to_signed(28,8);
H5 <= to_signed(28,8);
H6 <= to_signed(20,8);
H7 <= to_signed(8,8);
H8 <= to_signed(1,8);
H9 <= to_signed(5,8);
```

```
-- multiplications.

M9 <= H9*Xin;
M8 <= H8*Xin;
M7 <= H7*Xin;
M6 <= H6*Xin;
M5 <= H5*Xin;
M4 <= H4*Xin;
M3 <= H3*Xin;
M2 <= H2*Xin;
M1 <= H1*Xin;
M0 <= H0*Xin;

--adders
add_out1 <= Q1 + M8;
add_out2 <= Q2 + M7;
add_out3 <= Q3 + M6;
add_out4 <= Q4 + M5;
add_out5 <= Q5 + M4;
add_out6 <= Q6 + M3;
add_out7 <= Q7 + M2;
add_out8 <= Q8 + M1;
add_out9 <= Q9 + M0;



--flipflops(for introducing a delay).
dff1 : DFF port map(Q1,Clk,M9);
dff2 : DFF port map(Q2,Clk,add_out1);
dff3 : DFF port map(Q3,Clk,add_out2);
dff4 : DFF port map(Q4,Clk,add_out3);
dff5 : DFF port map(Q5,Clk,add_out4);
dff6 : DFF port map(Q6,Clk,add_out5);
dff7 : DFF port map(Q7,Clk,add_out6);
dff8 : DFF port map(Q8,Clk,add_out7);
dff9 : DFF port map(Q9,Clk,add_out8);


--an output produced at every positive edge of clock cycle.
```

```
process(Clk)
begin
   if(rising_edge(Clk)) then
      Yout <= add_out9;
   end if;
end process;

end Behavioral;
```

RTL Schematic:

Block Implementation

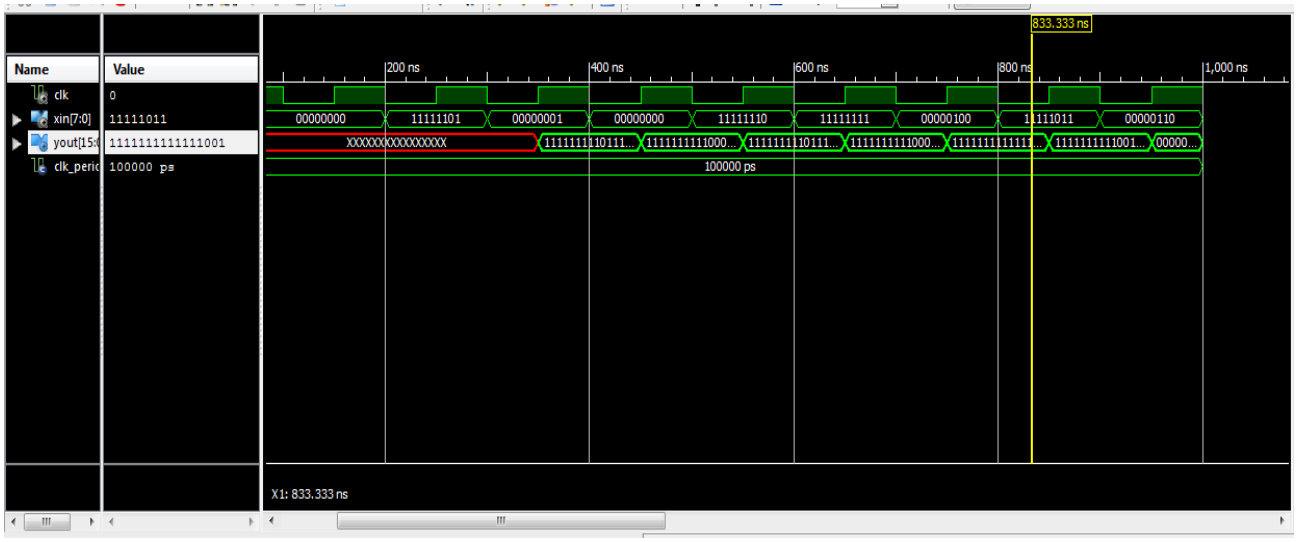

DFF
dff7
ADDER:7
Madd_add_out51
DFF
dff6
ADDER:8
Madd_add_out61
DFF
dff5
ADDER:9
Madd_add_out71
ADDER:10
Madd_add_out81
ADDER:11
Madd_add_out91
ADDERCONSTANT:1
Madd_M01
fir_10tap:1
DFF
dff2
DFF:1
dff1
XST_GND
DFF
dff5
ADDER:3
Madd_add_out11
DFF
dff4
ADDER:4
Madd_add_out21
DFF
dff3
ADDER:5
Madd_add_out31
DFF
dff2
ADDER:6
Madd_add_out41
ADDERCONSTANT:2
Madd_M81
COMBMULTIPLIERCONSTANT:1
Mmult_M51
COMBMULTIPLIERCONSTANT:2
Mmult_M71
inv
XST_GND
vcc
XST_VCC
fir_10tap

Simulation Results:

We simulated the result using Xilinx Simulator. The input we gave was 8 bit vector and filter response was also 8 bit vector. There was undefined value seen initially due to lack of instantiation of initial input values.

Then there was XXXXXX (no value) observed in the input which signifies the element delay generated in the designed system.

The test bench waveform output is:



Power Results:

FIR Filter Type	Power at 25 MHz (mWatts)	Power at 50 MHz (mWatts)	Power at 100 MHz (mWatts)	Delay (ns)
2- Tap Filter	**55**	62	76	18.424
4- Tap Filter	56	64	79	18.801
10- Tap Filter	57	66	82	18.924

We used **Xilinx Power Estimator (XPE),** downloaded the Spartan 3E XPE 11.1 file and gave the input parameters to the XPE file.

The lowest power was achieved in the 2-tap filter where we used 2 filter coefficients, 1 adder and 1 D-FF to generate one delay unit. The power achieved was **55** miliWatts compared to 82 miliwatts of 10- Tap filter.

Conclusion

Thus, we conclude that 2-Tap FIR Filter is the most efficient in terms of Power and Space acquired. Also, this filter is stable as compared to IIR filter. We are able to get the desired filter outputs using Kaiser Window method in MATLAB. We kept the cut-off frequency 150Hz and used the generated filter coefficients in design required filter. The overall delay is reduced as the number of delay register(D Flip Flop) and adders are substantially reduced while designing a low order FIR Filter.

REFERENCES

1. http://www.scribd.com/doc/96197076/Vhdl-Simulation-of-Fir-Filter
2. http://www.thecodingforums.com/threads/how-to-assign-a-hex-or-decimal-value-to-a-std_logic_vector-of-length19-bits.647395/
3. http://www.bitweenie.com/listings/vhdl-type-conversion/
4. DSD Lecture notes.