UNIVERSITÀ DI PISA

MSc in Computer Engineering
Electronic systems

FIR low pass filter

Project discussion and VHDL implementation

Gioele Carignani

2018/2019

Index

1 Introduction

In signal processing, a finite impulse response (FIR) filter is a filter whose impulse response, or response to any other finite length input, is of finite duration, because it settles to zero in finite time. The impulse response of an Nth-order discrete-time FIR filter lasts exactly N + 1 samples (from first nonzero element through last nonzero element) before it then settles to zero. For a causal discrete-time FIR filter of order N, each value of the output sequence is a weighted sum of the most recent input values:

$$y[n] = \sum_{i=0}^{N} c_i \cdot x[n-i] \tag{1}$$

1.1 Filter Design

When a particular frequency response is desired, several different design methods are possible, for example:

- Window design method: we first design an ideal IIR filter and then truncate the result by multiplying it with a finite length window function.
- Frequency Sampling method: this technique is the most direct technique imaginable when a desired frequency response has been specified. It consists simply of uniformly sampling the desired frequency response, and performing an inverse DFT to obtain the corresponding (finite) impulse response.
- Parks-McClellan method: The Remez exchange algorithm is commonly used to find an optimal equiripple set of coefficients. Here the user specifies a desired frequency response, a weighting function for errors from this response, and a filter order N. The algorithm then finds the set of N+1 coefficients that minimize the maximum deviation from the ideal.

1.2 Applications

Finite-impulse response (FIR) digital filter is widely used in several digital signal processing applications, such as speech processing, loud speaker equalization, echo cancellation, adaptive noise cancellation, and various communication applications, including software-defined radio (SDR) and so on. Many of these applications require FIR filters of large order to meet the stringent frequency specifications. Very often these filters need to support high sampling rate for high-speed digital communication. The number of multiplications and additions required for each filter output, however, increases linearly with the filter order.

2 Architectures

Different type of architectures are available depending on the speed and power constraints. To fulfill desired specifications the following optimizations can be done:

- Reduce total number of operations, in particular multiplications that are heavier
- Fixed-point arithmetic is cheaper and faster
- The area of a fixed point parallel multiplier is proportional to the product of the coefficient and data word lengths one could try to reduce their word length

Suppose a filter of order N has to be realized.

2.1 Direct form

This structure requires N memory locations for storing previous inputs and in terms of complexity it does N+1 multiplications and N additions. For building this structure the definition of Equation 1 is used to derive the equivalent circuit.

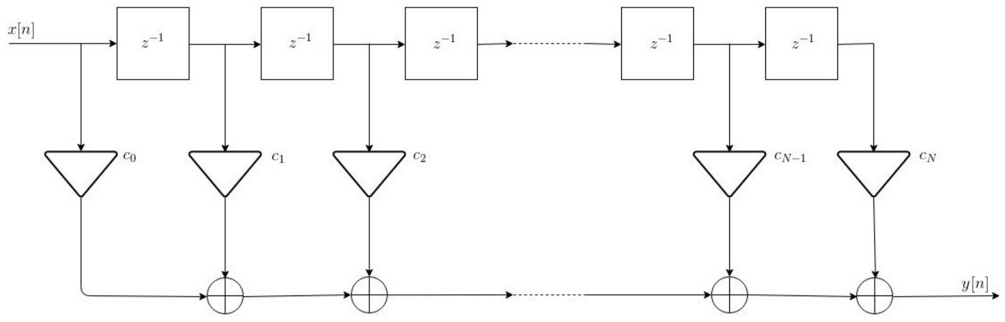


Figure 1: scheme of direct form

2.2 Transposed form

This structure is very similar at first look w.r.t. the previous one but in the former there was a big addition in the end, here instead there is a set of small additions separated by delay elements.

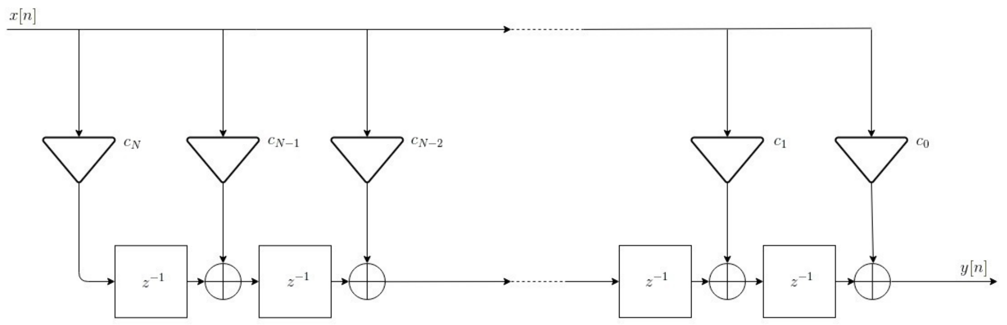


Figure 2: scheme of transposed form

2.3 Symmetric taps (linear phase)

FIR filters are often designed to have symmetry in the filter taps so this symmetry can be exploited in order to reduce the number of multiplications.

$$y[n] = c_0(x[n] + x[0]) + c_1(x[n-1] + x[1]) + ... + c_{N/2}(x[N/2])$$

The last term of the previous equation is needed only if N is even. With this type of implementation the number of multiplications is reduced to $\lfloor N/2 + 1 \rfloor$

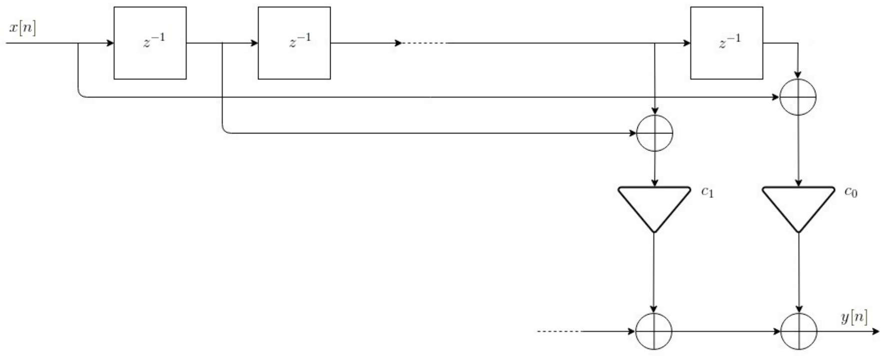


Figure 3: direct form with symmetric taps

2.4 Cascade form

The transfer function is expressed as product of second-order polynomial system functions via factorization:

$$H(z) = \frac{y[n]}{x[n]} = \sum_{i=0}^{N} c_i z^{-i} = \prod_{i=1}^{M_c} (\beta_{0i} + \beta_{1i} z^{-1} + \beta_{2i} z^{-2}) \tag{2}$$

Where $M_c = \lfloor (N+1)/2 \rfloor$. Assuming that N is even, this implementation needs N storage elements, $\frac{3N}{2}$ multiplications and N additions for each output value.

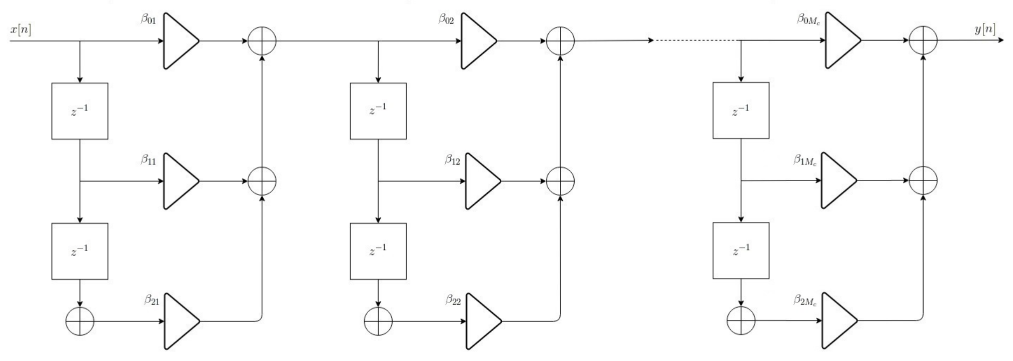


Figure 4: cascade form

To save computational complexity, Equation (2) is expressed as:

$$H(z) = G\prod_{i=1}^{M_c}(1 + \beta'_{1i}z^{-1} + \beta'_{2i}z^{-2}) \tag{3}$$

where $G = \beta_{01}\beta_{02}\cdots\beta_{0M_c}$,$\beta'_{1i} = \beta_{1i}/\beta_{0i}$, $i = 1, 2, \cdots, M_c$ in this way all β_{0i} are normalized to 1 as one can see in (Equation 3). Assuming that N is even N delay elements are needed, (N+1) multiplications and N additions, for computing each output value.

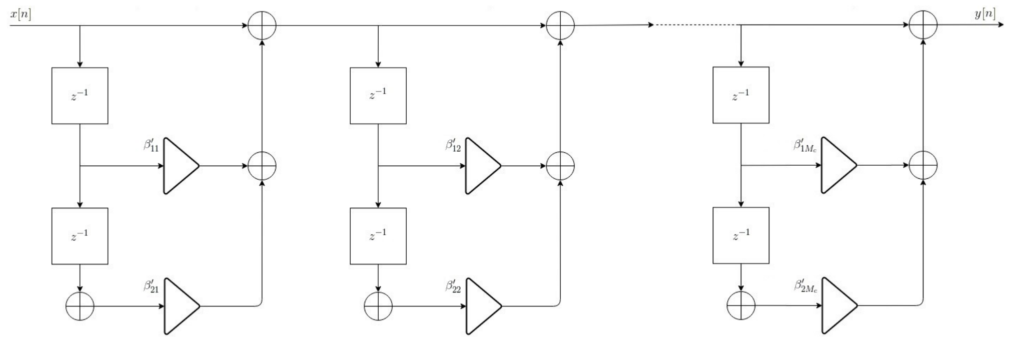


Figure 5: cascade form with lower complexity

3 Implemented Architecture

After analyzing the different architectures, the transposed form was implemented due to its overall realization simplicity and because it is the architecture that has the shortest path registry-logic-registry. Hence it permits to reduce the timing on the critical path and increase the clock frequency reachable when the VHDL code will be mapped on FPGA. A drawback w.r.t. the direct form is that this solution brings a higher latency for the input data, that will have to traverse the entire registry chain, nevertheless once the nework is at regime it will output new data per each clock cycle.

4 Integer conversion

The coefficients that are doubles were converted to integer on 16 bits through a quantization process performed by the following script:

```
b = 16;
lsb = -(-1) / (2^(b - 1)); %lsb for converting double to int
coeff = [0.0135,0.0785,0.2409,0.3344,0.2409,0.0785,0.0135]';
c_i = round(coeff/lsb);%integer coefficients
```

4.1 Output size

To realize the transposed architecture the result had to be sized:

$$\left\lceil log_2(2^{b-1} * \sum_{i=0}^{N} coeff_i) \right\rceil + 1 \tag{4}$$

Equation 4 returned a value for the output size of 32 bits.

5 Test Plan

In order to test the effectiveness of the filter the mathematical limits, given by the input size and on the coefficient values, had to be tested. To ease the testing of the filter with different inputs the test-bench inputs and outputs data from/to file.

5.1 Size limits

To stress out the size limits computed in section 4.1 the filter was fed with the maximum value in modulus (i.e. $-2^{b-1} = -32768$) repeated continuously and checked if the output corresponded to the value given by applying in Matlab expression 1 the following graphs are obtained:

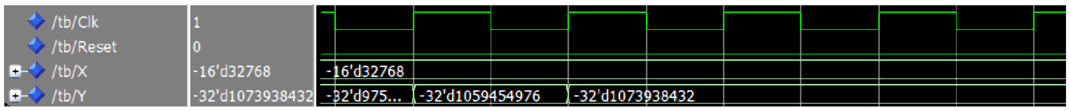


Figure 6: Test-bench of the filter with -32768 as constant input

The network at regime settles to -1073938432 that corresponds to the value returned by Matlab.

5.2 Prove of correctness

In order to check if the filter performed the convolution expressed in equation 1 a random input was tested and compared against the output of Matlab performing the same operation. The results once the network is at regime are matched by the output produced by Matlab convolution.

5.3 Filtering

As input to the network a square wave with a period of 14 samples was given. In Figure 7 the effect of filtering is highlighted where the first harmonic frequency is kept while other harmonics are cut converging in a sine wave.

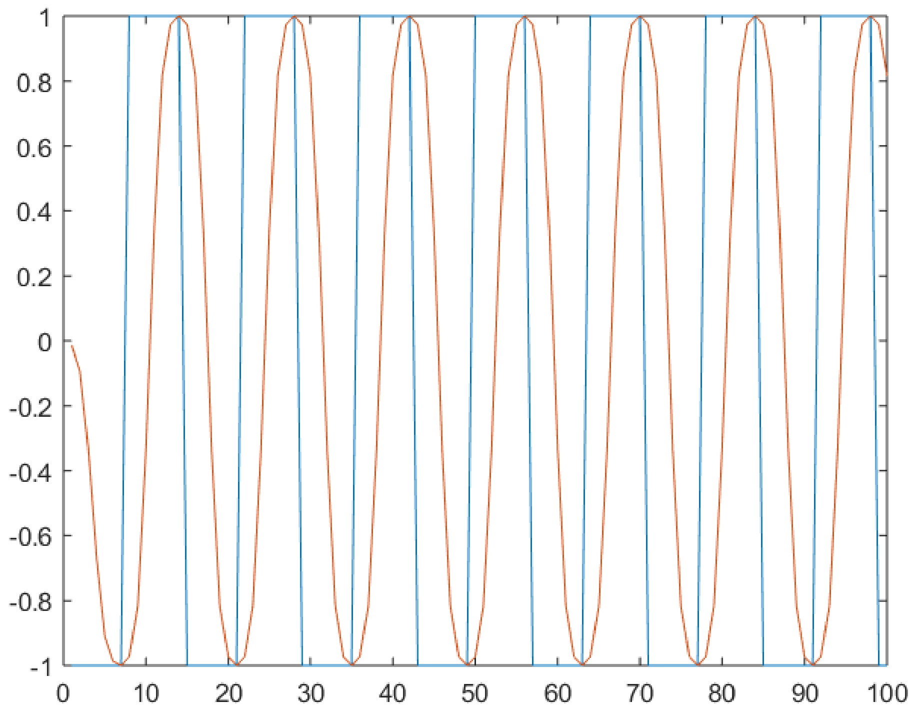


Figure 7: Filtered signal in red, input signal in blue

6 Synthesis

The hardware implementation of the project was carried out with the help of Vivado software by Xilinx. This software permits to map the VHDL code on Zynq-7000 board. To reach the final implementation and discover the maximum operating frequency the typical flow of Vivado was followed i.e. from VHDL code to RTL analysis, synthesis and as last implementation.

6.1 RTL Design

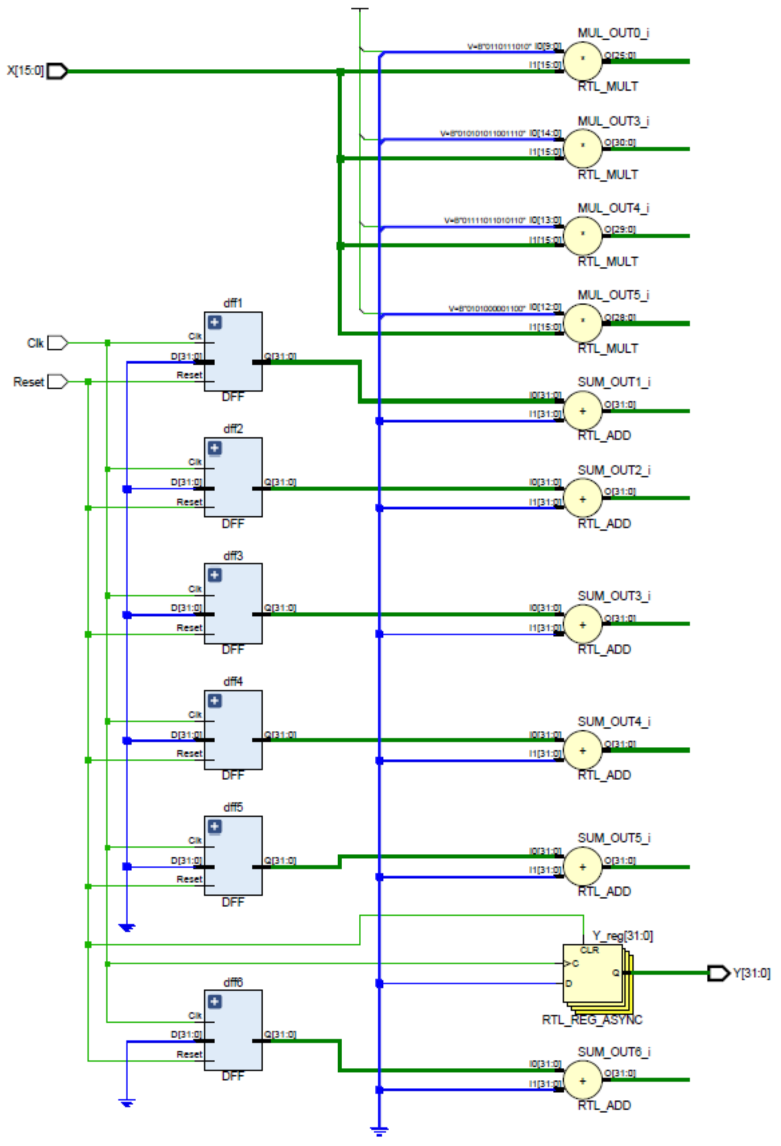


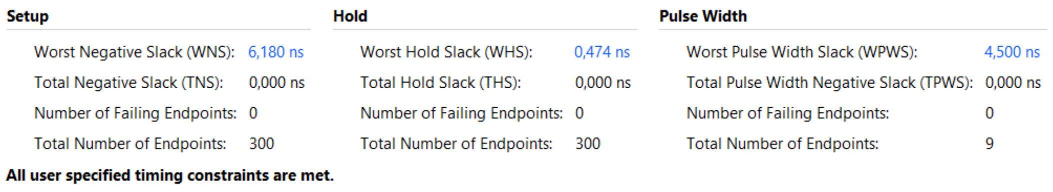
Figure 8: RTL design from vivado

6.2 Synthesis

A clock period of 10 ns was set.

6.2.1 Timing report

Timing report was fine and confirmed that all constraints are met:



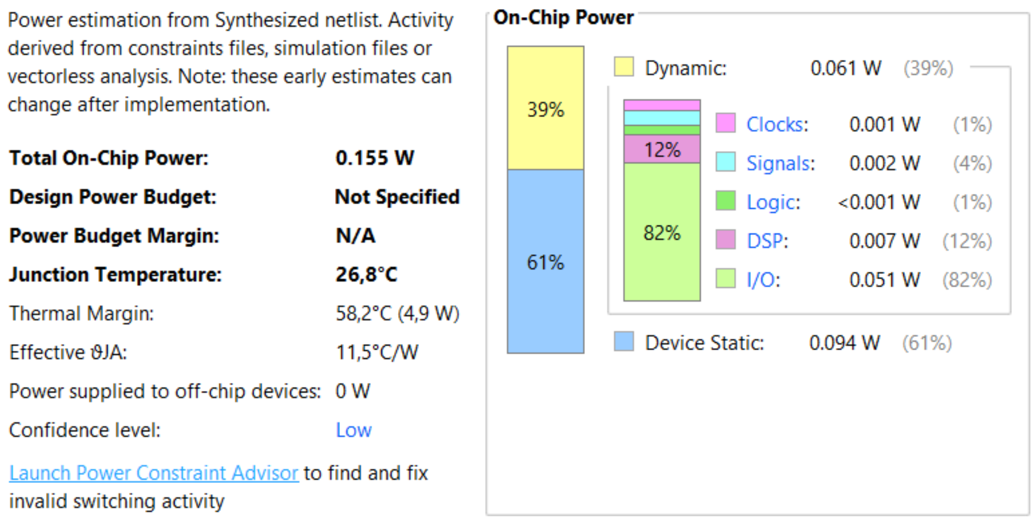
Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	6,180 ns	Worst Hold Slack (WHS):	0,474 ns	Worst Pulse Width Slack (WPWS):	4,500 ns
Total Negative Slack (TNS):	0,000 ns	Total Hold Slack (THS):	0,000 ns	Total Pulse Width Negative Slack (TPWS):	0,000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	300	Total Number of Endpoints:	300	Total Number of Endpoints:	9

All user specified timing constraints are met.

Figure 9: Timing summary from vivado

6.2.2 Power report

The dynamic power used by the device is mostly required by I/O operations as expected, and by DSP that is used for additions and multiplications.



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:	**0.155 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**26,8°C**
Thermal Margin:	58,2°C (4,9 W)
Effective ϑJA:	11,5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

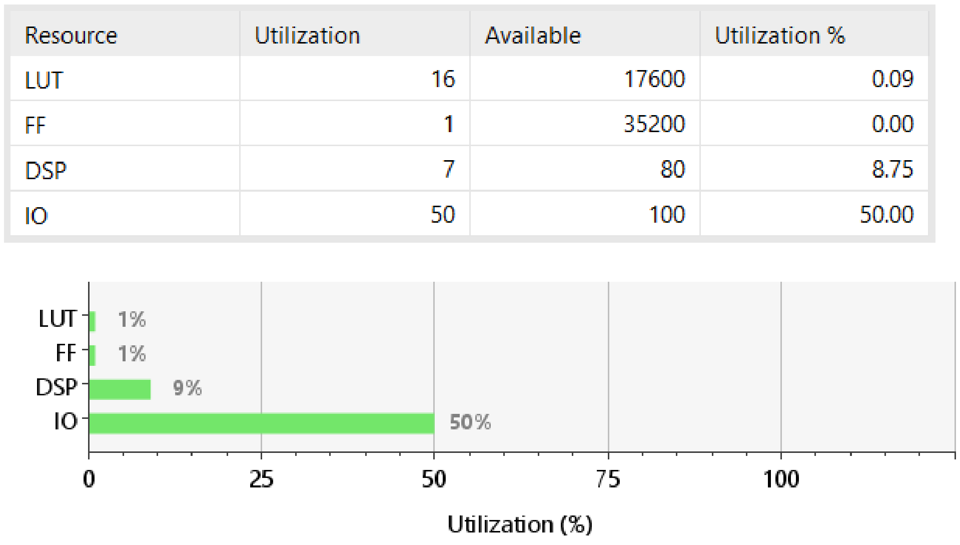
Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

Dynamic:	0.061 W	(39%)
Clocks:	0.001 W	(1%)
Signals:	0.002 W	(4%)
Logic:	<0.001 W	(1%)
DSP:	0.007 W	(12%)
I/O:	0.051 W	(82%)
Device Static:	0.094 W	(61%)

Figure 10: Power summary from vivado

6.2.3 Utilization report



Resource	Utilization	Available	Utilization %
LUT	16	17600	0.09
FF	1	35200	0.00
DSP	7	80	8.75
IO	50	100	50.00

Figure 11: Utilization summary from vivado

Figure 11 shows how the most widely used resource is the I/O this derives from the fact that the circuit has 16 bit input and a 32 bit output.

6.3 Critical path

Vivado shows how the critical path includes the sum and multiplication operation from the output of a registry till the input of the next registry.

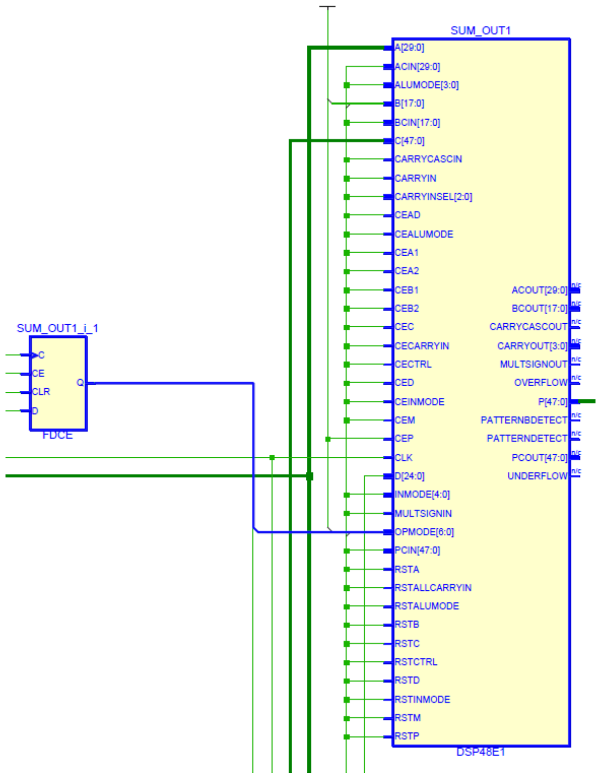
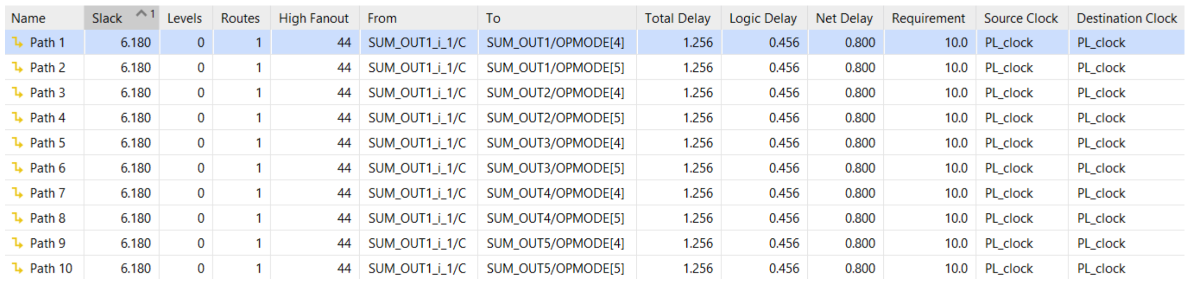


Figure 12: Critical path highlighted by vivado



Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock
Path 1	6.180	0	1	44	SUM_OUT1_i_1/C	SUM_OUT1/OPMODE[4]	1.256	0.456	0.800	10.0	PL_clock	PL_clock
Path 2	6.180	0	1	44	SUM_OUT1_i_1/C	SUM_OUT1/OPMODE[5]	1.256	0.456	0.800	10.0	PL_clock	PL_clock
Path 3	6.180	0	1	44	SUM_OUT1_i_1/C	SUM_OUT2/OPMODE[4]	1.256	0.456	0.800	10.0	PL_clock	PL_clock
Path 4	6.180	0	1	44	SUM_OUT1_i_1/C	SUM_OUT2/OPMODE[5]	1.256	0.456	0.800	10.0	PL_clock	PL_clock
Path 5	6.180	0	1	44	SUM_OUT1_i_1/C	SUM_OUT3/OPMODE[4]	1.256	0.456	0.800	10.0	PL_clock	PL_clock
Path 6	6.180	0	1	44	SUM_OUT1_i_1/C	SUM_OUT3/OPMODE[5]	1.256	0.456	0.800	10.0	PL_clock	PL_clock
Path 7	6.180	0	1	44	SUM_OUT1_i_1/C	SUM_OUT4/OPMODE[4]	1.256	0.456	0.800	10.0	PL_clock	PL_clock
Path 8	6.180	0	1	44	SUM_OUT1_i_1/C	SUM_OUT4/OPMODE[5]	1.256	0.456	0.800	10.0	PL_clock	PL_clock
Path 9	6.180	0	1	44	SUM_OUT1_i_1/C	SUM_OUT5/OPMODE[4]	1.256	0.456	0.800	10.0	PL_clock	PL_clock
Path 10	6.180	0	1	44	SUM_OUT1_i_1/C	SUM_OUT5/OPMODE[5]	1.256	0.456	0.800	10.0	PL_clock	PL_clock

Figure 13: Critical path list

6.4 Maximum operating frequency

From the timing report a positive Worst Negative Slack suggests that the circuit is not operating at the maximum frequency possible.

$$f_{max} = \frac{1}{t_{clk} - slack} = \frac{1}{10ns - 6.18ns} = \frac{1}{3.82ns} = 260Mhz \tag{5}$$

Equation 5 states that the maximum frequency at which the clock can be guided is 260 Mhz.

7 Implementation

The implementation was carried out without any warning. The data reported by the implementation confirms the conclusion derived from the synthesis step with slight differences such as the fact that the worst negative slack is now of $6.017ns$ hence the circuit has a maximum operating frequency of $251.1Mhz$ due to I/O ports being actually assigned and this brings some delay.

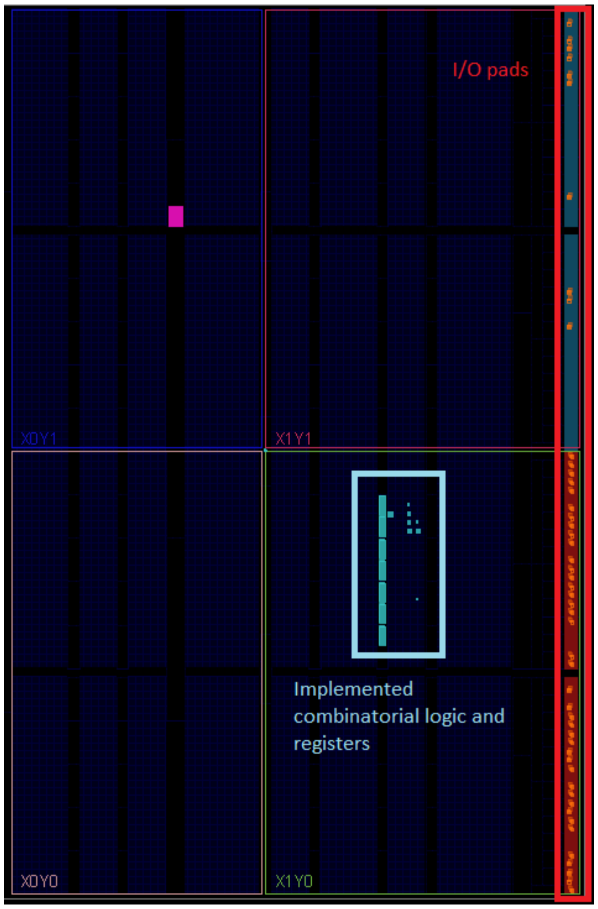


Figure 14: Implementation design from Vivado on Zybo board

From Figure 14 the high utilization I/O pads can be seen.

8 Conclusions

The work can be considered concluded, since the filter is compliant to its specifications. To reduce the number of multiplications the symmetric version of the filter could be developed since symmetric coefficients are given. The filter precision could be improved by increasing the number of steps obtaining a behavior more similar to an ideal filter. If further precision is required floating point numbers could be exploited at the cost of making the circuit more sophisticated.