International Journal of Reconfigurable and Embedded Systems (IJRES)
Vol. 4, No. 2, July 2015, pp. 63~70
ISSN: 2089-4864

FPGA Synthesis of Reconfigurable Modules for FIR Filter

Saranya R*, Pradeep C*, Neena Baby*, R Radhakrishnan**
*Department of Electronics and Communication, Saintgits College of Engineering, India
**Vidhya Mandhir Institute of Technology, Perundurai, Tamilnadu, India

Article Info

Article history:

Received Nov 13, 2014
Revised Dec 19, 2014
Accepted Jan 12, 2015

Keyword:

Data Path
Fault Detection
Fault Insertion
FIR Filter
FPGA
Reconfiguration

ABSTRACT

Reconfigurable computing for DSP remains an active area to explore as the need for incorporation with more conventional DSP technologies turn out to be obvious. Conventionally, the majority of the work in the area of reconfigurable computing is aimed on fine grained FPGA devices. Over the years, the focus is shifted from bit level granularity to a coarse grained composition. FIR filter remains and persist to be an important building block in various DSP systems. It computes the output by multiplying input samples with a set of coefficients followed by addition. Here multipliers and adders are modeled using the concept of divide and conquer. For developing a reconfiguarble FIR filter, different tap filters are designed as separate reconfigurable modules. Furthermore, there is an additional concern for making the system fault tolerant. A fault detection mechanism is introduced to detect the faults based on the nature of operands. The reconfigurable modules are structurally modeled in Verilog HDL and simulated and synthesized using Xilinx ISE 14.2. A comparison of the device utilization of reconfigurable modules is also presented in this paper by implementing the design on various Virtex FPGA devices.

Corresponding Author:

Saranya R,
Departement of Electronics and Communication,
Saintgits College of Engineering,
Kottukulam Hills, Pathamuttom P.O, Kottayam-686532.
Email: saranya181190@gmail.com

1. INTRODUCTION

Owing to the expeditious advance in algorithms and applications in Digital Signal Processing (DSP), there has been an emerging necessity for adaptable architectures with dynamic reconfiguration capabilities. The area of reconfigurable computing has been inspired by the considerable changes in the application space of DSP. FIR filters are used in majority of the DSP based electronic systems. The disclosure of challenging applications in requisites of power, speed, performance and reusability make it vital to design reconfigurable architectures.

FIR filters has extensive applicability but it may require a large number of coefficients to get the preferred requirement. Hence it results in the large number of slice for the FPGA design. Therefore, dynamic reconfigurable design of higher order tap FIR filters using traditional FPGA design techniques has certain disadvantages. One of the important disadvantages is the configuration time, which is the time used up for reconfiguration. This relies on the reconfigurable device and the technique of reconfiguration. Partial Reconfiguration can be used in this case. Partial reconfiguration results in less configuration time, provides flexibility and area competence for higher order FIR filters.

In this paper, we present the synthesis of reconfigurable modules (2-tap and 3-tap) for both serial and piplined FIR filters. The reconfigurable modules are modelled using an area efficient data path with online fault detection mechanism [2] based on the nature of operands. Separate multiplier and adder blocks are designed based on the concept of divide and conquer [1] approach to model the FIR filter. A well-known

technique to attain fault detection is duplication with comparison where similar calculation is performed twice and the outputs are compared to recognize errors. In order to detect a fault [12], fault needs to be introduced into the circuit. For the verification of the fault detection, faults are introduced in the '.xdl' file of the data path [2]. Design of reconfigurable modules is done in Verilog HDL. It is simulated and synthesized using Xilinx ISE 14.2. The design is implemented on various Virtex FPGA devices to obtain the better device utilization. The synthesis result is used to obtain the performance of the system in terms of area and delay.

The rest of the paper is organized as follows. Section 2 gives a brief description of related works in this area. Section 3 describes the architecture of the reconfigurable modules. Section 4 presents the evaluation and synthesis results. Finally the concluding remark and future work is presented in section 5.

2. RELATED WORKS

The idea of reconfigurable computing has been in existence for quite sometime [11]. Wang Lie and Wu Feng-yan published a paper on Dynamic Partial Reconfiguration (DPR) [4] in FPGAs and illustrated the advantages of early access partial reconfiguration. A reconfigurable FIR filter [14] design using dynamic partial reconfiguration is presented, which has area efficiency and flexibility allowing dynamical insertion and removal of partial modules using Xilinx Virtex-2 XC2v6000 FPGA. A digit-reconfigurable FIR filter [13] architecture with fine granularity is presented and implemented using CMOS technology.

The concept of extendable and reusable arithmetic units [10] have been proposed by several groups since it affects the performance of the entire system. An area and time efficient reconfigurable arithmetic unit [5], a low power low cost computational model for multimedia applications [6], organisation and FPGA implementation of MORA processor core [7] are proposed. Performance comparison of different previous reconfigurable data paths was illustrated and two new processing elements were proposed and its evaluation is done [8]. However previous approaches have many disadvantages such as complexity in the interconnection network, less flexibility etc and were overcome in the data path proposed by Purohit et al [1]. It can perform N-bit addition, multiplication, subtraction and accumulation operations.The proposed data path is reconfigurable and performs all these operations based on the control signals of the multiplexers.

3. PROPOSED ARCHITECTURE

Many of the applications involve repetitive arithmetic operations and therefore the design of arithmetic data path is important since it affects the overall performance of the system. The performance of the system is generally determined by the multiplier block since it is the most time consuming operation [3]. Hence the speed and area of the multiplier must be optimized and here the multiplier block is optimized by using the concept of divide and conquer approach [1]. Another major challenge is to guarantee fault free operation of the circuit since it affects the reliability of the system. The basic idea of an FIR filter is simple: the current output is obtained by multiplying the current input value by a constant, and adding that result to the previous input value times a constant, and adding that result to the next earlier input value times a constant, and so on. Consider a 3-tap FIR filter. It has 3 filter coefficients and can be described as follows:

$$y(t) = c_0 \times k(t) + c_1 \times k(t-1) + c_2 \times k(t-2) \quad (1)$$

'c_0',' c_1' and 'c_2' are the filter coefficients; 'k' and 'y' are the input and output signals, and 't' is the present time step. Equation (1) can be implemented by using registers, multipliers and adders. Both serial and pipelined FIR filters are modelled using the data path shown in Figure 2. Multiplier blocks are implemented by using the concept of divide and conquer approach. Addition process is also implemented similarly. Wallace tree multiplier [9] is used for both processesses.

3.1. Serial and Pipelined FIR Filters

Serial and pipelined 3-tap FIR filters [15] are shown in Figure 1. It consists of registers, multipliers and adders. Registers 'rt0', 'rt1' and 'rt2' are needed for each tap to hold $k(t)$, $k(t-1)$ and $k(t-2)$ respectively. For serial filter, the data moves to the right on each clock cycle, so that the register 'rt0' holds the current input sample, 'rt1' holds the previous input sample and 'rt2' holds the sample before the previous one [15]. The throughput of the serial filter can be improved by using pipelining. Pipelining means to break a large task down into a sequence of stages such that data moves through these stages and each stage produce output used by the next stage. All the stages operate concurrently resulting in better performance. It involves the addition of registers between the stages and these registers are known as pipeline registers. Here all the

inputs are available at the multiplier input at the same clock. All the multiplication operations are done in parallel and the individual results are stored in pipeline registers. We also need a multiplier for each tap to multiply the tap's '*k*' value by the constant '*c*' value. The output '*y*' is the sum of each tap's product. The 2-tap FIR filter has 2 coefficients and the structure is similar to Figure 1 but has only two registers and multipliers and a single adder.

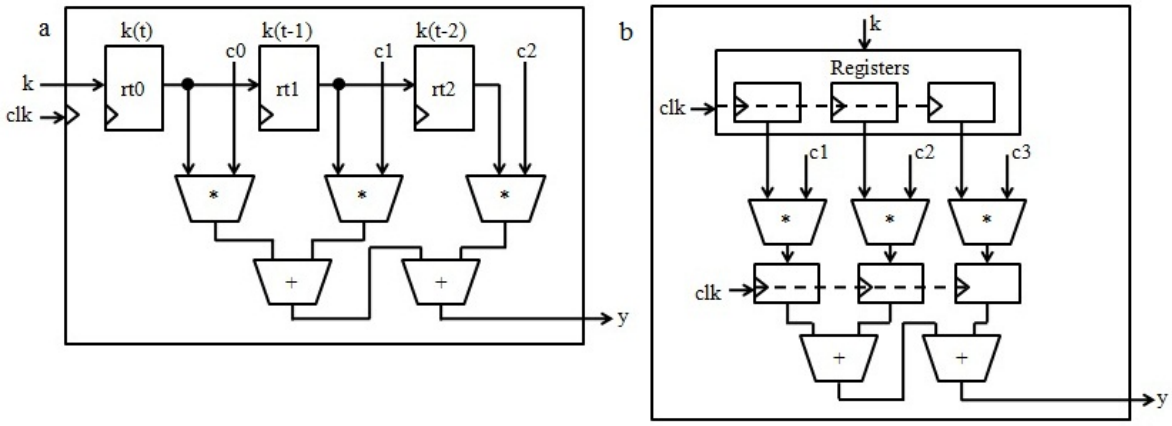

Figure 1. (a) Serial 3-tap filter (b) Pipelined 3-tap filter

3.2. Data Path Architecture

The architecture of the data path is shown in Figure 2. The data path can perform N-bit multiplication and addition operations. An online logic fault detection mechanism [2] is incorporated into the data path. The data paths consist of two N x N/2 Wallace tree multipliers, compressors, adders and carry completion logics. It also contains an equality comparator, equality detectors, AND gates and multiplexer which are used for fault detection. A technique known as divide and conquer approach is used. It involves breaking up of the multiplier into two that is, breaking up of the large multiplication into two smaller ones and adds up the partial products generated to obtain the final result.

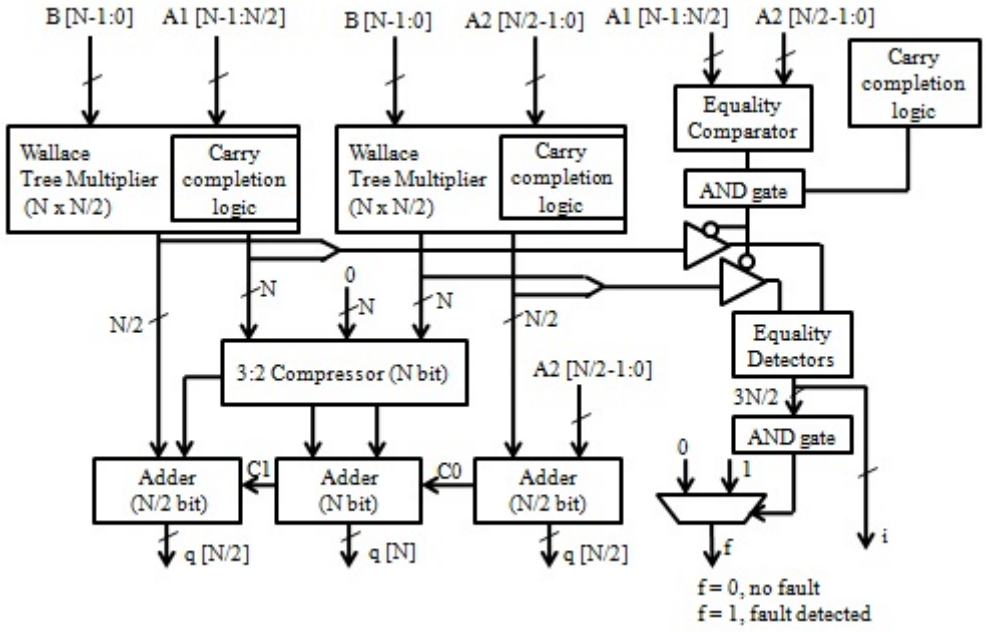

Figure 2. Generalized architecture of the data path

Consider two N-bit numbers ‘A’ and ‘B’. The number ‘A’ is equally divided into two numbers namely A1 [N-1: N/2] and A2 [N/2-1:0]. Two multiplication operations are performed simultaneously by using two Wallace tree multipliers and the result is then compressed using 3:2 compressors. The intermediate products are then added to obtain the final result. For multiplication process, the two multipliers perform B [N-1:0] x A1 [N-1: N/2] and B [N-1:0] x A2 [N/2-1:0]. The intermediate products are added using 3:2 compressors and ‘0’ is the third input to the compressor. The input to adder1 is ‘0’. The ‘N/2’ least significant bits of B [N-1:0] x A2 [N/2-1:0] forms the LSB bits of the final result. The ‘N’ most significant bits of B [N-1:0] x A2 [N/2-1:0] are added with the ‘N’ LSB bits of B [N-1:0] x A1 [N-1:N/2] to obtain the ‘N’ intermediate bits of the final result and the ‘N/2’ MSB bits of B [N-1:0] x A1 [N-1:N/2] forms the MSB of the final result. For addition operation, B [N-1:0] given to the left multiplier and A2 [N/2-1:0] given to the right multiplier is ‘1’. The multiplier on the left and right performs A1 [N-1:N/2] x 1 and B [N-1:0] x 1 respectively. The output of the compressor is added along with A2 [N/2-1:0] to obtain the final computation A+B.

In the data path two N x N/2 multipliers are used. By exploiting this feature of Dual Modular Redundancy (DMR), an online fault detection mechanism [2] is introduced into the data path. DMR uses two equivalent functional units and provides fault detection when the units that should give the same results give different results. If the operands A1 [N-1: N/2] and A2 [N/2-1:0] are identical then a fault can be detected. Initially by using an equality comparator that consists of an array of XNOR gates it is verified that whether A1 [N-1: N/2] and A2 [N/2-1:0] are identical or not. The output of the equality comparator along with the output of carry completion logic is given as the inputs to an AND gate. The output of the AND gate is given to the enable pin of a tri state buffer. The output of the two Wallace tree multipliers is given as the inputs of the tri state buffers. The outputs of the tri state buffers are then given to an equality detector that contains an array of XNOR gates. The output of the equality detectors are ANDed together and it is given to the enable pin of the multiplexer. If both the inputs of the equality comparator are same and the ouput of carry completion logics is ‘1’, the two tri state buffers are enabled. Then the outputs of the tri state buffers are available at the inputs of the equality detector. The output of the equality detector is given as the input of an AND gate and if its output is ‘0’, the circuit is fault free otherwise it is faulty. If the inputs of the equality comparator are different then its ouput is ‘0’ and even if the output of the carry completion logic is ‘1’, the tri state buffers get disabled and no fault detection mechanism is performed. This is an online fault detection mechanism since it donot require the system to shut down to detect the fault [2].

Carry completion detection logic is used to recognize the completion of multiplication operation. It is necessary since the fault detection is performed after the multiplication operation. Figure 3(a) shows the logic circuit of carry completion detection logic. It consists of Carry Transmission (CT) units, Full Adders (FAs), OR gates and a carry completion gate which is the n-input AND gate. Here, the inputs (A0 to An-1 and B0 to Bn-1) are given to both the FAs and the CT units. The sum outputs are obtained from the FAs and the carry outs and their compliments are obtained from the CT units. The output of each CT unit is given to an OR gate. The outputs of all the OR gates are then given to a carry completion gate. The value of carry completion signal signifies the completion of the operation. If the carry completion signal is ‘1’ then it indicates that the final carry is obtained and the multiplication operation is complete. Figure 3(b) shows the CT unit and is used to generate the carry signal and its compliment.

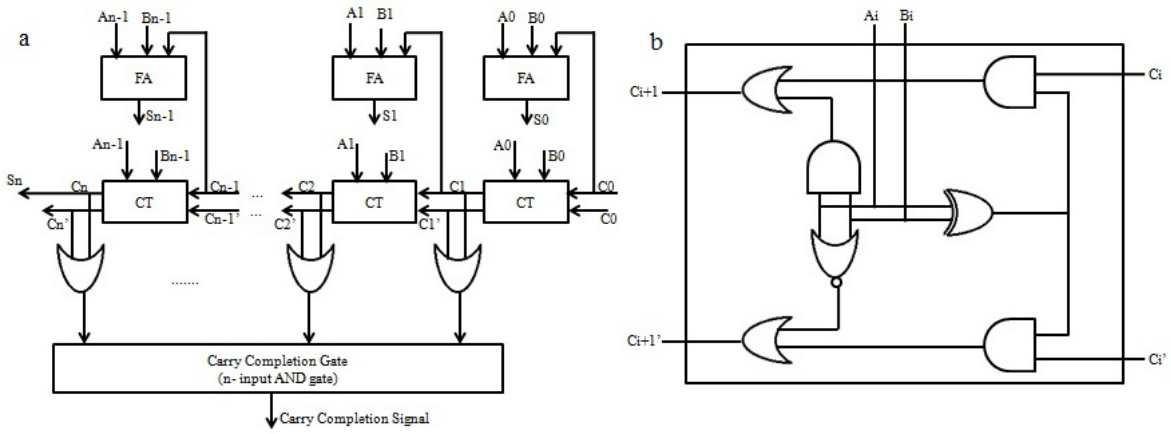

Figure 3. (a) Carry completion logic (b) Carry Transmission unit

In order to detect fault, fault needs to be introduced into the circuit. This can be done by converting the Native Circuit Description (NCD) file (non readable) of the fault free circuit which is obtained after the place and route process into Xilinx Design Language (XDL) by using a tool –XDL and this readable XDL file has to be edited to introduce fault. This file is then converted back to NCD and the NCD file back to Verilog netlist file and is simulated by applying the test vectors using test bench and is verified [2].

By using the above data path, 2-tap and 3-tap serial and pipelined FIR filters are modelled. These are the reconfigurable modules for making the FIR filter partially reconfigurable. Figure 4 shows the reconfigurable modules for the FIR filter. These modules are then synthesized separately to obatin the netlist and implemented on various Virtex FPGAs to determine the better device utilization.

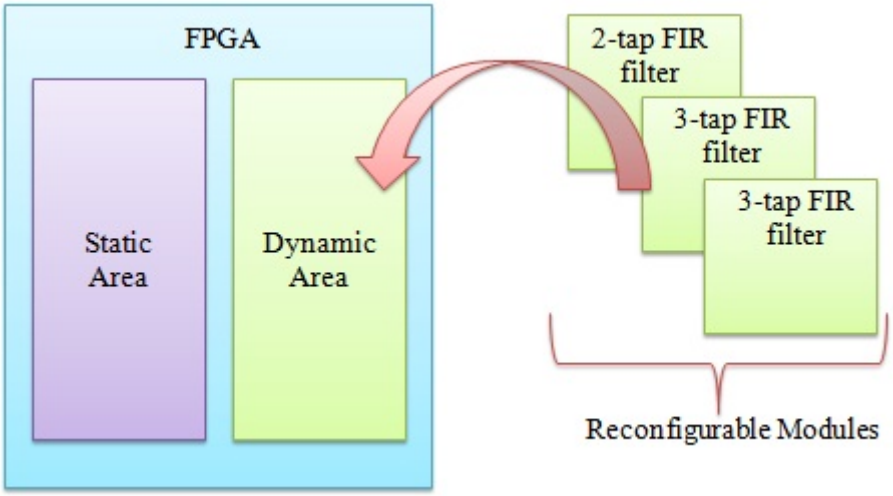

Figure 4. Reconfigurable modules for FIR filter

4. RESULT AND ANALYSIS

Inorder to evaluate the feasibility of both serial and pipelined FIR filters, it is designed using Verilog HDL and synthesized using Xilinx ISE 14.2. The input and coefficients of both the filters are assumed to be 8-bit. We implemented 8-bit (N=8) multiplication and 16-bit (N=16) addition for designing the filter. A half adder is also used to add the final two bits.

First, the functional simulation of both serial and pipelined FIR filter is done by using ISim with exhaustive test bench. Successful simulation is then followed by the synthesis process with various Xilinx FPGA's to obtain the maximum combinational path delay and device utilization for FPGA implementation.

Figure 5 and Figure 6 shows the simulation waveform for serial and pipelined FIR filter before and after fault injection. Here 'k' is the input, 'c0', 'c1' and 'c2' are the coefficients and 'y' is the output. 'i1', 'i2' and 'i3' are the output of equality detectors. 'f1', 'f2' and 'f3' is the output of fault detection circuit. When the value of either of c0 [7:4] and c0 [3:0], c1 [7:4] and c1 [3:0], c2 [7:4] and c2 [3:0] are same, the corresponding equality detector becomes active and before fault injection the output of fault detection circuit is '0'. When the values are different, then the equality detector becomes inactive and fault detection is not performed as shown in Figure 5. After fault injection, the output of the corresponding fault detection circuit becomes '1' as shown in Figure 6. Table 1 and Table 2 show the comparison of device utilization and combinational path delay of 2-tap and 3-tap serial and pipelined FIR filter on various FPGA devices. The proposed data path utilizes less number of slices, LUTs and IOBs when it is implemeted on Virtex-5. But when it is implemented on Virtex-4 and Virtex-6, the device utilization is more. So it is clear that Virtex-5 has better device utilization. The designs have different delays when implemented on various devices. It is clear that the pipelined FIR filter has less delay as compared to serial FIR filter. Figure 7 shows the comparison of 2-tap and 3-tap serial and pipelined FIR filters on various FPGA devices.

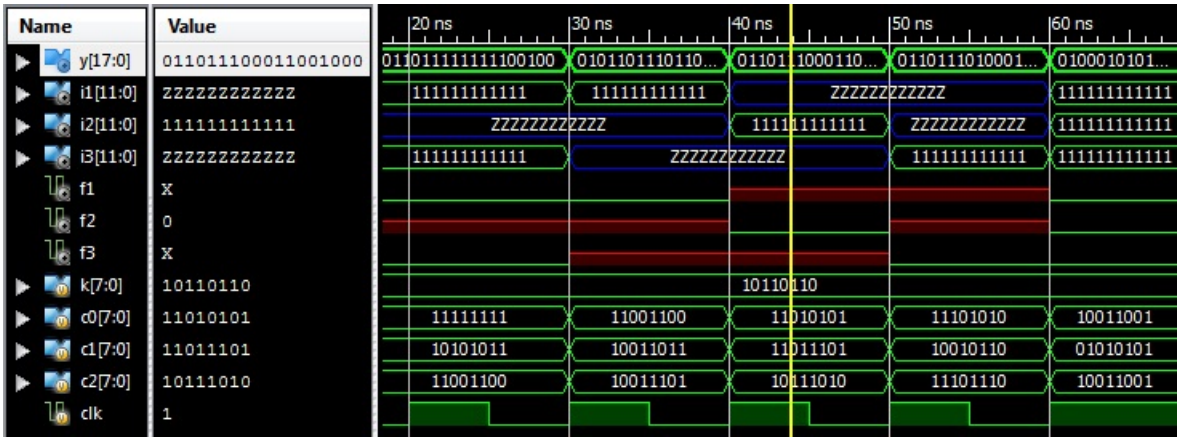

Figure 5. Simulation waveforms for FIR filter before fault injection

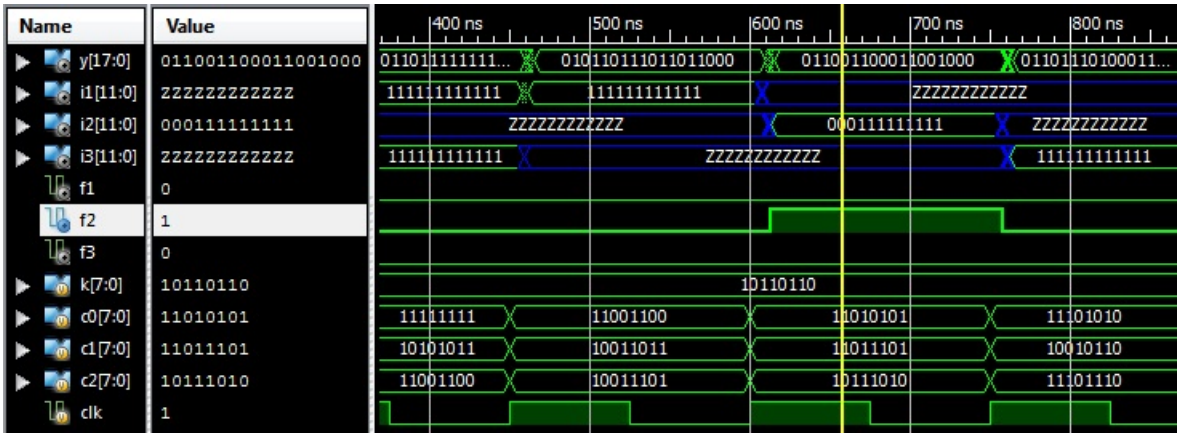

Figure 6. Simulation waveforms for FIR filter after fault injection

Table 1. Comparison of device utilization and combinational path delay of 2-tap serial and pipelined FIR filter

Device	Architecture	No. of occupied Slices (%)	No. of LUTs (%)	No. of IOBs (%)	Maximum combinational path delay (ns)
Virtex-4 (XC4VFX12)	Serial FIR filter	4	3	18	22.924
	Pipelined FIR filter	4	3	18	20.284
Virtex-5 (XC5VLX110T)	Serial FIR filter	1	1	7	16.841
	Pipelined FIR filter	1	1	7	14.268
Virtex-6 (XC6VCX75T)	Serial FIR filter	1	1	18	12.024
	Pipelined FIR filter	1	1	18	9.898

Table 2. Comparison of device utilization and combinational path delay of 3-tap serial and pipelined FIR filter

Device	Architecture	No. of occupied Slices (%)	No. of LUTs (%)	No. of IOBs (%)	Maximum combinational path delay (ns)
Virtex-4 (XC4VFX12)	Serial FIR filter	6	6	22	24.648
	Pipelined FIR filter	6	6	22	22.012
Virtex-5 (XC5VLX110T)	Serial FIR filter	1	1	8	18.696
	Pipelined FIR filter	1	1	8	15.928
Virtex-6 (XC6VCX75T)	Serial FIR filter	1	1	22	17.411
	Pipelined FIR filter	1	1	22	15.456

5. CONCLUSION AND FUTURE WORK

In this paper, we presented the design and synthesis of reconfigurable modules (2-tap and 3-tap) for developing a partially reconfigurable FIR filter. The data path was based on the concept of divide and conquer approach. By using this concept, the number of gates was reduced and thus the FIR filter implemented using this data path was area efficient. An online fault detection technique was also incorporated into the data path and a fault insertion technique was described to insert the faults so that by using the test vectors the faults can be detected. The implementation result shows that the reconfigurable modules have lower device utilization when implemented on Virtex-5 (XC5VLX110T-1FF1136T) FPGA.

Our future effort is to design a reconfigurable FIR filter that aims to attain all the objectives on the FPGA, which are set by Dynamic Partial Reconfiguration (DPR) by using a 2-tap and two 3-tap filters as reconfigurable modules in PlanAhead 14.2 on Virtex-5 (XC5VLX110T-1FF1136T) FPGA which supports Partial Reconfiguration. Partial Reconfiguration is the ability to reconfigure selected areas of an FPGA any time after its initial configuration [4].

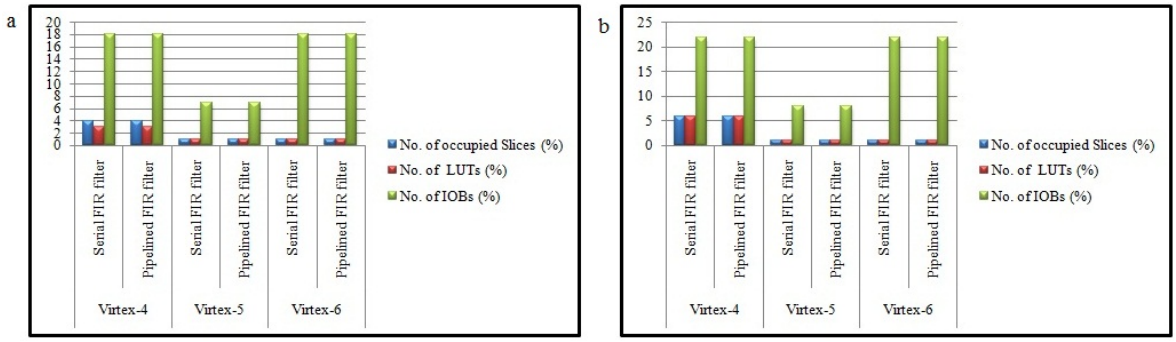

Figure 7. Comparison of serial and pipelined FIR filters on various devices (a) 2-tap (b) 3-tap

ACKNOWLEDGEMENT

The author would like to acknowledge the support of the Department of Electronics and Communication, Saintgits College of Engineering, Kottayam for technical assistance in simulations.

REFERENCES

[1] Sohan S. Purohit, Sai Rahul Chalamalasetti, Martin Margala, Wim A. Vanderbauwhede, "Design and Evaluation of High-Performance Processing Elements for Reconfigurable Systems", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 21, No. 10, October 2013.

[2] Pradeep C, Radhakrishnan R, Saranya R, Philip Samuel., "Synthesis of Data Path Architecture with Online Fault Detection Mechanism for Reconfigurable Systems", *Aust. J. Basic & Appl. Sci*., 8(10): 239-245, 2014.

[3] Rong Lin, "Reconfigurable Parallel Inner Product Processor Architectures", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 9, No. 2, April 2001.

[4] Wang Lie, Wu Feng-yan, "*Dynamic partial reconfiguration in FPGAs*", 2009 Third International Symposium on Intelligent Information Technology Application.

[5] Sotiris Xydis, George Economakos, Kiamal Pekmestzi, "Flexibility niming into Arithmetic Data-paths Exploiting a Regular Interconnection Scheme", *IEEE* 2007.

[6] S. Chalamalasetti, W. Vanderbauwhede, S. Purohit, and M. Margala, "*A low cost reconfigurable soft processor for multimedia applications: Design synthesis and programming model*", in Proc. Int. Conf. Field Program. Logic Devices, 2009, pp. 534–538.

[7] S. Chalamalasetti, S. Purohit, M. Margala, W. Vanderbauwhede, "*MORA-an architecture and programming model for a resource efficient coarse grained reconfigurable processor*", in Proc. 4th NASA/ESA Conf. Adapt. Hardw. Syst., San Francisco, CA, 2009, pp. 389-396.

[8] S. Purohit, S. Chalamalasetti, M. Margala, P. Corsonello, "*Power-Efficient High Throughput Reconfigurable Datapath Design for Portable Multimedia Devices*", in Proceedings of International Conference on Reconfigurable Computing and FPGAs, pp. 217-222, December 2008.

[9] C.S. Wallace, "A suggestion for a fast multiplier", *IEEE Trans. Electron. Comput*, vol. 13, no. 1, pp. 14–17, Feb. 1964.

[10] Yunan Xiang, Ryan Pettibon, Martin Margala, "*A Versatile Module for Adaptable Multimedia Processors*", Circuits and Systems, 2006, ISCAS Proceedings. IEEE International Symposium on 21-24 May 2006.

[11] Katherine Compton, "Reconfigurable Computing: A Survey of Systems and Software", *ACM Computing Surveys*, vol.34, no.2. pp. 171-210, June 2002.

[12] Rajamani Doraiswami, Chris P. Diduch, and Jiong Tang,"A New Diagnostic Model for Identifying Parametric Faults", *IEEE Transactions on Control Systems Technology*, vol. 18, no. 3, May 2010.pp 533-544.
[13] Yeong-Jae Oh et al., "A Reconfigurable FIR Filter Design Using Dynamic Partial Reconfguration", *School of Information and Communication Eng.*, Inha Univ., Incheon, Korea.
[14] Kuan-Hung Chen and Tzi-Dar Chiueh, "A Low-Power Digit-Based Reconfigurable FIR Filter", *IEEE Transactions on Circuits and Systems—II*: Express Briefs, vol. 53, no. 8, August 2006.
[15] F. Vahad, "Register-Transfer Level (RTL) Design", *Digital Design with RTL Design, Verilog and VHDL*, USA, 2010, ch.r, sec 5.3.