Design and implementation of a reconfigurable finite impulse response filter for adaptive systems

R. Saranya*

Department of Electronics and Communication Engineering,
NSS College of Engineering,
Palakkad, India
Email: saranya181190@gmail.com
*Corresponding author

C. Pradeep

Department of Electronics and Communication Engineering,
Mar Baselios College of Engineering and Technology,
Peermade, Idukki, India
Email: pradeepcee@gmail.com

R. Radhakrishnan

Department of Electronics and Communication Engineering,
Sasurie College of Engineering,
Vijayamangalam, Thirupur, India
Email: rlgs14466@rediffmail.com

Abstract: In this paper, we present the design and implementation of a reconfigurable finite impulse response (FIR) filter for adaptive systems with online fault detection mechanism. An area efficient data path with online fault detection mechanism depending on nature of operands is used to model the FIR filter and is based on the concept of divide and conquers approach. An online fault detection mechanism is introduced by utilising the feature of duplication with comparison where the same calculation is performed twice and the outputs are compared to identify errors. The design is modelled using Verilog HDL, simulated and synthesised using Xilinx ISE 14.2. The design is also modelled using Leonardo spectrum to show the area efficiency of the proposed data path. The design is evaluated using PlanAhead 14.2 on ML 505 development board with Virtex 5 (XC5VLX110T-1FF1136) FPGA which supports partial reconfiguration.

Keywords: dynamic partial reconfiguration; DPR; field programmable gate array; FPGA; FIR filter; data path; fault detection; PlanAhead.


Biographical notes: R. Saranya received her BTech in Electronics and Communication Engineering from the University of Kerala, India. She was a Research Scholar at the FACTS ELCi Lab, SAINTGITS College of Engineering, Kottayam, Kerala. Currently, she is working as an Assistant Professor at the NSS College of Engineering, Palakkad. Her research interests include testing of VLSI circuits, reconfigurable computing and FPGA-based system design.

C. Pradeep graduated in Electronics and Communication Engineering from the University of Madras in 1998. He received his MTech in VLSI Design from the Bharath University, Chennai and PhD in Information and Communication Engineering (VLSI) from the Anna University, Chennai. He is having more than 15 years of professional experience at various capacities of teaching in different engineering colleges, research and industry. He is currently working as the Principal of Mar Baselios College of Engineering and Technology, Peermade, India. His research interests are in online testing of arithmetic circuits, FPGA-based system design, reconfigurable systems, etc.
1 Introduction

Reconfiguration is the ability to modify the logic functions implemented in hardware without changing the system’s design. Compared to an application specific integrated circuit (ASIC), the logic structures of a field programmable gate array (FPGA) are lot more regular and flexible. FPGAs have reconfigurable computing technology which is still a new field of study. Dynamic partial reconfiguration (DPR) is even more challenging. It allows changing a part of the device while the surrounding logic continues to operate during that change. The field of reconfigurable computing has been enthused by the significant changes in the application space of DSP. Reconfigurable computing for DSP remains as a dynamic area of research as it necessitates the integration of more traditional DSP technologies. DSP application requires fast computations and flexibility of the design. Partial reconfiguration is a sophisticated technique, which improves the flexibility of FPGAs by permitting parts of a design to be reconfigured dynamically by overwriting parts of the configuration memory.

An finite impulse response (FIR) filter is a special kind of digital filter that can be employed for an extensive range of filtering goals. The disclosure of challenging applications in digital signal processing (DSP) makes it vital to design reconfigurable architectures. Traditional design flows do not support reconfiguration at all. When it comes to combined partial and dynamic reconfiguration, the situation is even worse since there is a lot more to take into consideration while designing a system. There are few FPGAs on the market supporting DPR. Two basic styles of DPR are difference and module-based partial reconfiguration. The former is used when a small change is made to the design while the latter uses modular design concepts to reconfigure large blocks of logic.

In this paper, we present the design and implementation of a reconfigurable FIR filter by using an efficient logic fault detection mechanism (Pradeep et al., 2014). In order to show the area efficiency of the proposed data path, filters are also modelled using array multiplier and ripple carry adder (RCA), Wallace tree multiplier and RCA. The basic idea of an FIR filter is simple: the present output is obtained by multiplying the present 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. Separate multiplier and adder blocks are designed to model the FIR filter using the concept of divide and conquer approach (Purohit et al., 2013). Moreover, there is an added concern for making the system fault tolerant. Hence, an online fault detection mechanism (Pradeep et al., 2014) is incorporated by exploiting the concept of dual modular redundancy (DMR) utilised in the data path. The introduction of the fault detection mechanism does not result in any increased area overhead. Online fault detection plays a significant role in systems dealing with critical application environments. A well-known technique to attain fault detection is duplication with comparison where the same calculation is performed twice and the outputs are compared to identify errors. Even if same calculation is performed twice, the delay of the proposed data path is less (Pradeep et al., 2014). In order to detect a fault, 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 (Pradeep et al., 2014).

However, there are certain drawbacks related with dynamic reconfigurable design of FIR filters with traditional FPGA techniques. One of the main challenges is the configuration time. This relies on the reconfigurable device and the kind of reconfiguration. Partial reconfiguration can be employed in this case. Partial reconfiguration takes this flexibility one step further, by reconfiguring selected areas of an FPGA any time after its initial configuration (Wang and Wu, 2009). Partial reconfiguration provides less configuration time, coefficient flexibility and area efficiency for higher order FIR filters. The proposed FIR filter is made dynamically reconfigurable by using PlanAhead 14.2. PlanAhead defines a reconfigurable partition and used to define the reconfigurable area and reconfigurable modules. Here, the FIR filter is made partially reconfigurable by using a 2-tap and two 3-tap filters as reconfigurable modules. The design is evaluated using PlanAhead 14.2 on ML 505 development board with Virtex 5 (XC5VLX110T-1FF1136) FPGA which supports partial reconfiguration.

The rest of the paper is organised as follows. Section 2 gives a brief description of related works in this area. Section 3 deals with the architecture of the proposed design. Starting from the basic notions of reconfiguration, the chapter gives a precise description on the design of reconfigurable FIR filter and on the architecture of the data path (Pradeep et al., 2014). Section 4 presents the synthesis and evaluation results. Finally, the concluding remark and future work is presented in Section 5.
2 Related works

The modern advances in reconfigurable computing are mostly derived from the technologies developed for FPGAs in 1980s. The flexibility, capacity and performance of this device have opened up entirely new concepts in high performance computing, forming the basis of reconfigurable computing. Even general purpose processors use the idea of reconfigurability by reusing computational components for independent computations and by using multiplexers to control the routing between these components.

A new coarse grained reconfigurable architecture (CGRA) MATRIX is proposed and it comprises a collection of identical basic functional units (BFUs) (Mirskey and DeHon, 1996). Every BFU contains an 8-bit ALU, 256 words of 8-bit memory and control logic. Reconfigurable architecture workstation (RAW) (Taylor et al., 2004) is another reconfigurable architecture and the proposal of RAW is to offer simple, extremely parallel computing architecture. Lin (2001) proposed a dynamically reconfigurable processor for calculating inner products. He also described the concept of reconfiguration by considering the multiplication of two 8-bit numbers using four $4 \times 4$ multipliers and that of two 16 bit numbers using 16 $4 \times 4$ multipliers. Wang and Wu (2009) published a paper on DPR in FPGAs and illustrated the advantages of early access partial reconfiguration.

The concept of extendable and reusable arithmetic units have been proposed by several groups since it affects the performance of the whole system. An area and time efficient reconfigurable arithmetic unit (Xydis et al., 2009), a low power low cost computational model for multimedia applications (Xiang et al., 2006), organisation and FPGA implementation of MORA processor core (Chalamalasetti et al., 2009) are proposed. Performance comparison of different previous reconfigurable data paths is illustrated and two new processing elements are proposed and its evaluation is done in Purohit et al. (2008). However, the disadvantages of previous approaches were overcome in the data path proposed in Purohit et al. (2013). 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. The architecture of the data path proposed in Purohit et al. (2013) is shown in Figure 1.

A reconfigurable FIR filter design using DPR is presented, which has area efficiency and flexibility allowing dynamical insertion and removal of partial modules using Xilinx Virtex-2 XC2v6000 FPGA (Oh et al., 2006). A digit-reconfigurable FIR filter architecture with fine granularity is presented and implemented using CMOS technology (Chen and Chiueh, 2006).

![Generalised architecture of the data path](Source: Purohit et al. (2013))
The implementation of modular systems having the capability of adding and withdrawing modules dynamically on FPGAs is presented (Sedcole et al., 2006). Modular reconfiguration is not agreeable to the architecture of Virtex and Spartan series FPGAs. However, DPR is possible in these devices. McDonald (2008) states that DPR can minimise the area and power utilisation. The partial bitstream rearrangement activity is proposed (Corbetta et al., 2009). A part of the system capable to renew the bitstream information can be used for the dynamic relocation of a reconfigurable module.

3 Proposed work

The proposed work deals with the design and implementation of a reconfigurable FIR filter by using an area efficient data path with an online fault detection mechanism (Pradeep et al., 2014). FIR filters has wide applicability but it may need a large number of coefficients to obtain the desired specification.

A reconfigurable FIR filter is proposed which can function as 2-tap, 3-tap, 5-tap and 8-tap FIR filters. Module-based partial reconfiguration is used. Here, a 2-tap and two 3-tap filters are the partial reconfigurable modules. Figure 2 shows the proposed reconfigurable FIR filter. The reconfigurable FIR filter is designed with a top module that has four 8-bit inputs and a 19 bit output. The output is set to 19 bit to suite the different tap filters. The reconfigurable module is instantiated with three instances: a 2-tap and two 3-tap FIR filters. This is the top static module.

The top module and the reconfigurable modules are synthesised separately to obtain the netlist. Appropriate configurations are implemented in each design to generate the full and partial bit streams for that configuration. For example, a 5-tap FIR filter is implemented by making two reconfigurable modules as 2-tap and 3-tap and the third reconfigurable module is made as black-box whereas an 8-tap FIR filter is implemented by making three reconfigurable modules as 2-tap, 3-tap and 3-tap. The different order filters can be implemented in less reconfiguration time by downloading the appropriate bit files.

3.1 FIR filter design

FIR filter computes the output by multiplying an input sample with a set of coefficients followed by addition. So, in order to model an FIR filter we require multiplier and adder blocks. Here, the multiplication and addition processes are based on the concept of divide and conquer (Purohit et al., 2013) approach. Separate multiplier and adder blocks are designed to model both serial and pipelined FIR filters.

FIR filter is a special kind of digital filter that takes a stream of digital inputs and generates a stream of digital outputs with some feature of the input stream modified. Mathematically, an N-tap FIR filter can be described as follows:

\[ y(t) = \sum_{i=0}^{N} c_i k(t-i) \]
Here, ‘t’ is the present time step, ‘k’ is the input signal, ‘c_i’ denote the filter coefficients and ‘y’ is the output signal. Each term in the expansion of the above equation is called a tap. The block diagram of an N-tap FIR filter is shown in Figure 3. It consists of registers, multipliers and adders. Registers ‘r0’, ‘r1’, ‘...’, ‘rn’ are needed for each tap to hold k(t), k(t – 1), ..., k(t – N), respectively. The data moves to the right on each clock cycle, so that the register ‘r0’ holds the current input sample, ‘r1’ holds the previous input sample and so on. 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.

Here, 2-tap and 3-tap FIR filters are designed and implemented. An FIR filter with two coefficients is known as a 2-tap FIR filter and that having three coefficients is known as a 3-tap FIR filter. They are described as follows:

\[ y(t) = c_0 k(t) + c_1 k(t-1) \]

\[ y(t) = c_0 k(t) + c_1 k(t-1) + c_2 k(t-2) \]

The above two equations are implemented by using registers, multipliers and adders. Multiplier blocks are implemented by using the concept of divide and conquer approach. Addition process is also implemented similarly. Wallace tree multiplier is used for both processes.

### 3.2 Data path architecture

The architecture of the data path is shown in Figure 4. The data path can perform N-bit multiplication and an addition operation as well as an online logic fault detection mechanism (Pradeep et al., 2014) is incorporated into the data path. The data paths consist of two N × N/2 Wallace tree multipliers, compressors, adders and carry completion logics. It also contains equality comparators and 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.

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

In the data path two N × N/2 multipliers are used. DMR uses two equivalent functional units and provides fault detection when the units that should give the same results give different results (Pradeep et al., 2014). If both the inputs of the equality comparator are same and the output 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 output 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 done. This is an online fault detection mechanism since it does not require the system to shut down to detect the fault.

Carry completion detection logic is used to recognise the completion of multiplication operation. It is essential since the fault detection is done after the multiplication operation. Figure 5 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. Figure 6 shows the CT unit. 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 to generate carry completion signal. 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 5  Carry completion logic

Figure 6  CT unit
In order to detect a fault, fault needs to be introduced into the circuit. This can be done by converting the native circuit description (NCD) file of the fault free circuit which is the output of place and route process into Xilinx design language (XDL) by using a tool – XDL and the XDL file has to be edited to introduce the fault.

4 Result and analysis

In order to evaluate the feasibility of both 2-tap and 3-tap FIR filters, it is modelled using Verilog HDL and synthesised using Xilinx ISE 14.2. The input and coefficients of 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. In order to show that the proposed data path is area efficient, filters are also modelled using array multiplier and RCA, Wallace tree multiplier and RCA. The filter is made reconfigurable by using PlanAhead 14.2 and implemented on ML 505 development board with Virtex 5 (XC5VLX110T-1FF1136) FPGA which supports partial reconfiguration.

4.1 Experimental result using Xilinx ISE 14.2

First, the functional simulation of both 2-tap and 3-tap FIR filters is done by using ISim with exhaustive test bench. Fault injection is done after the functional simulation process and is done by using the ncd2xdl and xdl2ncd conversion feature of Xilinx. Then, the post route simulation is again performed to verify the fault detection mechanism with appropriate test vectors. Successful simulation is then followed by the synthesis process with Xilinx Virtex-5 FPGA to obtain the maximum combinational path delay and device utilisation for FPGA implementation. Table 1 gives the device utilisation summary and maximum combinational path delay of 2-tap and 3-tap FIR filters when implemented on Virtex-5 (XC5VLX110T) FPGA. It shows the utilisation of slices, LUTs, IOBs, etc.

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Slice occupied (%)</th>
<th>LUTs (%)</th>
<th>IOBs (%)</th>
<th>Maximum combinational path delay (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-tap FIR filter</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>12.771</td>
</tr>
<tr>
<td>3-tap FIR filter</td>
<td>1</td>
<td>1</td>
<td>8</td>
<td>14.763</td>
</tr>
</tbody>
</table>

4.2 Experimental result using Leonardo spectrum

Table 2 shows the area of serial FIR filters when modelled using different multiplier and adder types.

Area is usually determined in terms of number of gates. It can be seen from the result that the filter modelled using the proposed data path has less number of gates, that is, 5,281 gates as against the area utilised by the FIR filter when modelled using array multiplier and RCA, Wallace tree multiplier and RCA, proving that the data path is area efficient. The area utilised by FIR filter when modelled using different multiplier and adder structures is shown in Figure 7.

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Multiplier and adder type</th>
<th>No. of gates</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIR filter</td>
<td>Array multiplier and RCA</td>
<td>5,742</td>
</tr>
<tr>
<td></td>
<td>Wallace tree multiplier and RCA</td>
<td>5,641</td>
</tr>
<tr>
<td></td>
<td>Proposed data path</td>
<td>5,281</td>
</tr>
</tbody>
</table>

4.3 Implementation result using PlanAhead 14.2

The FIR filter is made partially reconfigurable by using PlanAhead 14.2. The proposed reconfigurable FIR filter can function as 2-tap, 3-tap, 5-tap and 8-tap FIR filters. The top module is designed with respect to the proposed design. The netlist files generated in conformance with the PR
guidelines after the synthesis of the Verilog HDL code and the assigned user constraint file (UCF) is loaded to evaluate the proposed module. The three reconfigurable partitions in the design are then defined and black boxes are added to each. Each reconfigurable partition has one reconfigurable module. They are then added by loading the netlist files of a 2-tap and two 3-tap FIR filters. The reconfigurable partition region is then defined.

The first configuration is then created as 2-tap FIR filters. Here, two 3-tap reconfigurable modules are assigned as black box and the other module take the functionality, i.e., it act as 2-tap. The first configuration is then implemented and three more configurations are created as 3-tap, 5-tap and 8-tap filters. In case of a 3-tap filter, the 2-tap and one of the 3-tap modules are assigned with black box and the other module as 3-tap. In case of a 5-tap filter, the 2-tap and one of the 3-tap modules are assigned with their functionality and the other module as black box and in case of an 8-tap filter, the three modules are assigned with their respective functionalities. The three configurations are then implemented. The configurations are verified to ensure that the static implementation is constant across all configurations. The full and partial bitstreams are then generated for the various configurations.

The functionality is verified by downloading the bit streams to the Virtex-5 LXT ML 505 development board. The bit streams are downloaded by using joint test access group (JTAG) configuration cable and iMPACT tool. Figure 8 shows the experimental test set up. Initially the full bit stream is downloaded to initiate the device, then, the partial bit streams are downloaded.

Table 3 gives the slice logic and IO utilisation of each reconfigurable module. Table 4 gives the device utilisation summary and maximum combinational path delay of the top module. The definition of reconfigurable partition regions is shown in Figure 9. Table 5 gives the netlist estimation of top module.

5 Conclusions and future work
Partial reconfiguration allows the designers to decrease the size of the design by time multiplexing the hardware resources. Earlier, in order to change a design it requires new placement and routing of the design and also to shut down the system while making the changes. But by using partial reconfiguration, the designer needs only to place and route the modified design and new configurations can be dynamically inserted while the system is running.

A reconfigurable FIR filter was designed and implemented on Xilinx Virtex-5 XC5VLX110T-1FF1136 FPGA. The design was modelled in Verilog HDL. It was simulated and synthesised using Xilinx ISE 14.2. The proposed FIR filter was made dynamically reconfigurable by using PlanAhead 14.2. The reconfigurable modules were modelled in Verilog HDL and implemented on Virtex-5 FPGA which supports partial reconfiguration. Partial reconfiguration saves the silicon area by permitting several configurations to be exchanged in or out of the device and
give flexibility by substituting one configuration by the other. The different order filters were implemented in less reconfiguration time by downloading the appropriate bit streams. In mobile communication systems, multimedia applications as well as in space applications, the need for reconfigurable FIR filters has been increasing tremendously because of the advantage of less area, low cost and high speed of operation.

This proposed design can also be designed by difference based partial reconfiguration flow by identifying only the difference between the different FIR order filters and generating particular partial bit streams and can be reconfigured.

FPGAs reconfigurable computing technology is still a new field of study. FPGA usage is steadily increasing and is substituting ASICs on a standard basis. Reliability of electronics components is the highest requirement and the consequences of reliability problem are severe. Faults cannot be avoided in electronic systems despite the care we are taking in designing and building it. Faults in electronic devices leads to degradation in the consistency of the overall system. The primary aim is to obtain VLSI circuits that are capable of self-repair. The original device is partly reconstructed during the self-repair. It improves the reliability of electronic systems. Fault detection modules are required in self repairing systems. The future goal is to make this FIR filter self-repairable. In self-repairing systems, cells without any function are initially implemented as logic blocks or routing resources and are programmed later in case of failure to replace faulty cells. By developing a self-repairing algorithm, it helps in sustaining the system for a longer duration rather than shutting down the system completely. The applications of self-repair include mission critical digital systems, avionics, medical electronics, space probes, etc.

References


