Novel Implementation of FIR filters using Sequential Circuits

Rishitej Chaparala
Department of ECE,
Sasi Institute of Technology &
Engineering, Tadepalligudem.

Naga Raju Ravada
Assistant Professor,
Department of ECE,
Sasi Institute of Technology &
Engineering, Tadepalligudem.

Sudheer Kanuri
Assistant Professor,
Department of ECE,
Sasi Institute of Technology &
Engineering, Tadepalligudem.

Abstract:

The asynchronous paradigm has interesting features due to the lack of the clock signal and it is another option for the project of digital systems. This paradigm has several design styles, where the micro pipeline style is the most suitable one for FPGA platforms, due to the simplicity of its control. In this paper, we propose a five stage data path pipeline architecture that implements both array and booth multipliers, considering FPGAs as target devices. Through a case study, we show that the booth multiplier presents 14.99ns reduction of timing to that of array multiplier.

Keywords:

Micropipeline, multipliers;

I. INTRODUCTION:

FPGAs devices (Field Programmable Gate Array) have become a very popular way to develop and implement digital circuits due to their cost and design time. High performance FPGAs are implemented in Deep-Sub-Micron MOS technology (DSM-MOS). In this technology, the delay on the lines should be considered and may be higher than the delay of the logic gates [1], [2]. Digital systems are traditionally designed in synchronous paradigm, i.e. they use a global clock signal to synchronize their operations. They are quite popular due to their simplicity of design and availability of commercial CAD tools for automatic synthesis. In DSM-MOS technology, a clock signal requires attention due to its noise generation, electromagnetic interference and power consumption. Besides these factors, the distribution of the clock signal along the chip is a task with increasing complexity because of the clock skew problem, which drops the system performance. The overhead caused by the clock signal can reach 130% in a VLSI (Very Large Scale Integration) implementation [3] and worsens when FPGAs are employed.

The pipeline architecture known as MOUSETRAP, proposed in [11], has a good performance, it is based on logic gates and can be implemented on FPGAs (see Fig. 1). In this paper, we propose a FPGAs oriented architecture for micro pipeline design style (see Fig. 2). The proposed linear pipeline architecture employs flip-flops as registers, due to the large availability of these elements in FPGAs. The control is based on logic gates and it is mapped into two LUTs (Look-Up Table). The new pipeline architecture has a better performance, when compared to the MOUSETRAP architecture implemented on FPGA, because it has a smaller latency and a greater throughput. A digital FIR filter of fourth order shows the efficiency of our architecture on a FPGA platform.
II. MICROPIPELINE PROJECT EMPLOYING FPGA:

Programmable devices, such as FPGAs, are designed for synchronous designs [2]. Efforts for prototyping asynchronous design in commercial FPGAs [14], [15] and academic FPGAs [16], [17] have been reported recently. The problems in implementing asynchronous systems in commercial FPGAs are related to the control. They area) Process of mapping risk-free Boolean functions in logic blocks (macro cells).

The commercial tools used for de-composition and mapping Boolean functions in LUTs are not prepared to meet the requirements of logical hazard. This may cause a circuit malfunction, if manual intervention to fix the problem is not performed. The mapping function must satisfy the decomposition requirements proposed in Sigel et al. [18]. b) Internal routing process among macro cells can introduce significant delays. These delays can result in essential hazard and lead to the circuit malfunction [4], [5].

The circuit delay model defines how to solve the problem of essential hazard: the insertion of delay elements in the feedback lines or the employment of macro cells that satisfy the isochoric fork condition [4], [5]. The micro pipeline architecture can be linear or nonlinear. [19]. In this paper we focus only on the linear architecture. The micro pipeline style has as main feature, the simplification of the pipeline control, an important characteristic for asynchronous systems implemented in FPGAs.

The control is either distributed between stages or centralized, and it is responsible for the communication between the pipeline stages. In a FPGA platform, the control must be distributed in order to avoid hazard problems, as previously mentioned. The pipeline communication employs the handshake protocol with Request and Acknowledge signals[4],[5]. The communication between the stages can be performed in two different protocols: 4-phase or 2-phase. Fig. 3 shows the behavior of these protocols.

(a) 4 phases handshaking control

(b) 2 –phases

The linear micro pipeline design has two variants. In the first one, it is used only components of the synchronous paradigm (single-rail) and delay elements between stages. The delay is defined considering the critical path of each stage. The bundled-data implementation is one of the data encoding schemes used in asynchronous circuits. It represents N bits of data with N+2 lines, called “bundled”, where the two additional lines are the handshake signals, request and acknowledge. This architecture is called “micropipelinebundled-data”. The second architecture employs dual rail components [4], [5]. The components are synthesized using Delay Insensitive(DI) codes or dual-rail. In the case of dual-rail code, each Fig. 6. Controller’s logic circuit. signal is encoded with two bits. Four cases would be possible for a signal: a0a1=00 (null); a0a1=01 (1); a0a1=10 (0); and a0a1=11 (never occurs).

The design employing this code generates a signal of “end of operation” without the need of delay element, and it can be implemented with a simple circuit. The dual-rail linear micro pipeline architecture can be implemented in FPGA platforms, but there is a significant increase in the number of used macro cells, that compromises their performance and power consumption. On the other hand, some advantages need to be considered: this architecture does not need delay elements and timing analysis, moreover, it is more robust.

The proposed micro pipeline architecture operates in 2-phase handshake protocol and, therefore, the processing request occurs on both edges of this signal. Fig. 4 shows the general lay-out of a stage, which consists of a flip-flop based register and control. The control was specified in STG (signal transition graph) [20] (see Fig. 5). It is composed by the signals: Ri (request input), Ao (acknowledge output), Ai (acknowledge input), L (load) and Ro (output request). Fig. 6 shows the circuit of the logic control.
The procedure used to synthesize bundled-data micro pipeline systems attempts to use components and synthesis tools of the synchronous paradigm. The behavioral synthesis starts from the Control Data Flow Graph (CDFG), which represents the operations and their data dependency. The method can be divided into five steps: 1) Generation of the scheduled DFG employing the list scheduling algorithm [21]. 2) Based on the scheduled DFG, generation of the pipeline data-path employing single-rail components and the pipeline synthesis according to [21]. This step generates the synchronous pipeline. (see Fig. 7). 3) Execution of the desynchronization process; detection of the critical path of each pipeline step and calculation of the respective delay element.

4) Change the clock signal by the proposed asynchronous pipeline controller (see Fig. 8). 5) Synthesize the asynchronous pipeline generated in step 4.

In order to illustrate the use of proposed micro pipeline architecture, an asynchronous version of 4th order FIR filter was designed, according to the equation:

\[ y[n] = \sum_{i=0}^{\infty} h_i \cdot x[n-i] \]

where, \( y[n] \) and \( x[n] \) are the output and input signal samples, and \( h(i) \) are the FIR filter coefficients. Fig. 9 shows the first step, the scheduled GFD of the filter, obtained from the List Scheduling algorithm with three resource constraints: two multipliers and one adder. The second step of the proposed method generates a data-path pipeline with five stages. It was obtained by the behavioral synthesis [21], which defines the synchronous pipeline. Fig. 10 shows the pipeline data-path of the filter. The steps 3 and 4 make the desynchronization of the pipeline, generating the proposed pipeline, requiring six control sand delay elements.

III. EXPERIMENTAL RESULTS:
A comparison was made with the MOUSETRAP pipeline architecture in order to evaluate the performance of the proposed architecture.
The simulations were performed employing the Xilinx software, version 14.2, considering an Spartan 6 as target device. Fig. 11 and 12 show the simulations of the mousetrap control and the stages of the FIR filter. Fig. 13 and 14 show the simulations of the proposed pipeline control and FIR stages.

### Table 1. ARRAY MULTIPLIER

<table>
<thead>
<tr>
<th>Logic Utilization</th>
<th>Used</th>
<th>Available</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Slices</td>
<td>464</td>
<td>768</td>
<td>60%</td>
</tr>
<tr>
<td>No. of Slice flip-flops</td>
<td>251</td>
<td>1536</td>
<td>16%</td>
</tr>
<tr>
<td>No. of 4-input LUT’s</td>
<td>808</td>
<td>1536</td>
<td>52%</td>
</tr>
<tr>
<td>No. of bonded IOB’s</td>
<td>103</td>
<td>124</td>
<td>83%</td>
</tr>
<tr>
<td>No. of GCLK’s</td>
<td>3</td>
<td>8</td>
<td>37%</td>
</tr>
</tbody>
</table>

### Table 2

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Period</td>
<td>17.486 ns</td>
</tr>
<tr>
<td>i/p arrival time</td>
<td>30.149 ns</td>
</tr>
<tr>
<td>O/p required time</td>
<td>6.141 ns</td>
</tr>
</tbody>
</table>

The waveform depicted in the fig.11. Shows the result of 5 stage scheduled GFD filter section with the placement of array multiplier block. Files are inserting filter method in multiplication process. Because of inserting these methods we can get both filtered and multiplier outputs. Therefore, the output consists of both filtered & multiplier output values. To calculate this output initially multiply x0 value with impulse response values h0,h1,h2,h3,h4 Now, x1 value with h0,h1,h2,h3,h4 and x2 value with h0,h1,h2,h3,h4 and repeat same for x3,x4 values. So, by doing this process we will get the exact output. With this process we will get combined outputs nothing but, Intermediate values(filter) & multiplier values(original).

By placing booth multiplier block the same output result obtains to the 5 stage scheduled GFD filter. The output waveform for booth multiplier block resembles similar causalities to that of array multiplier output waveform result. In this project the scheduled GFD using array multiplier output result is said to be existed system where as, scheduled GFD using booth multiplier block result is the extension system.

### Table 3 BOOTH MULTIPLIER

<table>
<thead>
<tr>
<th>Logic Utilization</th>
<th>Used</th>
<th>Available</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Slices</td>
<td>416</td>
<td>768</td>
<td>54%</td>
</tr>
<tr>
<td>No. of Slice flip-flops</td>
<td>241</td>
<td>1536</td>
<td>15%</td>
</tr>
<tr>
<td>No. of 4-input LUT’s</td>
<td>749</td>
<td>1536</td>
<td>48%</td>
</tr>
<tr>
<td>No. of bonded IOB’s</td>
<td>99</td>
<td>124</td>
<td>79%</td>
</tr>
<tr>
<td>No. of GCLK’s</td>
<td>3</td>
<td>8</td>
<td>37%</td>
</tr>
</tbody>
</table>
The simulations were performed employing the Xilinx software, version 14.2, considering an Spartan 6 as target device. Fig. 11 and 12 show the simulations of the mousetrap control and the stages of the FIR filter. Fig. 13 and 14 show the simulations of the proposed pipeline control and FIR stages.

**Table.1. ARRAY MULTIPLIER**

<table>
<thead>
<tr>
<th>Area report:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Period</td>
</tr>
<tr>
<td>i/p arrival time(before clock)</td>
</tr>
<tr>
<td>O/p required time(after clock)</td>
</tr>
</tbody>
</table>

The tables 1&2 depicted above shows the hardware utilization summary of 5 stage scheduled GFD of the filter for both array & booth multipliers. The hardware utilization summary for booth multiplier reduces to that of using array multiplier.

The complexity also reduces for booth multiplier schematic compared to that of array multiplier. The tables 1.1&2.1 depicted above shows the delay report of 5 stage scheduled GFD of the filter for both array & booth multipliers. Compared to the operation of array multiplier the booth multiplier operation display less period than that of array multiplier. Therefore efficiency increases.

**IV. CONCLUSION:**

The linear micro pipeline style of the asynchronous paradigm is advantageous because the performance is increased and the control is simplified. In this paper, we proposed new pipeline architecture oriented for FPGAs implementations. The control employs only two LUTs, and the registers used are based on flip-flops, allowing a better distribution of macro cells in a FPGA. The implementation of the proposed FPGA control is free of essential hazard. Through a case study, we show that our architecture has a better performance than the MOUSETRAP architecture, considering the FPGA implementation.

**REFERENCES:**


**Author’s Profile:**

**Rishitej Chaparala** pursuing his M.Tech in Very Large Scale Integration from Sasi Institute of Technology and Engineering, West Godavari District, Tadepalligudem, Andhra Pradesh 534101.

**Naga Raju Ravada** Working as a Assistant Professor, Electronics and communication Engineering in Sasi Institute of Technology and Engineering, West Godavari District, Tadepalligudem, Andhra Pradesh 534101.

**Sudheer Kanuri** Working as a Assistant Professor, from Electronics and communication Engineering in Sasi Institute of Technology and Engineering, West Godavari District, Tadepalligudem, Andhra Pradesh 534101. Sudheerkanuri@sasi.ac.in