Power Consumption in Novel Digital Filters

Ian Wells and S. Thomas†
ian.wells@smu.ac.uk

Abstract

This paper describes an investigation into the power consumption of a Finite Impulse Response (FIR) filter, designed using primitive operator arithmetic (POF). The filter was compared with a standard FIR filter created using conventionally structured multiplier blocks. The filters were created using Matlab and VHDL coding techniques. In order to maintain control of the structural implementations, both filters were created using only structural modelling methods. Power consumption estimations were made using Xilinx’s XPower software which indicated a possible 34% saving in dynamic power consumption, and a possible 2.5% saving in total power consumption in favour of the POF filter structure.

Keywords
Primitive Operator Filter, Multiplier Block, Multiplierless Filter, Low Power Filter.

†Swansea Metropolitan, University of Wales: Trinity Saint David
Introduction

To reduce the size of digital filter architectures, the use of primitive arithmetic operators such as additions, subtractions and bit-shifts can be used to replace banks of multipliers. Known as POF or Multiplierless filters, these techniques have been used for many years as an efficient way of constructing digital filters [1, 2, 3, 4, 5].

To design an FIR filter using the POF technique, it necessary to use the basic a transposed direct form structure as shown in figure 1.

The multipliers in this format can be treated as a single block of combinatorial logic with a simultaneously applied common input. Using of one of the add shift algorithms for FIR filter design will allow the replacement of all the multipliers within the filter; meaning the coefficients are effectively hard coded into the hardware [6, 7].

The historical motivation for this process was to reduce the silicon area needed to implement a digital filter with significant area reductions being achieved with some of the proposed algorithms [6, 7].

More recent work however has concentrated on another aspect of POF filters which is the possibility of reduced power consumption through the use of such algorithms [8, 9, 10, 11, 12, 13].

The basic aim of this paper is to show that the use of the add shift algorithm detailed later, will allow the synthesis of lower power FIR filters on FPGAs as compared to filters using the conventionally structures multiplier units [6, 7].
While the algorithms are not optimized for FPGA implementation this initial work will seek to verify if the POF method merits further investigation as a low power algorithm for use on FPGAs.

**POF Design Algorithm**

**Algorithm Detail**

The algorithm that was adopted for this work is reiterated below [6]. Before proceeding however, the filter coefficients must be quantized to the required bit size (B). The $Q$ format techniques has been adopted where the number to be quantized is first multiplied by $2^n$ where,

$$ n = B - 1 $$

(1)

The resulting number is then rounded to the nearest integer value.

The algorithm can consequently be applied as below:

1. Arrange all the filter coefficients in ascending order.
2. Discard duplicate values.
3. Produce a chain of power of two values, by left shifting the input signal, so that the largest value is less than or equal to the largest coefficient. i.e. if the largest coefficient was 136 then the chain of power of two values would be 1,2,4,8,16,32,64,128,...
4. Add all values formed to an available coefficient list. These will be values that can be used to form coefficient product terms if required. If any of these values is equal to a coefficient then this value will be added to a formed coefficient list. These are the coefficient values of the filter bank itself.
5. Next take the smallest unformed coefficient from the list of required filter bank coefficients. Attempt to create this coefficient by adding two numbers that have been subjected to the least addition operations. This is done since only adders represent circuitry in this scheme and so these operations should be minimized to minimize the size of the circuit.
6. If this number has been successfully created then add it to the formed coefficient list.
7. Left shift the coefficient until it less than or equal to the largest coefficient. Add all the newly created values to the available coefficient list.

8. Search available coefficient list to see if a coefficient was created as a result of (step 7). Repeat stages (step 6) and (step 7) if a coefficient was created.

9. Repeat stages from (step 5) until all coefficients have been considered. This will produce the necessary data for the creation of a POF filter.

**Example Implementation**

The following simple example illustrates this process:

Consider a filter function having the Q-Format coefficients 2, 5, 9, 5, 2. To construct a POF filter we can remove any duplicate coefficients and arrange in order (5 & 9), then construct the filter as shown in figure 2, where each coefficient can be generated from a previous coefficient.

![Figure 2: Example POF filter](image)

**Method and Results**
POF Implementation

The algorithm used [6] requires that the filter coefficients are first quantized and represented as a Q-Format number. For example to achieve 10 bit quantization (Q9 format) the coefficients should be multiplied by $2^9$. In the case of the POF filter implemented for this work the following process was used:

1. Produce the sequence $n$ by arranging the coefficients (Q9 format) in ascending order, treating all coefficients as positives, even if they were generated as negatives (negative numbers can easily be dealt with later). Remove all zeros and duplicated coefficients.

   $$n = \{3 \ 8 \ 14 \ 15 \ 17 \ 19 \ 37 \ 56 \ 71 \ 81 \ 85\}$$

2. Currently, no coefficient values are available. Produce a series of numbers from 1 by left shifting 1 repeatedly (multiplying by 2) until the value of the highest coefficient is exceeded (do not include values larger than the highest coefficient). List any coefficients that are derived as a result of this step.

   $$\{1 \ 2 \ 4 \ 8 \ 16 \ 32 \ 64\}$$, $$\{8\}$$

3. Use these numbers to make as many of the coefficients not presently derived by step 2, starting with the lowest coefficient.

   $$\{3=1+2\}$$, $$\{17=1+16\}$$

4. Repeat step 2 on the coefficients derived in step 3. List any coefficients that are derived as a result of this step.

   $$\{3 \ 6 \ 12 \ 24 \ 48\}$$, $$\{17 \ 34 \ 68\}$$

5. Repeat step 3 on the lists derived in steps 2 and 4 (the available coefficients thus far).

   $$\{14=2+12\}$$, $$\{15=12+3\}$$, $$\{19=17+2\}$$, $$\{37=34+3\}$$, $$\{56=32+24\}$$, $$\{81=64+17\}$$, $$\{85=68+17\}$$

6. Repeat step 2 on the coefficients derived in step 5. The remaining coefficient (71) can be found by adding a number of the coefficients now available.

   $$\{71=56+15\}$$

A block diagram (figure 3), can now be drawn utilizing left shifts and additions to visually represent the complete multiplier block, which when constructed in VHDL will connect to the summation/delay line to complete the filter.
Filter Synthesis and Testing

In order to see if a primitive operator multiplier block would offer a power saving over a conventional multiplier block, two FIR filters with the same frequency response specifications were designed using Xilinx ISE [14] design suite. The designs were targeted for implementation using a Xilinx XC2VP30 FPGA. Both filters used the transposed direct form to obtain fair comparison.

With no stringent application requirements to adhere to, there was a high degree of flexibility available when selecting the design of the filter. It was therefore decided that the filter would be designed around a set of basic requirements. To test the filter, a suitable composite waveform would be generated that best suits the filters frequency response curve.

The Matlab [15] Filter Realization Wizard was used to generate the filter coefficients. A low pass filter was selected for the trials. The chosen design had an 800 Hz cut-off and a 10 kHz sample rate. A 20-order Gaussian Window was
chosen and all coefficients were quantized to 10 bits.

Examination of the filters frequency response curve indicated an attenuation of 0 dB at 550 Hz, and -37 dB at 1.1 kHz. It was therefore decided that the ideal input signal would consist of these two frequency components.

The filters frequency and phase responses are illustrated in figures 4 and 5 respectively.

The coefficients were converted to Q9 format prior to the implementation of the algorithm. The 21 filter coefficients values that describe both filters, after undergoing the quantization process are illustrated below:

\[ n = \{-14, -17, -15, -8, 3, 19, 37, 56, 71, 81, 85, 81, 71, 56, 37, 19, 3, -8, -15, -17, -14\} \]

Both filters were coded using VHDL. Structural modelling methods were strictly adhered to during the coding of all components, with the aim of maintaining control of the architectural implementation of both filters.
As each component was generated, ModelSim [16] simulation software was used to check their functionality. At the highest hierarchical level, both designs were tested by applying a composite waveform to their input. The waveform comprised of the two frequency components stated previously. The results of the simulation were plotted to a graph as shown in figure 6.

As expected both filter designs produced identical output waveforms.

![Filter Simulation Results](image)

**Figure 6: Filter simulation results**

After successful simulation and synthesis the designs were implemented for use within a Xilinx Virtex™-II Pro XC2VP30 FPGA. The FPGA is the central component within the Xilinx XUP-V2PRO Development Board as shown in figure 7 [17]. Translate and Map reports yielded zero errors or warnings throughout the implementation process for both filters. The Place and Route settings were as follows:

- Overall effort level (-ol): Standard
- Placer effort level (-pl): High
- Placer cost table entry (-t): 1

![XUP-V2PRO Development Board](image)

**Figure 7: XUP-V2PRO Development Board**
Router effort level (-rl): Standard

As with the Translate and Map reports, the Place and Route report yielded zero errors or warnings for both filters.

Following successful implementation of each of the filters, programming files were generated. The FPGA was then configured using the Xilinx’s iMPACT programming tool. Each filter design was configured and checked for functionality in turn.

The required input signal was generated using a counter within a process statement which was included in each design. An external Astable circuit was used to generate the 10 kHz clock source. Each filter was run under identical conditions.

After both filter models had been successfully configured and tested, the .ncd design file for each model was used by the XPower Analyzer software to generate Power analysis results shown in table ??1. The results indicate a saving of 34% in favour of the POF filter design in the dynamic (or switching) power consumption. The total predicted power consumption was reduced to 2.5% due to the high quiescent Power consumption of the device. This translates to a total saving of 2.77 mW.

<table>
<thead>
<tr>
<th>Table 1: XPower Analyzer Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quiescent Power (in mW)</td>
</tr>
<tr>
<td>Conventional Filter</td>
</tr>
<tr>
<td>POF filter</td>
</tr>
<tr>
<td>Power Saving (mW)</td>
</tr>
<tr>
<td>% Saving</td>
</tr>
</tbody>
</table>

Conclusion

Previous studies of POF filter implementations had concentrated on the potentially large savings in silicon area that could be achieved by using POF filter techniques. With the advent of advanced FPGAs such as the Xilinx Virtex
devices however the motivation for using such algorithms has largely dissipated. Recent researchers however have looked again at POF algorithms from the perspective of the potential power savings that they offer.

This work attempted to investigate the validity of one of the simplest POF algorithms with regard to power savings as compared to a standard filter implementation on a Xilinx XC2VP30 FPGA.

The results generated from software based tests indicate power savings can be found when selecting Primitive Operator Filter structures over a traditional multiplier array structured filter.

Although the results for the total Power savings only indicate a small saving, the results for the dynamic power savings are of most significance. These results indicating a saving of over 34% illustrate the true saving available through the use of the POF design method.

It is worth noting that device utilisation figures for both filters are approximately <2%, indicating very little use of the available resources. To this extent the use of a smaller FPGA with lower quiescent power consumption would provide far better total power consumption figures whilst offering a more realistic reflection of real world usage.

In conclusion, this work presents a method for reducing power consumption in digital FIR filters. Whilst a saving of approximately 2.5 mW may seem small, the true saving of 34% in dynamic power consumption indicates the true worth of the use of the algorithm. It is therefore believed that further investigation in to the use POF algorithms with regard to low power FIR filter implementation is valid.

References


References


References


This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution 3.0 Unported License (http://creativecommons.org/licenses/by/3.0/).

©2013 by the Authors. Sponsored and Licensed by HCTL Open, India.
Professor Ian Wells
Obtained the B.Sc. in Physics from Manchester University, Manchester, UK in 1982, the M.Sc. in Microelectronics and Digital Signal Processing from the University of Westminster, London, UK in 1986 and the Ph.D. in Signal Processing Architectures for Speech Recognition from the University of the West of England, Bristol, UK in 1996.

He is currently the Assistant Dean in the Faculty of Applied Design and Engineering at Swansea Metropolitan, University of Wales: Trinity Saint David, Swansea UK where he has worked since 1996. He was previously the Research director of Technium Digital, University of Wales Swansea, Swansea, UK, which is a high technology incubator unit. He also worked as an electronic engineer in AB Electronics, South Wales, UK after completing his first degree. His current research interests include ultrasonic non-destructive testing, medical signal processing, communications systems, sensor networks and computer networks.

Professor Wells is also a Fellow the Institution of Engineering and Technology (FIET), a Fellow of the British Computer Society (MBCS), a Senior Member of the IEEE and a chartered Engineer (C.Eng.).

Stuart Thomas
Obtained his B.Eng. degree (First Class Honors) in Computer Systems and Electronics Engineering from Swansea Metropolitan University in 2009.

He worked as a Technician and Engineer for IVA Co Ltd for thirteen years, primarily in the fields of automation and control system design. His main responsibilities include the design and manufacture of special purpose equipment for the manufacturing industry and general project management.

His current research interests include microelectronics design, digital circuits and systems design, device engineering and semiconductor physics. He has a keen interest in the design of power efficient circuits and working with reconfigurable devices. Presently he is perusing a Ph.D. in semiconductor physics at Imperial College London.