ARQMCM – STUDY ARCHITECTURES FOR SYSTEMS WITH MULTIPLE CONSTANT MULTIPLIERS

Carlos Alexandre Marvão Lavadinho

Instituto Superior Técnico
Lisboa, Portugal

carloslavadinho851@msn.com

ABSTRACT

This work presents a study on different architectures, with multiple constant multipliers, for FIR filters. The filters are automatically generated by a tool G2FIR developed for this work. The tool is able to generate synthesizable VHDL description of FIR filter for three different architecture: serial architecture with accumulator, parallel architecture with multipliers and parallel architecture with multiple constant multiply (MCM).

In order to evaluate the different architecture a set of benchmark filter were synthesized (with design_vision from Synopsis), using the AMS 0.35 µm technology, for a range of input bit witch size (8, 16 and 32 bits). By comparing the area, power and operation frequency (max delay), obtained after synthesis, we conclude that the serial architecture has a reduce area but the parallel architecture with MCM has a better performance researching operating frequency and power dissipated.

Index Terms— FIR architectures, MCM, area, power, frequency.

1. INTRODUCTION

There are different types of digital processing systems, where an input is multiplied by a set of constant coefficients. Systems such as FIR (Finite Impulse Response) filters or FFT’s are usually, performed by Digital Signal Processing (DSP) units or by dedicated hardwired architectures. The hardwired dedicated units can be implemented with different architectures to maximize performance and minimize power consumption and area. One of architectures with particular interest, when multiplications by constants are involved, is the utilization of shifts and additions/subtractions to implement those multiplications [3]. In FIR filters the input signal is multiplied by a serie of constants, all these operations can be implemented using shifts and additions/subtractions in a block usually called as Multiple Constant Multiplications (MCM).

The use of DSP’s for filter implementations has the advantage of allowing the adjustment of the filter parameters or even changes the filter function. However, this general propose signal processing unit has a penalty in the area and power required by the filter when compared to hardware equivalent.

The use of algorithms like MCM [4] to implement the digital multiplication allows to obtain a better performance from the DSP’s, reducing the area and power[5] that would be required if digital multiplication algorithms where directly implemented.

For this study it was developed a software tool that, given a set of filter coefficients, automatically generates different FIR architectures. The different architectures have alternative ways to do multiplications of the input signal by the coefficients. We obtained the results of area, power and frequencies for the different FIR filters architectures, using set of benchmark filters and conclude that parallel architecture with MCM shows the best performance in frequency and power.

2. FIR FILTERS

The FIR filters are characterized by realizing the convolution of the input data samples with the desired unit impulse response of the filter. The output y(n) of FIR filter is obtained by the sum of the latest N input data samples x(n) multiplied by a constant that distributes the weight of the previous samples as shown in Equation (1)[6]:

\[ y(n) = \sum_{i=0}^{N-1} a_i x(n - i) = a_0 x(n) + a_1 x(n - 1) + \ldots + a_{N-1} x(n - N + 1) \] (1)
With the inspection of the equation we can obtain the digital implementation of digital FIR filter. The filter architecture in direct and transported form are illustrated in Fig. 1 and Fig. 2.

In both figures, a(0), a(1), ..., a(N-1) represent the coefficient filters and Z⁻¹ represent the delay.

In this work, three different FIR architectures were considered: a serial architecture with accumulator and multiplier [7]; a parallel architecture with multipliers and a parallel architecture with a MCM implementation [2]. We only considered FIR filters with linear phase because they can be easily designed, by making the coefficient sequence symmetric, and are the most common to be implemented.

3. ARCHITECTURES

In this section we describe, with some example, the architectures that used to implement the FIR filters.

A. Serial Architecture with accumulator and multiplier

This architecture realizes sequentially the operations of the filter. To implement the serial architecture it is needed: a multiplier, an adder, a FIFO, a memory, a control unit and a output register to store the results of partial operation need to complete the output.

The FIFO is a shift register with N (number of coefficients) registers to store input data samples. The filter coefficients are stored in a memory (synchronous with the clock edge). The multiplier receives as operator one value from the FIFO and other from the memory. Since we are only considering linear phase FIR filters we know that all set of coefficients are symmetrical. Therefore, the control unit runs sequentially through the memory with just N/2 position (first up then down). Note that special attention in the design of the control unit is needed when odd number of coefficients are present. An example of filter implemented using a serial architecture and with coefficients 1, 7 and 11 is illustrated in Fig 2.

B. Parallel Architectures

Parallel architectures generate a new value on the filter output on each clock cycle. With the registers values that are between the adders. All the coefficients are multiplied in parallel by the same input, using a fully-parallel implementation in the transposed form for the filter architecture, and then added with as shown in the Fig. 4. We used the transported form, because this filter architecture has less delay (the maximum delay is the series of a multiplier with an adder), and most of the glitching is filtered by the registers, thus reducing de power [7].

This architectures use the same adding and registers unit, implement with N adders/subtractors and registers, independently the way that constants are multiplied by the input signal. However, if a coefficient is zero it is implemented only its register. In the constant multiplication unit constants are always considered in module, thus reducing the number of coefficients, and when adding negative coefficients we will use a subtractor. So, the adders and registers unit (ARU) is responsible to define the correct signal of the coefficients.
B.1 Parallel architecture with Multipliers

The parallel architecture with Multipliers uses common multipliers to multiply all the coefficients by the input signal at the same time. An example of this architecture is shown in Fig. 5, for the coefficients 1, 7 and 11.

B.2 Parallel architecture with MCM

In the parallel architecture with MCM we have replaced the multipliers by a set of additions of shifted versions of the input signal [3] [8]. For the coefficients that are powers of two the multipliers are replaced only by shifted versions of the input signal. The Fig 5 and the Equation 3, show an example of how 7x and 11x can be replace by a set of additions of shifted versions of the input signal.

\[
\begin{align*}
7x &= 3x + 4x \\
&= (2^1x + 2^2x) + 2^2x \\
11x &= 3x + 8x \\
&= (2^1x + 2^3x) + 2^3x
\end{align*}
\]  

When same input is to be multiplied by a set of constant coefficients that can be accomplished by sharing partial terms of the coefficient multiplications, its possible the sharing of resources to realizes the same operations with less area and power. For example 3x is a partial term that is shared in the computation of 7x and 11x. It is possible to use only three adders and shifted versions of the input signal, as illustrated in the Fig. 6 and in the Fig. 6. The Fig. 6 shown an example of parallel architecture with MCM, for the coefficients: 1, 7 and 11.

4. VHDL GENERATOR

The G2FIR is the tool developed to get automatically the VHDL descriptions of the different architectures for the FIR filters. This generator is a program developed, in the C programming language, that receives a file with the characteristics of filter FIR and generates, in VHDL, the description of the filter architecture. The architecture of this generator, G2FIR, as shown in Fig.7.

The generator was developed with the capability to be easily configured with different parameters. The configuration options are selected on the command line, with several switches placed in front of the file name selected, as shown in the example of the Fig 8. Table 1 list all the possible configuration option and their line command switches.

The existing library of VHDL files, describing the common operators (adder, subtractor and multiplier), are used when the generator need to use those operators. It is possible to
use other operators different of the generics operators, with
the last 3 options shown in table 1.

C:\FIR\G2FIR exemplo.fir -t:c

Fig. 8 Example of command line, choose the parallel architecture with
MCM, from the filter in exemplo.fir.

Table 1 with G2FIR options.

<table>
<thead>
<tr>
<th>Options switches</th>
<th>Choose:</th>
</tr>
</thead>
<tbody>
<tr>
<td>-t:a</td>
<td>Serial architecture accumulator and multiplier</td>
</tr>
<tr>
<td>-t:b</td>
<td>Parallel architecture multiplier</td>
</tr>
<tr>
<td>-t:c</td>
<td>Parallel architecture MCM (by omission)</td>
</tr>
<tr>
<td>-b:ni</td>
<td>ni bits of the input signal (16 by omission)</td>
</tr>
<tr>
<td>-o:no</td>
<td>no bits of the output signal (not limited by omission)</td>
</tr>
<tr>
<td>-a:adder_name</td>
<td>Another adder (in the library)</td>
</tr>
<tr>
<td>-s:subtractor_name</td>
<td>Another subtractor (in the library)</td>
</tr>
<tr>
<td>-m:multiplier_name</td>
<td>Another multiplier (in the library)</td>
</tr>
</tbody>
</table>

The Table 2 shows the different names of the files with
the description of the architectures that are automatically
generated by G2FIR, for an input file with the name:
example.filter.

Table 2 with the output files of G2FIR.

<table>
<thead>
<tr>
<th>Architectures</th>
<th>VHDL files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial</td>
<td>example_arq_acum.vhd</td>
</tr>
<tr>
<td></td>
<td>example_memory.vhd</td>
</tr>
<tr>
<td></td>
<td>example_uc.vhd</td>
</tr>
<tr>
<td></td>
<td>example_fifo.vhd</td>
</tr>
<tr>
<td>Parallel with multiplier</td>
<td>example_arq_multiplier.vhd</td>
</tr>
<tr>
<td></td>
<td>example_reg_uni.vhd</td>
</tr>
<tr>
<td></td>
<td>example_arq_mem.vhd</td>
</tr>
<tr>
<td></td>
<td>example_arq_parallel.vhd</td>
</tr>
<tr>
<td></td>
<td>example_arq_mcm.vhd</td>
</tr>
<tr>
<td></td>
<td>example_reg_uni.vhd</td>
</tr>
<tr>
<td></td>
<td>example_arq_mcm.vhd</td>
</tr>
</tbody>
</table>

The files to be read by the program with the description of
the filters coefficients should follow a pre-defined format.
The files have two sections, one showing the filter
coefficients, other indicates the partial terms for the MCM
unit. For the serial architecture with accumulator and
multiplier and for the parallel architecture with multipliers,
this section must exist but it can be empty since it will not
be used.

The first section of the file, starts with the sentence
***Implementation of coefficients***, to
precede the filter coefficients, one per line in the following
order $a_0, a_1, ..., a_{N-1}$. The second section of the file starts
with the sentence ***Implementation of Partial
Terms***, followed by the partial terms, used by MCM
architecture. The order in which sections appear in the file
must be respected. The Fig.9 has shown an example, of the
file, with the information of the coefficients and the partial
terms, with the coefficient equal to 14 and the partial terms
3 and 7.

***Implementation of coefficients***
14 = +7<<1

***Implementation of Partial Terms***
3>>0 = +1<<1+1<<0
7>>0 = +3<<0 +1<<2

Fig. 9 Example of file with the filter information.

5. RESULTS

The FIR filter architectures were generated for nine different
filters with different characteristics. The results of area,
power and frequency were obtained by the synthesis for
input signal with 8, 16 and 32 bits, in this paper only show
the plot of: power and delay to 32 bits. The results of area,
power and frequency were obtained by the synthesis for
input signal with 8, 16 and 32 bits. The synthesis results presented were made using the
tool design_vision, from Synopsys® (version v3.70),
using c35_corelib from AMS. The different
architectures considered were compiled with the tool option
exact_map, the remaining options were left by default.
The clock period required was of 50 ns, this clock
specification is more than enough for all the architectures
and all filters studied.
Table 3 shows the characteristics of the nine filters used as
our benchmark.

Table 3 with characteristics filters.

<table>
<thead>
<tr>
<th>Filters</th>
<th>Coefficients</th>
<th>Diversity Coefficients</th>
<th>MCM operators</th>
<th>Series operators</th>
<th>Bits Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>121</td>
<td>19</td>
<td>10</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>101</td>
<td>24</td>
<td>17</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>16</td>
<td>15</td>
<td>5</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>81</td>
<td>38</td>
<td>28</td>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>121</td>
<td>47</td>
<td>34</td>
<td>5</td>
<td>12</td>
</tr>
<tr>
<td>6</td>
<td>60</td>
<td>22</td>
<td>20</td>
<td>3</td>
<td>14</td>
</tr>
<tr>
<td>7</td>
<td>60</td>
<td>28</td>
<td>29</td>
<td>3</td>
<td>14</td>
</tr>
<tr>
<td>8</td>
<td>60</td>
<td>29</td>
<td>28</td>
<td>3</td>
<td>14</td>
</tr>
<tr>
<td>9</td>
<td>100</td>
<td>49</td>
<td>45</td>
<td>3</td>
<td>16</td>
</tr>
</tbody>
</table>
The result of area were obtained using the Report area (to obtain total area) and Report reference (obtain area of the different units) of design_vision. The area of serial architecture with accumulator and multiplier grows linearly with the number of bits of input signal. For this architecture the memory and control unity have always the same size for each filter, independently of the size of the input signal. The FIFO, the operators and the output register has a linear growth with the number of bits of the input signal. The FIFO is the most important unit in the total area of architecture serial with the accumulator and multiplier, because the FIFO has to store N (number of coefficients). So the filters with more coefficients will occupy a large area like the filter 1 or 5. Fig. 10 and Fig. 11 show plot of area of the three architectures of this study, for 16 and 32 bits input signal, respectively.

The parallel architecture with multiplier has a quadratic growth with the number of bits of the input signal. This results because the multiplier unit has a quadratic growth and this unit is the most important of the units of this architecture. For an input signal of 32 bits the multipliers unit has an average of 73% of the total area of the filters using this architecture. So filters with a great diversity of coefficients like the filters 5 and 9, occupy more area then the others filters because they will have more multipliers. Note that the ARU has a linear growth with the input signal size.

The parallel architecture with MCM has a linear growth with the number of bits of the input signal, because the all units of this architecture have a linear growth. The ARU unit is the most important of the units, with an average 88% of the total area for an input signal of 32 bits. As the ARU is the most important, filter with a greater number of coefficients, like the filters 1 and 5, will occupy a large area. The architecture serial with accumulator and multiplier is the smallest, occupying on average less than 40% of the parallel architecture with MCM. To parallel architecture with MCM observe an average decrease of 80%, when compared with the parallel architecture with multipliers.

The result of power can be obtained from the Report power of design_vision. The serial architecture with accumulator and multiplier has relatively low power, on average 18 mW per clock cycle. However since it is a serial architecture, it calculates in sequence the results of the filters coefficients, will require N+1 cycles (N is a number of filters coefficients) to perform the calculation. Note that filters with more coefficients have higher energy consumption.

For the parallel architecture with multiplier the ARU is the most important, representing on average 50% of the total power. The power lost in the interconnections is on average 35% (because de quadratic growth of multiplier unit), and the multipliers unit is the less important. The ARU and the interconnections are the more important reasons by power consumptions, filter with great number and diversity of coefficients, like filters 5 and 9, will have higher power dissipation. Fig. 12 shows the plot power for all filter architecture with 32 bits input width.

For the parallel architecture with MCM has the ARU is the most important in the consumption power, on average 67% of total power. The power lost in the interconnections is on average 25% of the total power, and the MCM unit is the less important. As the registers unit is the most important, filters with great number of coefficients (table 3 Fig. 9) will spend a higher power. Fig. 12 shows the plot power for 32 bits input width.

When comparing the different architectures can conclude that architecture serial has a higher consumption of energy (consuming less power in a clock cycle). Because the architecture serial needs, more than one clock cycle (N+1) to calculate a result of filter.

The power consumed by the parallel architecture with MCM has averaged a 30% reduction in relation to architecture with parallel multipliers. Although, for booth architectures the ARU are more important, in power consumption, but the parallel architecture with multipliers for the same filter has higher power, because the losses in the interconnections
(35%--with multiplier; 25%--with MCM) and the multiplier unit are higher. So filters with great coefficients diversity, the power reduction of the parallel architecture with MCM will be higher. Fig. 12 shows the power plot with the differences between the parallels architectures.

The result of frequency (delay of critical path) can be obtained from the Timing report of design_vision. The serial architecture with accumulator and multiplier has the longest delay of all architectures, with 27 ns on average. The critical path consists of an adder and a multiplier, that has to have a large bit width to support all operations. The mathematical operations are done sequentially, is necessary N+1 cycles to get the correct filter output, need on average 2.2 ms to realize all the operations. The critical path of the parallel architecture with multiplier consists in a multiplier and an adder. The critical path of the multiplier is the most important in the total of the delay. If the input signal has a larger number of bits than the filters coefficient, the signal input will define the size of the multipliers. When the input signal define the size of the multipliers (all multiplier with the same size for this example), the differences in the critical path are the dimensions and the fan-out of de adders in the registers unit. Fig 13 shows the delay plot obtained for 32 bits input width. The critical path of the parallel architecture with MCM consists in a series of adders (from MCM unit and registers unit). The MCM unit has a higher weight in the critical path that the registers unit. So, the filters with the largest chain of adders will have the largest delay time (Table 3), e.g. filter5. However, the fan-out and the dimension of the adder also influence the delay of critical path. Fig 13 shows the plot of delay time for 32 bits input width. The serial architecture with accumulator and multiplier has the worst frequency and the speed to calculate the output filter, because need N+1 cycles to calculate the filter output. The parallel architecture with MCM has always better operating frequency than other architectures. The parallel architecture with MCM has an average operating frequency 35% higher than the parallel architecture with multipliers.

6. CONCLUSIONS

This study concluded that the serial architecture with accumulator and multiplier occupies less area than the others architectures. The area of this architecture has a linear growth with the number of bits of the input signal. This architecture consumes less power in clock cycle, but need N+ 1 cycle to calculate the filter output. Although the serial architecture consumes less power and spends more energy. This is the smallest architecture with the worse operating frequency.

The parallel architecture with multipliers is the architecture that, from our study, has the largest area after implementation. The area has approximately a quadratic increase with the number of bits of input signal. This architecture consumes more power than the other architectures, but has a lower consumption of energy than serial architecture. The operating frequency is bigger than serial architecture with accumulator and multiplier, but is less than the architecture parallel with MCM.

The parallel architecture with MCM occupies less area than parallel architecture with multiplier, but takes more area of serial architecture with accumulator and multiplier. The area of parallel architecture with MCM has a linear growth with the number of bits of the input signal. This architecture has the lowest energy consumption and in the operating frequency.

This study concludes that the serial architecture with accumulator and multiplier is the worst solution, when priorities are frequency and energy. But it’s the best solution when the priority is the area.

The parallel architecture with MCM is the best solution when the priorities are frequency and energy. This study concludes also that parallel architecture with MCM is always a better solution than parallel architecture with multiplier.
7. REFERENCES

[1] Levent Aksoy, Eduardo da Costa, Paulo Flores and José Monteiro
“Exact and Approximate Algorithms for the Optimization of Area and
Delay in Multiple Constant Multiplications” IEEE TRANSACTIONS
ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS
AND SYSTEMS, VOL. 27, NO. 6, JUNE 2008.

[2] Levent Aksoy, Eduardo da Costa, Paulo Flores and José Monteiro
“Optimization Algorithms for Multiple Constant” Chapter in book
“Advanced Topics on VLSI Design”, pages 71-99. Nº2 in innovation

decomposition for power and hardware optimization in linear DSP
synthesis,” IEEE Trans. Very Large Scale Integ. (VLSI) Syst., vol. 8,

less FIR Filters with Minimum Number of Additions” In: Proceedings of

algorithms for optimization of area and delay in digital filter synthesis,”


[8] Potkonjak,M., Srivastava,M., Chandrakasan, A.: Efficient Substitution
of Multiple Constant Multiplication by Shifts and Additions using
Iterative Pairwise Matching. In: Proceedings of Design Automation