menu

Scalable Parallel Architecture for ultra fast FFT in an FPGA

Jean-Marc Rémondeau.

Xilinx Europe.

Jeanm@xilinx.com

ABSTRACT:

In this paper a high performance FFT processor architecture suitable for FPGA is proposed. The aim is to cover the very high data throughput applications, above 100M Samples/sec for any size up to 4096 Complex points, in a single FPGA. Firstly, the architecture described shows how to take advantage of all the dedicated FPGA's resources. Then, by using a very simple and powerful distributed control, it allows a highly scalable FFT architecture. Finally, with the help of a good design methodology, the very high performance/area ratio obtained puts the FPGA solution in the very intensive computing domain, at the lowest cost.

1 Introduction

Many implementations of high speed Fast Fourier Transform have already been done. So far, few reach the 100 Ms/s barrier. The so-called pipelined FFT has always provided high data rate. However, none of them addresses the field of the very high-speed transform above 100 M Samples/sec data throughputs.

A good implementation was proposed by FFT chip manufacturers [1] using multiple FFTs in parallel [2]. This high performance technique has, however, two disadvantages. First, it is limited by the speed of each FFT chip. Then, the complexity of the interconnection and control of the system become very high as the parallelism grows. Therefore, the cost, still lower than several DSP boards, is high. Others solutions, such as the Column implementation [3], provide a very low latency, but require many chips as the number of points grows. This paper covers the row pipelined parallel Radix-R approach.

2 Fully parallel pipelined FFT

Many others ASICs FFT are based on a bit serial architecture in a data parallel structure [4]. This approach requires more flip-flops than the bit parallel. Also often used in FPGA designs [5], it requires parallel to serial registers between each stage [6]. Thus, the performance/area ratio is not as good as a bit parallel approach. By using the XILINX VIRTEX™ Fast Carry logic [7], a bit parallel adder/subtracter can be almost as fast as a bit serial one, allowing a 16 bits DataPath to run at near 200 MHz, while halving the number of registers.

A natural and efficient approach is to use a data parallelism equals to the Radix. Many papers already shown this technique [4]. For radix-4 FFT, 4 datum are processed in parallel, so the rotators can be better utilized, given that only three are needed, plus a delay unit to compensate the pipeline. Compare to 4 FFT chips in parallel, it saves 25% of the area only for the execution unit. This technique can also be used with a bit serial approach. However, to achieve a very high speed transform both Bit parallel and Data parallel are required.

2.1 Architecture overview

The general architecture of the design, shown figure 1, is very similar as the standard pipelined FFT except that several datum are processed in parallel, thus multiplying the speed. One of the properties of the pipelined FFT is that the size of the memories decreases at each stage. This is interesting in XILINX VIRTEX FPGA, as we can use the large 4K bits Block Select-RAM™ first, and then use the smaller 16 Bits of Distributed Select-RAM™. When input datum are taken in a natural order, a first memory stage is required to permute the samples. For a 1024 points, the normal input flow is: x(0), x(1), x(2) and x(3), but x(0), x(256), x(512) and x(768) will be processed at the same time in the first stage. Then, another memory bank, of size 1024, might be used to reverse the outputs to get a natural order.

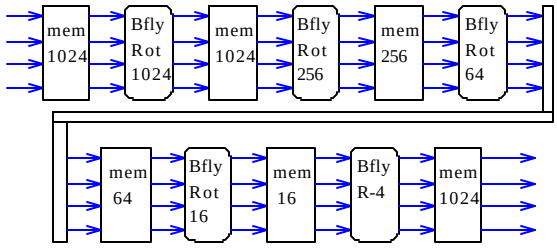

Figure 1: The parallel FFT Data-Path

2.2 Butterfly Radix-4

A direct 4 complex points FFT is built using 8 adders and 8 subtracters, see figure 2. For 16 bits, the area utilized is 2 columns by 32 rows of VIRTEX Configurable Logic Blocks. This is twice smaller than a direct 4 points Discrete Fourier Transform.

:

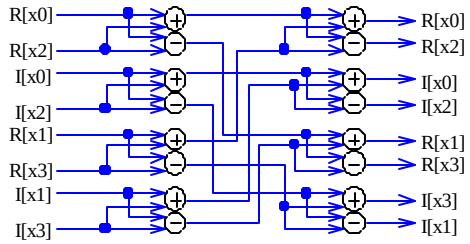

Figure 2: The Butterfly Radix-4 Unit

2.3 Rotator

There are 2 ways to design a rotator. The first one is to use complex multipliers requiring 4 multipliers plus 1 adder and 1 subtracter. The VIRTEX devices have 4 built in 1 bit Multipliers in each CLBs [7], allowing to do fast and compact multipliers of any size. The speed, above 150 MHz for a 16 bits, allows the use of only 2 multipliers and 1 add/sub to build a complex multiplier running at 75 Ms/s data rate.

The other way is the CORDIC rotator. It is perfectly suited for FFT [8], because of its high performance/area efficiency. The pipelined CORDIC is often used for high performance signal processing, in addition, it fits perfectly in XILINX FPGA [9]. With the same area of the complex multiplier described above it allows a data rate of more than 130 Ms/s in the VIRTEX device. Both CORDIC and rotator with multipliers require the same amount of ROM to store the coefficients. However, there are more latency with the pipelined CORDIC, 16 compared to 5 for 16 bits, and we often have to compensate the inherent gain. When the latency is not an issue, the CORDIC method is preferred for the high speed FFT, see figure 3.

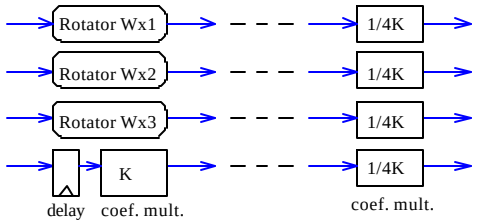

Figure 3: Rotation Unit DataPath, with gain compensators

The phases generation only needs a N/4 counter, for an N bits FFT. Then, at each stage, this address bus is successively divided by 4 to match with the corresponding table coefficients. Between each stage, a Time Skew Buffer is inserted to compensate the pipelined delay.

The area used for each 16 Bits rotator is 17 columns by 9 rows VIRTEX CLBs. The ROM containing the coefficients table is done with one embedded VIRTEX BlockRAM for the 1024

and 256 stages. The 64 and 16 points stages can use the remaining Look Up Tables in the CORDIC module, thus consuming no additional CLBs.

The delay element, built with the VIRTEX dedicated shift registers [7], is connected with a constant coefficient multiplier which compensate the gain of the 3 CORDIC rotators. Then, if required, the 4 outputs of the final FFT stage are passed through 4 constant coefficient multipliers to re-adjust the gains of all the stages.

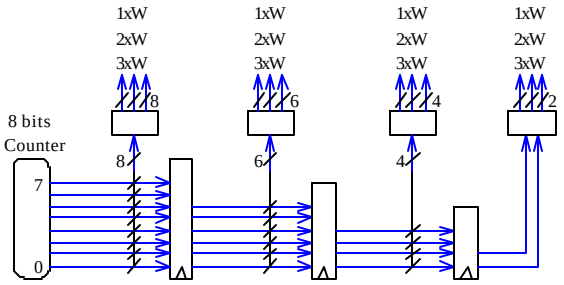

Figure 4: The whole Rotation Unit Control Path for 1024 points

2.4 Memory Architecture

This extremely fast transform shows some issues in managing the memory bandwidth. Some good implementations have already been done using a number of memories bank equals to the Radix used. For Radix-R, R memories are used [4]. However, they require 2 levels of crossbar RxR to do the data permutation. In addition the memory controller could be complex. Others, by choosing a suitable architecture [10], avoid the data transposition, while increasing the complexity.

The architecture proposed performs the Matrix transposition being processed at each stage of the FFT. The base idea is to use RxR memories, each of size N/R^2 for a Radix-R N points FFT. The high number of small memories required might be an issue for ASIC designs. However, the large amount of registers and distributed RAMs available, and the sufficient number of BlockRAMs easily allow this implementation in XILINX VIRTEX FPGA, see figure 6. For the Radix-4 1024 points FFT, the 2 first memory banks use the 4K bits BlockRAMs, then the 256, and 64 points stages use the distributed RAM, and finally, the last 16 points stage use the registers. Figure 5 shows the memory mapping and transposition for the last stage, the 16 points FFT. For the others stages, the technique is the same except that each element contains several datum, 4 for the 64 points, 16 for the 256, 64 for the 1024, and so on. The common memory address bus accesses the 16 RAM elements.

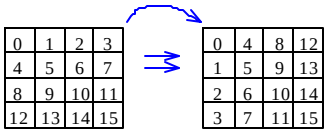

Figure 5: The Matrix transposition for the last stage

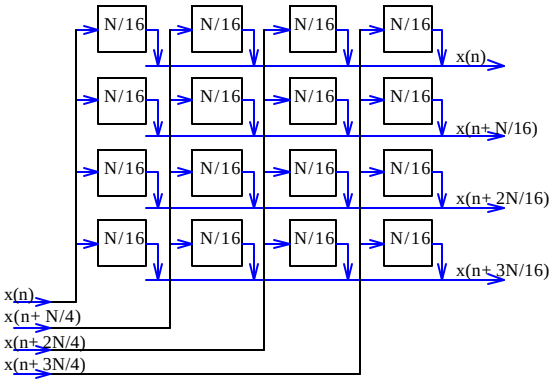

Figure 6: Memory architecture DataPath for N points Radix-4 FFT. The Double buffering and the I part are not shown for clarity

The memory architecture providing itself the partial transposition, the Control Unit is easy to do. The address bus is common for all the RAMs. The Write Enable signals are common for each row, and the Output Enable signals are common for each column, see figure 7. Using a double buffering, with the next set of datum, the context is switched by using the MSB+1 address bus. The input RAM, say A, becomes the output RAM B. At each stage, the total amount of memory used for N complex points is 4N. The output Switches use the VIRTEX dedicated Tristate Buffer for Radix-4 and higher, therefore, use no additional logic. Mixed Radix can also be

designed using this technique. For example, a radix-8 stage connected to a radix-4 stage requires 8 columns by 4 rows of memory. In that case Data flow must be adjusted, this solution is not covered in this paper.

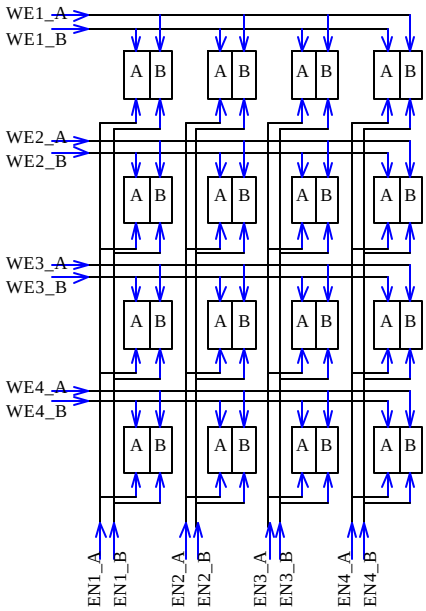

Figure 7: Memory Array Control lines

As it is shown figure 8, the Control unit is very similar to the Phase Rotator Control Unit. After each stage, the number of phases and the memory size decrease by a factor R, for a given Radix-R. In order to be in timing with the Rotators, this control bus is in fact shared with the Rotation Unit Control Bus.

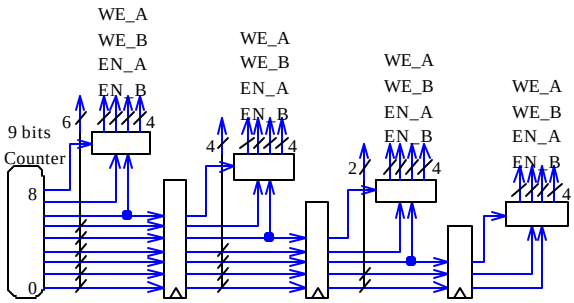

Figure 8: Memory Array Control logic for 1024 points FFT Radix-4

2.5 Performance and Flexibility

The highest speed transform is obtained when using the largest possible Radix and the implementation of all the FFT stages. A 4096 points pipelined FFT radix-8, running at 125 MHz can reach the 1 Giga s/s barrier. The same FFT with a radix-4 structure can support the 500 Ms/s with almost half the area. At the low end, 3 rotators, 2 butterflies radix-4 and 2 memory arrays are enough to achieve 100Ms/s sustained for a 1024 points FFT. However, in that case the parallel FFT is not pipelined anymore, so the control is less easy than described above. Table 1 shows the performance and area of different radix-4 FFT sizes. Although the highest data throughput was the main goal of the project, the latency obtained is very low, far below many others solutions.

System Freq.	125 MHz		**500** MS/sec	
FFT size	**4096**	**1024**	**256**	**64**
Latency in µs	**11,10**	**2,87**	**0,79**	**0,24**
CLBs Area	**4481**	**3188**	**2665**	**2142**
1/Latenc.*area	0,02	0,11	**0,48**	**1,95**
(Ms/s)/area	111,58	156,84	**187,6**	**233,43**

Table 1: Device utilization and performance for a 16 bits FFT Radix-4

Compared to table 1, table 2 shows the results for a radix-2 FFT. The simplicity of the architecture allows a higher frequency, thus, keeps an interesting data throughput. Known to be less efficient than higher Radix, it can still be a good solution when latency is not an issue.

System Freq.	150 MHz		**300** MS/sec		
FFT size	**512**	**256**	**128**	**64**	**32**
Latency in µs	**3,83**	**2,06**	**1,15**	**0,66**	**0,39**
CLBs Area	**1585**	**1280**	**1167**	**1094**	**941**
1/Laten.*area	0,16	**0,38**	0,75	**1,38**	2,75
(Ms/s)/area	189,3	**234,4**	257,1	**274,2**	318,8

Table 2: Device utilization and performance for a 16 bits FFT Radix-2

2.6 Design methodology

Although the XILINX VIRTEX FPGA are very fast and have rich routing resources, the highest performance/area ratio can only be obtained by using intensively all the dedicated resources combined with a very good floorplanning. The designs described are done using the Relative Placement Macro technique [11]. They can even be designed with a higher flexibility by using the XILINX LogiCore™ methodology [12]. In either case, the use of RPM is the key for very high and predictable performances while keeping the smallest area as possible. The final placement for the whole design can be made by using tools like the XILINX Floorplanner especially to control the I/Os interaction and placement. The engineering time for this extra effort doesn't exceed 10% of the entire development time, while getting 20% to 40% more performance, and even 50% on a huge, 2 Millions gates, design such as a 4096pts 16Bits FFT @ 1 Gs/s.

3 Conclusion

The aim of this paper was to cover the Fast Fourier Transform in real time condition, with very high data rate. This open the door for real time Radar/Sonar application where not loosing data sample is crucial. The field of Ultra Fast Convolution is also possible now in a simple way: a single VIRTEX FPGA can perform a 1024*1024 real convolution, assuming 16 bits data path, with a continuous data throughput of 500 Ms/s. There were no "simple" solution on the market before. We needed to use many DSP boards, or a custom multi FFT chips board. Consequently, a significant cost reduction is obtained. A study can also be made on power consumption improvement. The power control is easily done by using the dedicated Die-temperature sensing Pins.

With its high arithmetic performance, its high memory bandwidth and its power management capability, the VIRTEX FPGA is clearly an ideal solution for the very intensive computing demand.

4 References

[1] LH9124, SHARP, DSP Application Note.

[2] C. J. Ju., "FFT-Based Parallel Systems for Array Processing with Low Latency", ICSPAT 1997, San Diego, California.

[3] COBRA: A High Speed FFT Chip, Data Sheet. Dr T. Chen, DEE, Colorado State University.

[4] J. Melander, T. Widhe, L. Wanhammar, "A Radix-r FFT/IFFT Architecture with distributed control unit", NORSIG'96.

[5] R. J. Andraka, "Building a High Performance Bit Serial Processor in an FPGA", Proceedings of Design SuperCon'96, Jan 1996, P5.1 - 5.21.

[6] L. Mintzer, "Large FFTs in a single FPGA", ICSPAT'96.

[7] XILINX, "The programmable Logic Data Book" 1999 Edition.

[8] A. M. Despain, "Fourier Transform Computations Using CORDIC Iterations", IEEE Transactions on Computers, Vol.23, 1974, pp.993-1001.

[9] R. J. Andraka, "A survey of CORDIC algorithms for FPGA based computers", FPGA 98 Monterey CA.

[10] D. R. O'Hallaron, P. J. Lieu, L. P. Withers, J. E. Whelchel, "Computing the Pipelined Phase-Rotation FFT", Proceedings of the Scalable High Performance Computing Conference, Knoxville, TN, IEEE, May, 1994, pp. 462-469.

[11] XILINX Library Guide 1999.

[12] S. Mohan, R. Wittig, S. Kelem, and S. Leavesley, "The Core Generator Framewok", FPD '98, Montreal, June 1998.