High throughput design & implementation of multi- FFT/IFFT core in FPGA for hardware acceleration

Hassan Raza
Department of Electronics & Computer Sc. Engineering
Visvesvaraya National Institute of Technology, Nagpur – 440010, INDIA
hassan_raza@ece.vnit.ac.in

Abstract - This paper presents design and implementation of self contained FFT/IFFT transform cores tightly coupled on an FPGA. The FFT/IFFT cores used are designed for OFDM base band transceiver circuit of IEEE 802 .11a WLAN system for calculating the transform. The design consist of a high end bus interconnect named as Processor to board crossbar (PBX) to route the data among the connected FFT/IFFT cores. The PBX design is highly scalable to accommodate desired number of cores. The prototype system consisting of 8 FFT and IFFT cores and is ported on to an FPGA device to get throughput of 8X. It is concluded that the throughput increases nX times the n- number of cores connected.

Keywords: FFT/IFFT, hardware accelerator, FPGA, multiprocessor interconnect

I. INTRODUCTION

High performance computing industry needs to reinforce the concept of '*Throughput Computing*' [1] where in the computing paradigm is throughput oriented instead of pure performance. System designed to achieve high throughput concentrate more on overall work performed. The extent of throughput computing is vast, in this context multithreading and multi-core technologies are used predominantly to increase throughput. Using such design aspects performance can be improvement by a factor of 30x as compared to conventional systems. The discussed concept of achieving throughput splits the design engineers community into two with[1] software engineers striving to increase thread[2] [3]level parallelism while the hardware engineer taking on with accommodating more and more cores on a single chip. One find's FPGA [2] as one of the many solution to achieve high throughput. FPGAs provide the power of flexibility, high speed, high gate density necessity to accommodate more cores and low NRE cost. FPGA is treated as the flexible part of the static hardware at least up to its I/O pins. It provides liberty to change the internal design, as long as the I/O remains where it was. Using this unique feature designer can cast a flexible hardware consisting of task devoted processing element to be configures as and when required. The concept well known as - '*FPGA hardware accelerator*' [3] is becoming popular as performance accelerator or simulation accelerator. [4].

In the hardware realization of OFDM base band transceiver circuit for IEEE 802 .11a WLAN [5], designing of FFT/IFFT block [6] creates a real challenge. According to the specification transceiver has to perform 64-point FFT/IFFT within 3.2 μs time frame. The conventional radix-2 FFT algorithm [7] requires 192 complex butterfly operations for a 64-point FFT computation. One butterfly operation has to be completed within 16.6ns which is not administrable in addition to that complex twiddle factor handling, storage and complex intermediate data adds up to the complexity. The radix-4 FFT algorithm is most popular and has the potential to satisfy the current timing need. However, a single radix-4 butterfly [7] requires three complex multiplications and eight complex additions.

[1] Hassan Raza

Thus, in order to carry out one radix-4 butterfly operation per clock cycle, one needs to complete 12 real multiplications and 16 real additions at each cycle. Practical implementation of a real multiplier to output in one clock cycle is a challenge. One can think of designing a dedicated hardware to achieve this required level of performance or one can even go for FPGA based hardware accelerator where throughput is guaranteed.

In this paper we describe hardware accelerator for IEEE 802.11a specific FFT/IFFT core. The hardware acceleration is achieved by off laying the transformation job onto uniformly connected FFT and IFFT core on an FPGA. The architecture comprises of 32 bit bi-directional link to PCI bus interface, FFT/IFFT block and a Bus transform design to interconnect the required number of cores. The bus transform architecture is implemented to behave as a crossbar switch and distributes job among the FFT/IFFT cores. The design uses minimum gate resource and can accommodate n number of cores subjected to maximum available resource. Example inputs demonstrate the implementation and performance of the complete system and results in terms of throughput achieved are discussed.

II. Related work

Keith reeves et al [8] has discussed about hardware accelerator platform for embedded DSP. They have discussed about reconfigurable processor architecture with multi – floating point multiplier, FFT and IFFT unit. They propose performance improvement for general purpose FFT/IFFT. We present an analogous design with FFT/IFFT core used in IEEE 802.11a specific OFDM base band transceiver design where specification of the cores changes as compared to general purpose core.

III. FFT/IFFT for IEEE 802.11a OFDM transceiver System

The FFT/IFFT is an integral component of the PHY layer of OFDM-based communication systems. According to the specifications of IEEE 802.11a the OFDM transceiver has to perform a 64-point IFFT (in the transmit direction) or FFT (in the receive direction) within 3.2 μs. This implies that a highly specialized architecture has to be used to satisfy this tight timing constraint. Also, from a power dissipation point of view an implementation using dedicated hardware is beneficial when compared with a general-purpose DSP architecture [8]. The 64-point FFT can be reformulated in terms of 2D 8 point FFT as described in [5]. First performing an 8-point FFT of the appropriate input data slot, then multiplying them with twiddle factor and finally once again generating an 8-point FFT of the resultant data can compute the 64-point FFT.

The VHDL module for 64-point FFT [9] is designed using 2D 8-point FFT with FPGA as the target device. The 8-point FFT unit takes 11 cycles to complete one FFT operation. Pipelining gives successive FFT-8 every 9th cycle. The real multiplier takes 7 clock cycles and complex multipliers also takes 9 cycles. Thus pipeline is perfectly balanced. About 160 cycles are required to complete entire 64 point FFT. Here every cycle is the time required to finish 16-bit addition operation. Thus to finish entire operation in 3.2 μs as per the 802.11a standard definition, we have to finish this in 20 ns which requires 50 MHz frequency for its operation. A single 64-point FFT core when implemented on Vertex -5 series sx240 device runs at maximum frequency of 43 MHz. It is seen that a single core not meet the timing constrain if we consider the FPGA implementation of the complete OFDM transceiver circuit. The single core needs to be redesigned with further code optimized to achieve the desired speed

IV. System architecture & design

The system design progress with component based design and relies on use of IP blocks along with industry standard interfaces [10]. Components and megacore IP blocks provided by Altera [11] and Xilinx are extensively used in the design. These cores are device specific and are optimized to provide maximum performance with least resource usage. The bus transform module to interconnect the required number of cores is named as Processor to board crossbar (PBX). The PBX is divided into two sub module.

System to processing element crossbar path (SPX) & Processing element to System crossbar path (PSX).

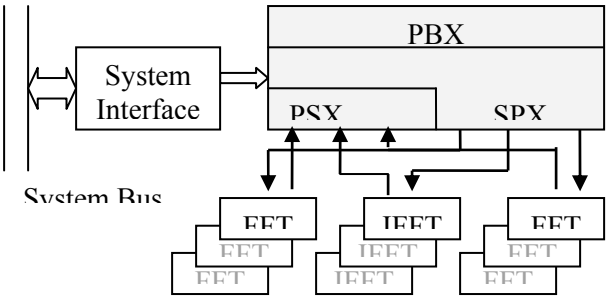

Figure 1: Block representation of system to processing element crossbar path (SPX) & processing element to system crossbar path (PSX)

This module takes care of the inward and outward flow of data from the system interface to the processing cores.

A. SPX internal architecture & design

The SPX unit consists of two functional blocks as inscribed in figure 4

1. A 128x5 array for temporary storage of data termed as swapin here.
2. Control logic to issue all control signals.

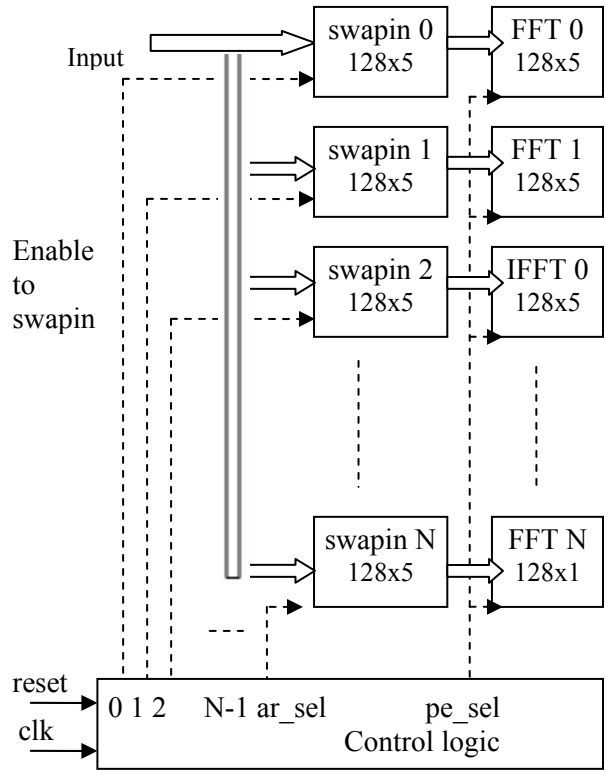

Figure 2: Internal Blocks of system to processing element crossbar path (SPX)

Input 32 bit wide data from the system interface is made available to all the swapin blocks. The swapin blocks are temporary storage slice registers of FPGA whose size is equal to the input requirement of FFT/IFFT block. The number of swapin blocks required depends upon the number of processing elements (N) connected to the crossbar. For example - as visible in the figure 4 cores, if four processing elements such as FFT or IFFT are connected accordingly the four swapin blocks are connected. . The size of the swapin depends on input bits required by the processing elements. We have preferred 64 point FFT and IFFT design as the default PE which requires input stream of 127 bit length with 5 bit depth. It becomes essential to hold this set of data till the PEs is ready to perform operation. Eventually 128x5 bit array is required for N number of PEs.

Each swapin block is enabled for 20 clock cycles by select lines provided by the control logic unit. This is done to fill the swapin block (20 clock cycle x 32 bit data = 640 bits data) required by a single FFT or IFFT module. The array unit is designed with D flip flop as the basic building block to hold a single bit data. Care has been taken to avoid inferring latches during synthesis of the design

The control logic design is constructed by means of a 10 bit free running counter. A series of control signal are generated as per the specific clock count.

B. PSX internal architecture & design

The PSX unit also consists of three functional blocks as inscribed in figure 5

1. 16x16 Array to hold the output data from PEs termed as swapout here.
2. A 2:1 bus multiplexer to consolidate the data to 32 bit output
3. Control logic to issue all control signals.

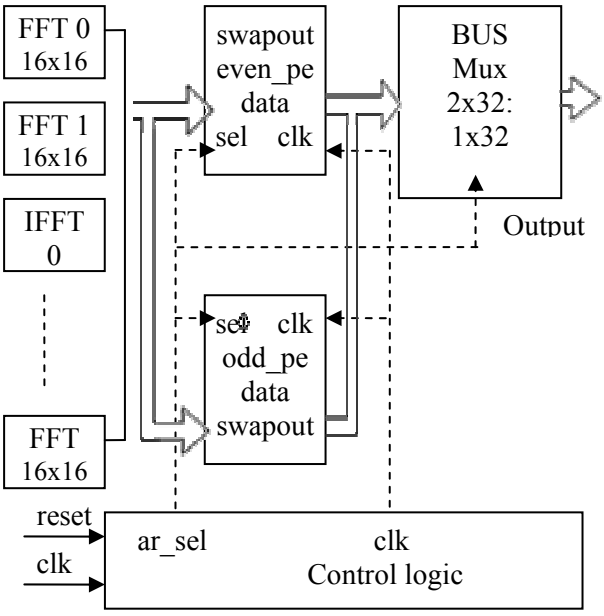

Figure 3: Internal Blocks of processing element to system crossbar path (PSX)

The processed data output from the PEs are hold into a 16x16 array which are slice registers of FPGA. They are two in numbers and termed as swapout here. All even PEs is connected to the even array (swapout0) and all odd PEs are connected to odd array (swapout1). The array size so chosen is due to the output size of the FFT and IFFT PE which provides 16x16 output lines. To reduce the design space trade of is done here to reduce the required number of swapout blocks. At any point of time all PEs will not provide simultaneous output, and considering this limitation the outputs of even PEs are routed to even array while that of odd PEs are routed to odd array. Eventually consecutive PEs are connected to even or odd swapout block. A 2:1 multiplexer is unit to multiplex the data output from even and odd swapout to one output stream of 32 bit.

The output from the array is in the form of 32 bit packet so it requires 8 clock cycle to vacate the data from array. The data from either of the array is directed back to the system interface unit.

The same control logic takes care to generate the select signal for the even and odd array. Either of the two is selected at a time and the same signal is made available to the bus multiplexer system.

V. Experimental results

The complete system design is implemented in VHDL. EDA tools such as ISE (Xilinx) synplify_pro (synplicity) [12] and Questasim (model) [13] were used. Design was implemented and verified with due care for any warning or error. A set of input data was provided to al the PEs through a testbench. It was seen that changing the number of cores connected the throughput gets affected. The more the cores, the more is the throughput and with a slight trade off between number of PEs connected and throughput required. We concluded that for our design under consideration with 32 bit of data per clock cycle available 20 clock cycles is needed to meet the data requirement of one FFT core and each of this core takes 160 clock cycle to compute the transform. Thus 180 clock cycles is required by a single core to complete one transform. Eight such FFT/IFFT core when connected to each other work independently but the data supply to all get pipelined. After a burst period of 160 clock cycle we get the transform done in every 20 clock cycle. The same has been summarized in table 1 below. As a resultant of the exercise we get a throughput of 8x as compared to a single unit.

The synthesis results of eight 64-point FFT/IFFT core when implemented on Vertex -5 series sx240 device attains a maximum frequency of 39 MHz. the time period of single clock cycle comes to be 25ns and within 20 such clock cycle we get the transformation done. 0.5 μs time is required for the FFT/IFFT transform which is far below the expected 32 μs time frame. We can still further reduce this operating frequency of 39 MHz to 6.25 MHz to reduce the power requirement of the circuit.

Selected Device : 5vsx240tff1738-2							
FPGA resource	Available	1 FFT/IFFT core		4 FFT/IFFT core		8 FFT/IFFT core	
		Resource utilized					
Slice Registers	149760	7125	4%	27869	18%	54470	36%
Slice LUTs		15416	10%	59752	39%	112476	75%
LUT flip flop		16193		62714		118478	
IOBs	960	68	7%	73	7%	74	7%

BUFG	32	9	28%	16	50%	16	50%
Maximum Frequency		40.791MHz		42.602MHz		38.124MHz	

Table 2: Synthesis report summary of the design with 1, 4, 8 cores

Synthesis result of the complete system has been shown in the table 2. It gives resource utilization of the device vertex 5 SX240-2 considering different number of FFT/IFFT cores. Values in column 3 indicate the FPGA resource usage for single FFT/IFFT core only. As shown in column 4 and 5, FPGA resource usage for 4 cores and 8 cores respectively. Close analysis of result depicted in the table reveals fantastic facts about the implemented design. When we go for 8 cores as compared to single core penalty for 4% slice registers is imposed where as saving of 5% slice LUTs. In addition to that 8% saving on LUT flip flop. We have considerable saving in IOBs and BUFGs, while the maximum frequency of operation is roughly same. We conclude that the PBX architecture designed does not consume large FPGA resource instead as the design grows with increase in number of cores it get balanced or even saving of some resource. The PBX design FPGA resource utilization do not adds up to the resource requirement.

VI. CONCLUSION

Using FPGA based hardware accelerator for rapid prototyping of multiprocessing elements seems to be futuristic step for increasing throughput. Design of BUS transform interconnect logic requires consideration for scalability and resource usage. It should take care of synchronous data routing to all cores connected. Routing data to and fro different cores involve critical timing issue, hence synchronization is important. While designing individual processing element (PE) efficient architecture selection is needed to avoid redesign. The PE should be optimized to consume less resource and work at high frequency. Scaling of the design depends largely on the resource available and the resource requirement of the PEs.

REFERENCES

[1] Sun Microsystems Inc., http://www.opensparc.net, accessed on June 09, 2008.

[2] Xilinx Inc., http://www.xilinx.com, accessed on July 03, 2008.

[3] Geno Valente XtremeData Inc, “T o accelerate or not to accelerate...”Article. http://www.pldesignline.com/208403201, June 13, 2008.

[4] IMAGE simulation accelerator - Powai labs pvt ltd Inc., http://www.powailabs.com, accessed on July 15, 2008.

[5] K. Maharatna, E. Grass, and U. Jagdhold, “A novel 64-point FFT/IFFT processor for IEEE 802.11a standard,” in Proc. ICASSP’03, vol. II, pp. II-321–II-324.

[6] John Proakis, Dimitris Manolakis “Digital Signal Processing”, Prentice Hall, 4 edition, April 7, 2006.

[7] S Chan, C. Yu, C. Tsai and J. Tang, “*A new IFFT/FFT hardware implementation structure for OFDM application*”, The 2004 IEEE Asia-Pacific Conference on Circuits and Syt., Dec 6-9, 2004.

[8] K. Reeves, K. Sienski and C.Field, “Reconfigurable hardware accelerator for embedded DSP”,.

[9] R Phadkhe, “Design of baseband transmitter and frequency synthesizer for IEEE 802.11a WLAN, MTech thesis VNIT, Nagpur.

[10] International Roadmap committee. *The International technology roadmap for semiconductors: 2005.* http://public.itrs.net, accessed on July 15, 2008.

[11] Altera Inc., http://www.altera.com, accessed on July 03, 2008.

[12] Synplicity Inc., http://www.synplicity.com, accessed on July 10, 2008.

[13] Mentor graphics Inc., http://www.mentor.com, accessed on 10-07-2008.