Hardware design of Real time Configurable Variable point FFT for multi-rate applications

K.L.Ramesh
University College of engineering Kakinada
East Godavari Dist., A.P, INDIA-533003
kondepudi.ramesh@gmail.com

Smt. K. Jhansi Rani M.Tech
Assistant professor
JNTUK, kakinada
Jhansi.rani@gmail.com

***Abstract*— Fourier transform algorithm has encompassed diverse fields of engineering including specialized fields like radars, communications and image processing systems. Therefore there have been continual efforts to improve the efficiency of FFT implementation in real time systems and other hardware. To reduce design time and time to market, FPGA vendors have developed IP cores which can be readily used in our applications. But these IP core designs though efficient are highly abstract and do not provide the designer to modify them according to his requirement which leads to inefficient design realization. Vendor provided IP cores do not give access to FFT kernel matrix thus restricting the configurability and efficiency of using them. In this paper we have designed a customized architecture to perform FFT with access to twiddle factors for improved configurability. The designed architecture is further modified to perform variable point FFT targeted for application in multirate systems. The architecture designed is generic and can be implemented on any vendor platform.**

Keywords- Fast Fourier Transform, Field programmable Gate Arrays, IP core, SAR Image processing, kernel matrix.

I. INTRODUCTION

With the proposal of FFT by Cooley and Tukey [1] in 1965, DFT analysis has been exploited in diverse fields of engineering. In specialized fields like radars it has given a new direction in detection and estimation analysis of moving targets[2]. In Image processing, FFT is used to compress the data to improve processing efficiency [1] and reduce bandwidth for efficient transmission. It is also used for noise removal in an image. In SAR Image processing 2D FFT is used to convert the phase space to a real space (complex) image [7][8]. Gautam et.al[10] proposed variable point FFT using CORDIC for OFDM applications. It is apparent that FFT has wide utilization space which led many of the hardware vendors to design standard FFT IP core, which can be used by the engineers directly in their system design. Though IP cores reduce design cycle time they do it at the expense of implementation configurability. They do not provide access to the kernel matrix or twiddle factor matrix thus decreasing the option of configurability for designers. Added to the complexity, the above mentioned applications are performed in real time which further constrains the designer in using standard IP cores because of the lack of provision for any further optimizations. In this paper we have designed a customized architecture to perform the FFT with simple memory blocks. The design is highly configurable with access to twiddle factors and hardware efficient. The details are explained in rest of the paper which is organized as follows. Section II explains the need for the architecture. Section III describes the proposed architecture in detail. Section IV elaborates on the simulation results, timing analysis and hardware requirements. Section V provides the conclusion followed by references.

II. NEED FOR THE ARCHITECTURE

In multiple data rate applications or in case of radars with multiple scan rates FFT has a major role to play. In these kind of applications the FFT block should be capable of handling this online configurability instead of using individual FFT block for different data lengths of computation which are redundant and consume more hardware. In [2] we have proposed a architecture for variable point FFT for multiple scan rate radars using IP core. In that architecture, the FFT block of highest length that could be encountered in the design is considered and for the rest of the data lengths encountered , the same design is used by appropriate zero padding. In this scenario the major drawback is the increase in clock rates to handle lower order of FFT computation. The increase in clock rates is to provide for the time of zero padding. Consider for example an application in which mostly 32 point FFT is to be done and occasionally 64 point FFT computation is needed to be performed. For this case IP core has to be designed for 64 point though it is seldom encountered and for 32 point it has be zero padded every time. And to provide real time processing the zero padding is done with increased clock rates. Consider another example in which we have to change the FFT block from 128 point FFT to 64 point FFT. In this case the IP core has to be re generated and should be properly integrated into the design if we have to avoid zero padding. But in the presented architecture since we have access to twiddle factors, the design reconFig.s its timing signals and registers online and solves both the problems mentioned above. Next section explains the proposed architecture which is highly configurable and performs variable point FFT without increasing the clock rates thus realizing a comparatively low power design. Also the design consumes less hardware resources as seen section IV.

III. PROPOSED ARCHITECTURE

Data in real time flows continuously and therefore the first step is to buffer the data to provide for processing time for FFT operation. In the architecture presented all the processing is done in this buffering duration and no additional time is used. Fig. 1 shows the block diagram of the proposed architecture. Each module is explained in rest of the section.

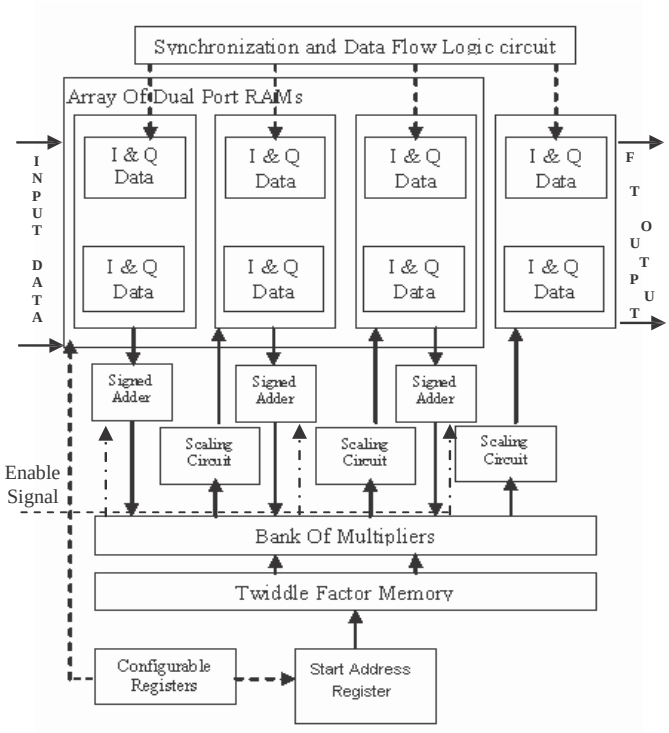

Figure 1. Block Diagram of 8 point FFT for presented architecture

a) *Dual Output Port RAMs :*

From the butterfly diagram in Fig. 3 it can be understood that before multiplying the data with twiddle factors the data has to be processed for addition and subtraction. To do this operation we have used dual output port RAMs or 3 port RAMs (one input and two output). First port reads the data serially from the memory. Second port data is reordered into using carefully designed read address logic to align with the serial data to provide for straight forward addition and subtraction operations. Both the data are passed to signed adder circuit which in fact is an adder subtractor circuit with enable signal to govern the required operation to be done. The same enable signal is used for read logic generation for second port. We used "$2*Log_2N$" dual output port RAMs for the computation of N^{th} order FFT to provide for buffering of both I and Q data for each stage of operation. Each stage in Fig. 3 can be modelled as buffering RAMs used to process the continuous data from previous stage. The time required for buffering the data is used to process the data across signed adder, multiplier and scaling circuit. The architecture is essentially designed as continuous set of buffering RAMs at the outset with all the processing done during the buffering time. The depth of the dual port RAMs is defined by the highest order of input data expected during its functionality.

b) *Twiddle Memory :*

Twiddle coefficients are the crux of DFT operation. Exploiting their symmetry properties FFT has been proposed by Cooley and Tukey[4] to realize DFT using only second column of the twiddle matrix shown in Fig. 2 for any N point DFT and rearranging them as shown in Fig. 3. In [11] we have presented a method to generate twiddle factors from any higher point twiddle matrix to any lower order upto 2. For a variable point FFT operation this can be used to generate twiddle factors for different orders required for the design. Alternatively we can store the twiddle data for all the orders that are encountered in a single memory and read them out according to the selection. Since only N memory places are required to store twiddle data for each computation, the total depth of the memory will be "N1 + N2 + N3 + …." Where NI, N2,N3 etc are the possible orders of the design. For our case we have consider 4 orders of configurability. It can also be observed from Fig. 3 that the same memory can be used to generate twiddle coefficients for subsequent blocks of FFT computation. This method is simple to implement with small changes in twiddle read address for each block of FFT operation. To further reduce the memory requirement we can use the single memory of depth N1 (N1 > (N2,N3,N4…)), the highest order in the design and write new set of twiddle coefficients into the memory during first cycle of operation whenever a change in order is sensed and read them out appropriately throughout the design.

1	1	1	1	1	1	1	1
1	W_N^1	W_N^2	W_N^3	W_N^4	W_N^5	W_N^6	W_N^7
1	W_N^2	W_N^4	W_N^6	W_N^8	W_N^{10}	W_N^{12}	W_N^{14}
1	W_N^3	W_N^6	W_N^9	W_N^{12}	W_N^{15}	W_N^{18}	W_N^{21}
1	W_N^4	W_N^8	W_N^{12}	W_N^{16}	W_N^{20}	W_N^{24}	W_N^{28}
1	W_N^5	W_N^{10}	W_N^{15}	W_N^{20}	W_N^{25}	W_N^{30}	W_N^{35}
1	W_N^6	W_N^{12}	W_N^{18}	W_N^{24}	W_N^{30}	W_N^{36}	W_N^{42}
1	W_N^7	W_N^{14}	W_N^{21}	W_N^{28}	W_N^{35}	W_N^{42}	W_N^{49}

Figure 2. 8 X 8 Twiddle Factor Matrix

c) *Configurability of design:*

The timing signals from the crucial element in this design. A single delay can cause erroneous results. Fig. 4 shows the timing signals for 8 point FFT for explanation. As can be seen there is overlap between write enable of next block with the read enable of present block. That is the data read out from present block is processed and written into the next block of RAMs in the same cycle. As said earlier the RAMs act as series of buffering elements. After initial latency of "$N*Log_2N$" design computes FFT output continuously. When the change in order of input data is sensed the configurable registers are loaded with updated values and accessed through out the design reconfiguring all the timing signals and address generation logic of RAMs for updated order. There is no

latency for this update of the design and the processing continues without any lag. Even if we consider single twiddle memory option the memory is written with updated coefficients simultaneously along with the writing of first block of input data, so the updated coefficients are ready when the data is read out for processing.

d) Bank of Multipliers

The processed data from RAMs and output data of twiddle memory are synchronously passed to multiplier blocks for computation. For this we have used 4 real multipliers and 2 adders for each complex multiplication operation. From the method mentioned in Lyons[3] the number of multipliers can be reduced to 3 for complex multiplication with an increase in adder circuitry. This will result in further reduction in hardware utilization of the presented design as adder circuit consumes less hardware compared to multipliers.

e) Scaling Circuit

In every stage of computation there is increase in bit width due to processing. Scaling circuit is designed to trim the data appropriately to the required resolution and passed on for further processing.

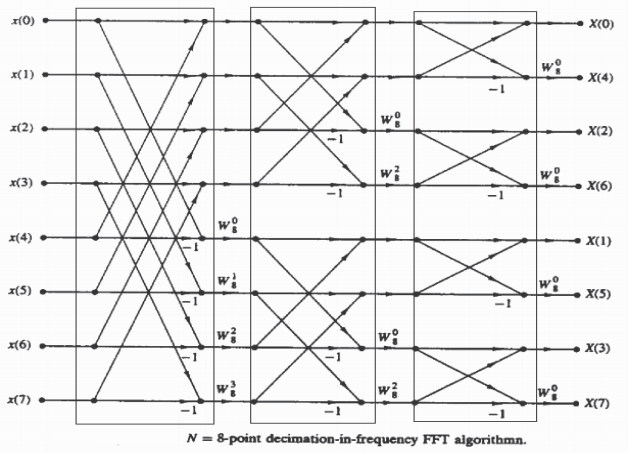

Figure 3. 8 Point FFT Butterfly diagram

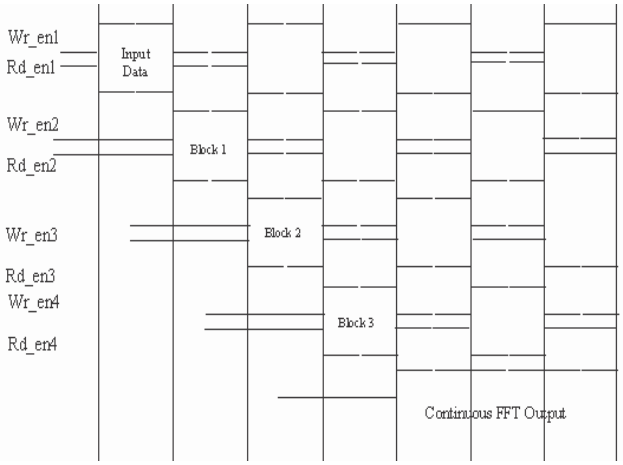

Figure 4. Timing Signal for 8 Point FFT

IV. SIMULATION RESULTS

The architecture is developed in Modelsim to verify the synchronization of data flow through the memory blocks and the functionality of the design. We have considered three sets of data with frequency content of zero (i.e DC), (fr/2) and (fr/4). The data is stored in ROM and is read out continuously to mimic the real time situation. We have designed the architecture to operate for 4 different orders of FFT

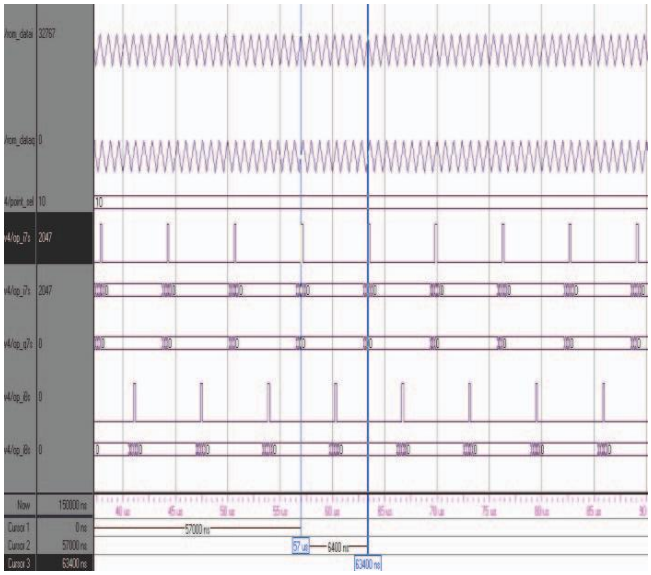

Figure 5. Output of 16 point FFT

The design responds to the stimulus very accurately. The results shown Fig.s '5' and '6' demonstrate the functionality. Fig. 5 is the snapshot of modelsim output for 16 point FFT with fr/4 input data. The plot shows the output of alternating buffers for explanation. Finally the output is combined into a single channel. The design peaks in the correct bin and the output is repeating consistently since the data is fed back in loop back mode. The design is working on 5 MHz clock which should give the time difference between two alternating outputs as 6.4 usec

$$t1 = [16 * 200\text{ ns} * 2] = 64\text{ us}$$

N * Clock Period * Alternating Output

which can be verified from the timing markers in the waveform output. The same data is tested with 64 point FFT and it can be seen the output between two alternating sets is exactly 256us

$$t2 = [64 * 200\text{ ns} * 2] = 256\text{ us}$$

N * Clock Period * Alternating Output

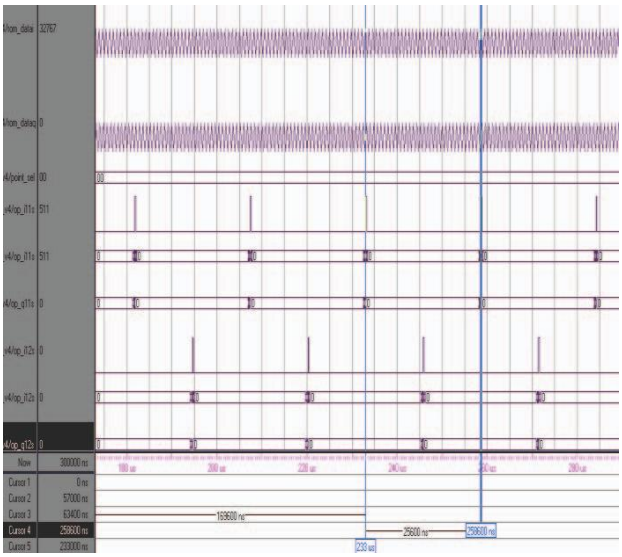

Figure 6. Output of 64 point FFT

and the output peaks in 16th filter in this case which is bit reversed order of 2nd filter for 6 bit bin address compared to
4th filter in Fig. 5 which is bit reversed order of 2nd filter for 4 bit bin address. The design takes care of all these changes when re conFig.d. The design is tested with different frequency input and with different orders changed online and is found to be functioning properly. After functional verification the design is synthesized in FPGA to verify hardware resource usage and timing constraints. The architecture consumes less hardware compared to the IP core and meets all the timing requirements as shown in Fig.s 7 and 8.

Quartus II Version	8.0 Build 215 05/29/2008 SJ Full Version
Revision Name	config_fft
Top-level Entity Name	fft_block_v5
Family	Stratix II
Device	EP2S90F1020I4
Timing Models	Final
Met timing requirements	Yes
Logic utilization	2 %
Combinational ALUTs	1,261 / 72,768 (2 %)
Dedicated logic registers	98 / 72,768 (< 1 %)
Total registers	98
Total pins	66 / 759 (9 %)
Total virtual pins	0
Total block memory bits	84,480 / 4,520,448 (2 %)
DSP block 9-bit elements	80 / 384 (21 %)
Total PLLs	1 / 12 (8 %)
Total DLLs	0 / 2 (0 %)

Figure 7. Resource usage of "configurable 64 point FFT " in Presented Design

Quartus II Version	8.0 Build 215 05/29/2008 SJ Full Version
Revision Name	fft64
Top-level Entity Name	fft_core_top
Family	Stratix II
Device	EP2S90F1020I4
Timing Models	Final
Met timing requirements	Yes
Logic utilization	5 %
Combinational ALUTs	1,954 / 72,768 (3 %)
Dedicated logic registers	3,417 / 72,768 (5 %)
Total registers	3417
Total pins	66 / 759 (9 %)
Total virtual pins	0
Total block memory bits	12,032 / 4,520,448 (< 1 %)
DSP block 9-bit elements	24 / 384 (6 %)
Total PLLs	1 / 12 (8 %)
Total DLLs	0 / 2 (0 %)

Figure 8. Resources for 64 point FFT IP core

V. CONCLUSION

In this paper we presented a simple architecture to perform FFT using pipeline of memory blocks. The architecture is configurable online and consumes less hardware. The design is scalable to any order depending on the requirement. The design can be conFig.d as required and optimized depending on the requirement because of the provision to access to twiddle factors.

REFERENCES

[1] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," Mathematics of Computation, Vol. 19, No. 90, 297-301, 1965.

[2] Chandrakanth V, Bhishm Tripathi, Wasim nasir, P jena, R kuloor , " Design and Implementation of a Highly Configurable Low Power Robust Signal Processor for Portable Ground Based "Multiple Scan Rate" Surveillance Radar" Proceedings of International Radar Symposium 2010.

[3] Richard G Lyons, "Understading Digital Signal Processing " second edition, Prentice Hall , March 2004

[4] Gonzalez and Woods, Digital Image Processing(2nd ed), 2002.

[5] W. Wolf, FPGA-Based System Design. Prentice Hall, 2004. [6]

www.altera.com

[7] Steve Plimpton, Gary Mastin, Dennis Ghiglia, "Synthetic Aperture Radar Image Processing on Parallel Supercomputers ".

[8] S R Degraaf, " SAR imaging via modern 2D spectral estimation methods" , in Proc of SPIE Optical Engineering and Aerospace sensing, Orlando, Florida, Apr 1994.

[9] P. Vaidyanathan, Multirate Systems and Filter Banks. Englewood Cliffs, NJ: Prentice-Hall, 1993.

[10] V.Gautam, KC Ray, Pauline Haddow, " Hardware efficient design of Variable Length FFT Processor" Proceedings of DDECS 2011.

[11] Chandrakanth V, Devendernath Ch, Shamnar A, " VLSI implementation Of Sliding window DFT" , Accepted in ICSIPR 2013.