On the Implementation of 128-Pt FFT/IFFT for High-Performance WPAN

Clare Huggett*, Koushik Maharatna*, Kolin Paul**
*University of Bristol, Bristol, UK
**Indian Institute of Technology, Delhi, India
Email: {ch9346, Koushik.Maharatna}@bristol.ac.uk; kolin@cse.iitd.ac.in

Abstract - **This paper deals with the efficient realization of a 128-pt FFT/IFFT processor for application in IEEE 802.15.3a standard. The 128-pt FFT/IFFT architecture has been designed by devolving it into one 8-pt and one 16-pt FFT. The 16-pt FFT was decomposed again and two separate 128-pt FFT algorithms have been developed, *viz.*, 8x4x4 and 8x2x8. Their relative merits and demerits have been analyzed from the algorithm as well as implementation point of view. The architectures have been prototyped on a Virtex II FPGA. The results indicate that the 8x2x8 architecture is better suited for the above mentioned purpose.**

I. INTRODUCTION

High performance Wireless Personal Area Networks (WPAN) is currently the focus of research and development. IEEE 802.15.3a standard is intended for the development of such a WPAN system with a target data rate of up to 480 Mbps. Although the standardization committee has not yet finalized the standard, the most popular proposal under consideration uses a Multi-band Orthogonal Frequency Division Multiplexing (OFDM) based physical layer (PHY) [1] implementation. FFT/IFFT is one of the most computationally intensive components in such an OFDM-based PHY implementation. In this paper we concentrate on the efficient implementation of the FFT/IFFT processor for the above mentioned standard.

Proposal [1] states that a 128-pt FFT/IFFT has to be performed within 312.5 ns. This massive computation rate can only be supported either at a very high frequency (achievable with the 90 nm technology) or using massive parallelism. However, since the target application is of a mobile and portable nature, the second approach may lead to significant power consumption which is in direct contradiction with the application goal. Thus, algorithm level reformulations as well as innovative design techniques have to be developed to simultaneously satisfy the power and timing constraints.

This paper reports a high-throughput low-power 16-bit fixed point 128-pt FFT/IFFT implemented on a FPGA. It may be noted that there are implementations [2] which report a higher throughput but they do so by having parallelism and duplicating functional units like multipliers. This of course gives rise to "big" circuits which consume unnecessary large amounts of power. Our contribution in this paper represents a "sequential" design which operates at the required frequency and is extremely low power. We achieve this by totally avoiding "complex multipliers" and replacing them by simple pipelined shift-and-add units which helps us to achieve high operating frequency at low power. We propose two different algorithmic reformulation and subsequent architectures for computing 128-pt FFT. The architectures are targeted towards 90 nm process. Since the process has just been announced, we choose to illustrate and prototype our systems on the Virtex II family [3] of FPGA which are built with a 130 nm process. We then scale the FPGA performance suitably to estimate the ASIC performance following certain previous comparisons [4] established in literature. The remainder of the paper is structured as follows: Section II deals with the algorithmic reformulation and comparison of the proposed methods with the existing approaches. In Section III the respective architectures have been described while Section IV provides the implementation details of the architectures on FPGA and compares their relative performances. We summarize our contribution and draw some conclusions in Section V.

II. THEORETICAL GROUNDWORK

The N-point Discrete Fourier Transform (DFT) of a complex data sequence B_k is defined by [5]

$$A(r) = \sum_{k=0}^{N-1} B(k) W_N^{rk} \qquad (1)$$

where r, $k \in \{0, 1, \ldots, N-1\}$ and $W_N = e^{-j2\pi/N}$, known as the twiddle factor.

Now considering $N = MT$, $r = s+Tt$ and $k = l+Mm$, equation (1) can be rewritten as

$$A(s+Tt) = \sum_{l=0}^{M-1} W_M^{lt} \left(W_{MT}^{sl} \sum_{m=0}^{T-1} B(l+Mm) W_T^{sm} \right) \qquad (2)$$

For the present problem considering $M = 8$ and $T = 16$ one may write

$$A(s+16t) = \sum_{l=0}^{7} W_8^{lt} \left(W_{128}^{sl} \sum_{m=0}^{15} B(l+8m) W_{16}^{sm} \right) \quad (3)$$

Equation (3) suggests that the 128-pt. DFT can be computed by first computing an 8-pt DFT on the appropriate data slot (described by equation (3)), then multiplying them by 105 non-trivial complex twiddle factors and computing the 16-pt DFT on the resultant data with appropriate data reordering.

The 16-pt DFT can be decomposed further using the same method. In the present work, we have explored two different types of decomposition of it *viz*., 4x4 and 2x8. The important point to be noted here is that for realization of the 8-pt DFT using a Decimation-In-Time (DIT) FFT algorithm, explicit multiplications are not required. The constants to be multiplied for the first two columns of the FFT flowgraph are either 1 or *j*. In the third column, the multiplication of $1/\sqrt{2}$ can be realized using hard-wired shift-and-add operations. Similarly, using a radix-4 FFT for realization of 4-pt DFT, one does not need explicit multiplication either. As shown in Table 1, using these decompositions, a significant reduction in the number of required complex multiplications can be achieved compared to the existing techniques.

Algorithm	Non-trivial complex multiplication
Radix-2	258
Radix-4	204
Split-Radix	186
8x4x4	168
8x8x2	152

Table1. Comparison of the multiplicative complexity of the proposed schemes with other existing schemes

Apart from the reduction of complex multiplications, another advantage of these decompositions is that of the 105 non-trivial twiddle factors required in the proposed schemes, only 16 are unique. This can be exploited to reduce the overall size of twiddle RAM which is one additional advantage compared to the other existing schemes.

The IFFT can be performed using the same structure. First, the real and the imaginary parts of the input have to be swapped and the computation proceeds as a forward FFT. At the output, the real and imaginary parts are swapped back and the output is scaled by 128.

III. ARCHITECTURE

The generalized architecture of the 128-pt FFT can be derived from equation (3) and is shown in Fig. 1. The processor consists of an input unit, an 8-pt FFT, a multiplier unit, a 16-pt FFT, a resorting unit, an output unit and a master control counter.

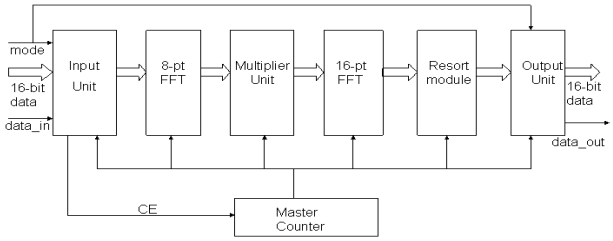

Fig. 1. Block diagram of the proposed FFT processor

A. Input Unit

The input unit consists of an input register bank having 16-bit wordlength that can store 113 complex samples. It is equipped with two single bit signals "*mode*" and "*data_in*". While the first one enables the processor to understand which mode of operation is needed (whether FFT or IFFT), the second one indicates the presence of valid data at the input. After the assertion of the *data_in* signal, serial data is inputted at every clock cycle at the 112^{th} position of the input register bank and at every clock cycle the complex data having index i is shifted to the $(i-1)^{th}$ position where $i \in \{1, .. 112\}$. The register bank is provided with eight pairs of 16-bit fixed wired outputs corresponding to the registers having index $8j$, where $j \in \{0, \ldots, 7\}$. When the input buffer is filled, the appropriate data (see equation (3)) gets self aligned with these hard-wired outputs and is delivered in parallel to the first 8-pt FFT unit. This operation is continued in every cycle. A 7-bit counter controls the serial input of the data in the register bank, which is enabled with the assertion of the *data_in* signal. When the 112^{th} position of the input register bank is full a signal *CE* is asserted to enable the master control counter. The subsequent operations are controlled by the master counter. This method of downward shifting of data in conjunction with its self-alignment with the hard-wired outputs reduces the number of multiplexed signals by a factor of 112 and 14 at the input and output of the input unit respectively. This massive reduction of wiring allows a more efficient layout of the processor.

B. 8-pt FFT Unit

It has been discussed in Section 2, that to implement the 8-pt FFT one does not need to use conventional digital multipliers. This fact opens up the scope to implement a fully parallel 8-pt FFT unit. In our implementation we realized a pipelined 8-pt FFT architecture following the 8-pt DIT FFT flowgraph where each complete column of the 8-pt FFT is computed in a single clock cycle. Thus, a complete 8-pt FFT can be computed in three clock cycles. The multiplications by the factor $1/\sqrt{2}$ in the third column are realized using hard-wired shift-and-add operations and thus require minimal hardware.

C. Multiplier Unit

It has already been mentioned in Section 2 that to carry out 105 non-trivial complex multiplications our architecture needs only 16 unique sets of twiddle factors. The complete operation can be carried out by changing the signs or swapping the real and imaginary coefficients of them. However, in the steady state, the data from the 8-pt FFT unit arrives at every clock cycle, consequently a single complex multiplier unit cannot be used to take the full speed advantage offered by the parallel 8-pt FFT. On the other hand, employing 8 complex multipliers will increase the area and power consumption of the entire system. However, since the twiddle factors used to multiply each of the incoming samples are known *a priori*, one can realize each of them using a simple shift-and-add technique. As an example, a constant 0.99895 can be decomposed to $1-2^{-10}-2^{-12}$ with 16-bit accuracy. Thus the multiplication of any input by this constant becomes a series of additions/subtractions of right shifted values of the input quantity. This approach has previously been used in [6] which resulted in a significant performance gain in terms of power consumption and area. Since the target architecture in the present case must run at a high frequency to satisfy the required data rate, we realized each of these 16 constants in a pipelined way and arranged them in parallel. The multiplier unit also has a temporary storage register bank that can hold 128 complex data. Again, hard-wired connections are used like the input unit and the data after multiplication enters the registers at locations 127 – 120. At every cycle, the data is down shifted as a block of eight until the register bank is full. This approach saves signal wire multiplexing by a significant amount. To deliver the data to the 16-pt FFT unit once again the hard-wiring and data self-alignment strategy used in the input unit has been employed and thereby substantially reducing the number of signal multiplexing again.

D. 16-pt FFT

Using the decomposition schemes proposed in Section 2, the 16-pt FFT module is realized using two different kinds of further decomposition which resulted in two different architectures *viz*., *Arch*1 (4x4 decomposition) and *Arch*2 (2x8 decomposition) for 128-pt FFT. *Arch*1 contains two 4-pt FFTs where each of the 4-pt FFT is realized using radix-4 butterfly. On the other hand, *Arch*2 contains a 2-pt and 8-pt FFT. Once again, the 8-pt FFT module is realized following the similar strategy described for the first 8-pt FFT module. Both 16-pt FFT architectures have a multiplier unit, which operates in a similar fashion to the main multiplier unit. Here only three pairs of unique constants are required. Once the data has been multiplied, it is put directly into a register bank able to hold 16 complex samples. The inputs to this block are hard-wired and the data is entered at either every 4^{th} / 8^{th} register depending on the architecture. At every cycle the data is down shifted until the buffer is full. At this point another data sample is ready to be input, so rather than hold-up this data a second 16 register buffer is used. The data is transferred to this buffer and is down shifted at every clock cycle in blocks of 4 (*Arch*1) or 8 (*Arch*2) until all data has been passed onto the next module.

E. Resort Unit

To make the reshuffling at the output to form the full 128-pt FFT simpler it was spilt into two stages. The resort module takes the data directly from the 2^{nd} FFT in the 16-pt FFT module and reorders it to form a set of 16 data. This is done by hardwiring the output of the second FFT to every $4^{th}/2^{nd}$ register and down shifting it at every clock cycle.

F. Output Unit

The output has a dual structure of the input unit. It consists of a complex bank of 128 registers and a 7-bit counter. The data is hard-wired into this register bank at every 8^{th} register position and is down shifted at every clock cycle until the buffer is full. Once full, the buffer is emptied from the 0^{th} register while continuing down shifting of the data until it is empty. When the first sample is outputted the signal *data_out* is asserted high and is held high until the buffer is exhausted. This indicates valid data at the output. If the IFFT has to be performed, the real and imaginary parts of the data are swapped and are shifted to the right by 7 bits (scaling by 128). As with the input module this is also indicated by the signal *mode*.

G. Master Control Counter

The 7-bit master counter is responsible for synchronizing the entire system. It is turned on with the assertion of the signal *CE* and it stops counting when the last set of data has reached to the output buffer. The master counter is used to generate an enable signal for the modules at different time instants. This easily translates to implement clock gating in the entire circuitry with no additional design effort.

IV. IMPLEMENTATION AND PERFORMANCE EVALUATION

The architectures described in the previous section have been coded in VHDL. The parallel-to-parallel 128-pt FFT is performed in 73 and 104 clock cycles for *Arch*1 and *Arch*2 respectively. The architectures were synthesized using the Xilinx tool chain. The target board was Alphadata ADM-XRC II which has a Virtex II 6000 series FPGA. The Virtex II FPGA represents a product from 130 nm technology. The results of post place and route are shown in Table 2. It is clear that the second architecture is more area and power efficient while runs almost at the same frequency as the first one. The number of clock cycles required for *Arch*2 can be brought down from 104 to 56 by simply duplicating the 2-pt FFT butterfly structure

four times in the 16-pt FFT module. Since the basic 2-pt FFT is very simple, this added parallelism will result in a minimal hardware overhead. The maximum operating frequency required to satisfy the goal can be tailored further by adding another degree of parallelism with minimum area overhead.

Parameters	***Arch*1**	***Arch*2**
# of slices	20751	20580
# of slice Flip Flops	22437	23171
# of 4-input LUTs	32604	32245
# of IO	67	67
Clock load	23203	23905
Maximum frequency (MHz)	63	62
Power Estimate (mW)	712	337

Table2. Comparison of *Arch*1 and *Arch*2 implementation

The maximum operating frequency is limited by the routing resources available in the FPGA. The critical paths have up to 25 – 37 % of routing involved and hence limit the highest frequency achievable.

When implemented on a same process technology, it has been reported [4] that the ASIC performance scales up to 2 - 2.5 times compared to an FPGA implementation. If we scale the performance to a higher process technology (90 nm), the throughput is expected to be about at least 4 times. Hence we expect that our proposed architectures can easily meet the performance requirement of computing the 128-pt FFT in 312.5 ns in 90 nm process (ASIC).

During implementation, we optimized our architectures by removing the set-reset logic for many of the internal Flip-Flops of the computation blocks. This resulted in area savings to a great amount (1:7). The 16 unique twiddle factors are synthesized in a LUT and thus, avoiding the requirement of RAM. The floorplan of *Arch*2 is shown in Fig. 2.

The power estimates, using XPower, are also presented in Table 2. While it is impossible to accurately scale exact power dissipation figures for ASIC, from design experience these estimates show that if implemented in 90 nm technology we can expect the architectures to dissipate at least about 1/4 of the power. Furthermore, each of the sub blocks operates at a precise value of the control counter. Thus, significant power reduction is further possible by employing clock gating while implementing the proposed architectures in ASIC.

V. CONCLUSION

Two potential architectures for realizing 128-pt FFT/IFFT processor for IEEE 802.15.3a standard have been proposed and implemented on a Virtex II FPGA. Though it is natural that the FPGA implementation gives much lower maximum frequency of operation compared to the ASIC implementation, it sets a baseline for choosing an appropriate architecture. In our implementation it is found that the architecture based on 8x2x8 decomposition of the 128-pt FFT performs better in terms of area compared to the architecture based on 8x4x4 decomposition while both of them give significant reduction of algorithmic complexity compared to the existing schemes.

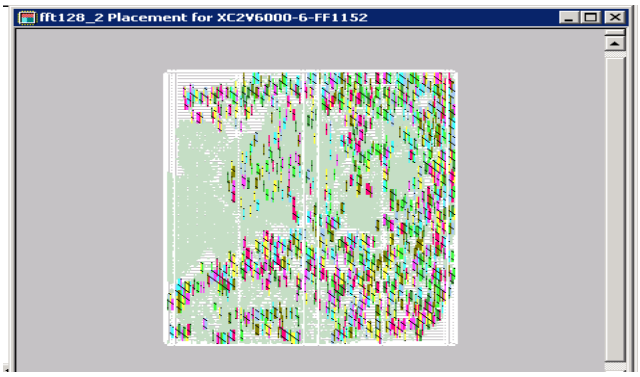

Fig. 2: Floorplan of *Arch*2

In terms of power, *Arch*2 shows approximately 52% less power consumption than that of *Arch*1. The number of required clock cycles of *Arch*2 can be brought down further by adding another degree of parallelism in its 16-pt FFT module with very nominal area overhead. We expect that when implemented in the form of ASIC using 90 nm technology, both of the architectures will satisfy the timing constraint and *Arch*2 will consume much less power compared to the *Arch*1 and thus is a better choice.

REFERENCES

[1] http://grouper.ieee.org/ groups/802/15/pub/2003/ May03/ 03142r1P802-15_TG3a-TI-CFP-Document.doc

[2] I. S. Uzun and A. A. Bouridane, "FPGA Implementations of Fast Fourier Transforms for Real-Time Signals and Image Processing", *Proceedings of IEEE International Conference on Field-Programmable Technology (FPT)*, 2003.

[3] http://www.xilinx.com/virtex2

[4] R. J. Peterson and B. L. Hutchings, "An assessment of the suitability of FPGA-Based system design for use in DSP", *5th Intl Workshop on FPL and Application*, Oxford, England August 1995.

[5] A. M. Despain, "Very fast Fourier transform algorithms hardware for implementation", *IEEE Trans. Comput.*, vol. C-28, no. 5, pp. 333 – 341, May 1979.

[6] K. Maharatna, E. Grass and U. Jagdhold, "A 64-point Fourier transform chip for high-speed Wireless LAN application using OFDM", *IEEE J. Solid-State Circuits*, vol. 39, no. 3, pp. 484 – 493, March 2004.