ON THE HARDWARE DESIGN OF FRONT-END PROCESSING IN THE SDR SYSTEMS

Najam-ul-Islam Muhammad, Raymond Knopp (Institute Eurecom, Sophia-Antipolis - France,
{muhamman, Raymond.Knopp}@eurecom.fr);
Renaud Pacalet (TELECOM ParisTech, Sophia-Antipolis - France,
Renaud.Pacalet@TELECOM-ParisTech.fr)

***Abstract*— A hardware accelerator capable of carrying out the air-interface processing inside a baseband architecture is presented. The architecture is not only flexible to adopt to the different environments but also carries out the operations quite efficiently. The memory scheme to realize the different set of operations with the same organization is also discussed along with its management by the proposed processor. The performance analysis with respect to the current target technology (FPGA) is described along with a critical analysis describing the limitations of the designed air-interface processor.**

I. INTRODUCTION

The emergence of different applications working at different frequencies and using different mechanisms was one of the key reasons leading to Software Defined Radio (SDR) based systems. The proliferation of wireless communication standards has made sure that the SDR systems are an integral part of wireless communication world. The daily life wireless appliances are on the way of adding multiple services on the single device, thus integrating more and more standards to the already wide range. The mobile devices today and in future have to cater the different air-interfaces ranging from 2G to 3GPP LTE standards. This, in turn, requires the need of flexible architectures that can cater the increasing demands in an efficient manner.

In this article, we first discuss the proposed flexible baseband prototype plateform by Institute Eurecom in collaboration with TELECOM ParisTech. We focus on the air-interface processing inside the baseband architecture, and explain the design approach and the flexibility our design offers for variety of operations. The generic parameterizable baseband board is aimed to address the SDR applications and fulfills the processing requirements of 2G, 3G, 4G, broadcast communication and wireless LAN standards. The goal of the baseband prototype architecture is to investigate the different possible solutions for the SDR applications, hence the target technology chosen is FPGAs. This reduces the design cycle and cost, and provides the flexibility as well. Later on, the finalized architecture will be adapted to the System on Chip technology.

The baseband architecture is composed of two FPGAs; a high level control module (the Interface and Control FPGA) and a digital signal processing engine (Processing Engine FPGA). The designed baseband prototype is shown in figure 1 [10]. Both the FPGAs are Xilinx Virtex-5; the control FPGA is LX110T while the DSP Processing Engine FPGA is LX330 [9]. The control module is based on a SPARC CPU (LEON3 from Gaisler Research), and it manages the processing engines in the other FPGA.

The architecture is designed by assigning a set of similar tasks to different hardware accelerators; each of these accelerators is designed independently while considering the requirements of all the wireless communications standards. For example, the channel decoder takes care of trellis-based decoding algorithms: namely Viterbi and Turbo; and the design is in compliance with the wireless communications standards including: IEEE 802.11a/g (WLAN), IEEE 802.16 (WiMAX), 3GPP UMTS and 3GPP UMTS-LTE. The design of each hardware accelerator or Intellectual Property (IP) follows a generic pattern for the benefits such as: Ease of communication among the IPs, Coherence of the design across the baseband board, Facilitate the debugging process, and addition of the functionalities in future etc.

The nucleus of this article is the design, internal architecture and performance analysis of one of the IPs of our baseband design; the Front End Processor (FEP). The FEP is responsible for air-interface level computations in the SDR systems. The different air-interfaces for the SDR applications include: Orthogonal frequency-division multiplexing/multiple-access (OFDM/A), Single Carrier FDMA (SC-FDMA), Space-division multiple access (SDMA), and Wideband Code Division Multiple Access (W-CDMA). The set of operations to be performed at these air-interface are: Channel Estimation, Data Detection, Synchronization, and Carrier Phase Offset Estimation etc.

The approach adopted by us for the hardware design of the FEP is based on frequency domain computations. The Discrete Fourier Transform (DFT) based approach for different individual air-interface systems has been well documented. The receiver architectures based on the DFT have been proposed for OFDM based systems [2], [3], while the DFT based receivers for HSDPA [4], WCDMA [5] and GSM [6] can also be found in the literature.

In the following section, we describe the Front End Processor, then move to its functional specifications in section-3. The memory subsystem of the FEP is discussed in setion 4, before presenting a performance analysis of the IP in the last section.

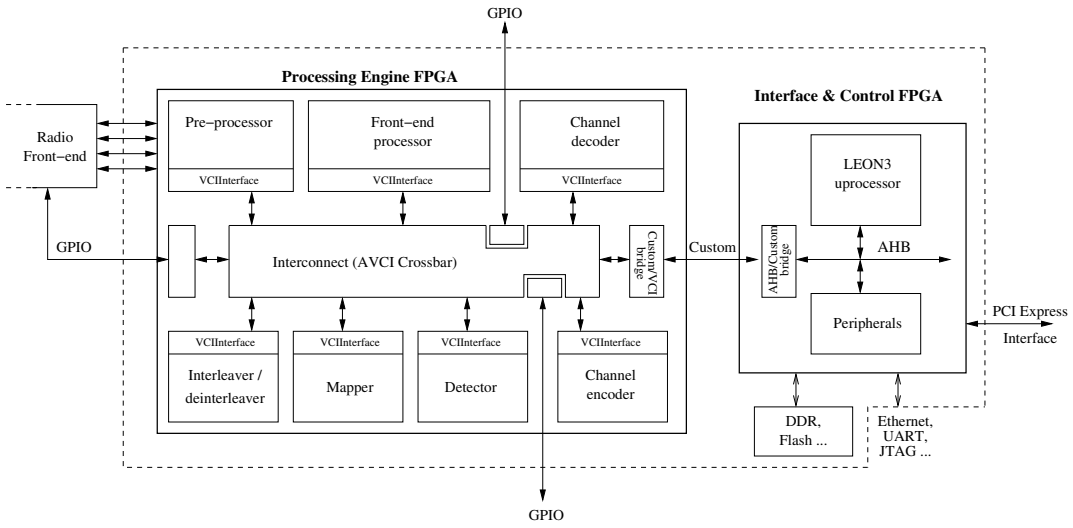

Fig. 1. Baseband Processing Architectural Overview

II. Front End Processor

Our design is an effort to build a processing block for all the different air-interfaces based on the DFT. The processor is highly flexible, parameterizable and caters all the different operations for the standards discussed in the previous section. The FEP itself is composed of different macro computational blocks, which were pointed out during the analysis of the air-interfaces. These macro blocks perform all the tasks at the air-interface of the SDR system. The functions are:

- Discrete Fourier Transform (DFT), Inverse DFT (IDFT)
- Operations over sub-band level (Energy Calculations, Maximum, arg-max Calculations, Dot Product)
- Component-wise-Operations (addition, product, division)

The IP structure of the FEP conforms to the generic structure described in [10]. It consists of the VCI-Interface, the micro-controller (6502), the DMA engine, the FEP core, and FEP Memory Subsystem. The IP is controlled by the micro-controller, the local processor also performs low level transactions among IPs, transfers data inside IP, and processes commands. The VCI Interface is used for the communication between the IP and the VCI compliant Interconnect (AVCI Crossbar) [8] shown in figure 1. The DMA engine is responsible for data trafficking between the IP and the outside world, i.e. other IPs and the Global Memory. The Memory Subsystem of the FEP is described later in the article along with its interface with the FEP Core. An overview of the IP shell is depicted in figure 2. The FEP module is controlled by the main processor (LEON3), which passes the command words to the VCI-Interface, main processor governs the local micro-controller (6502) and has access to memory subsystem.

III. FEP Functional Specifications

A. DFT

The Discrete Fourier Transform (DFT) supported by the FEP is for the input vector sizes of

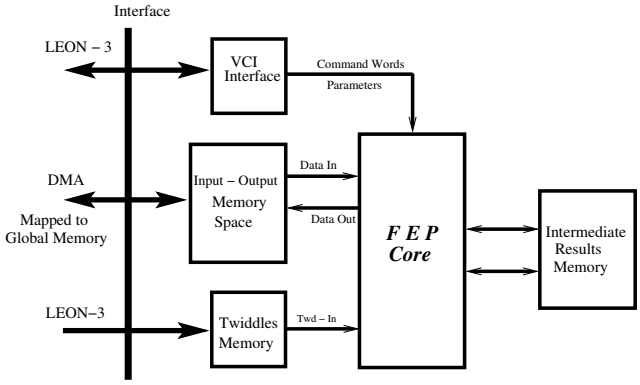

Fig. 2. FEP Block Diagram

$\{8, 16, 32, 64, 128, 256, 512, 1024, 4096\}$. This is based on the analysis of requirements for different standards in question. The DFT of an input vector X is given by:

$$DFT_N(X[k]) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} X[n].e^{-\frac{2\pi jnk}{N}}, k \in [0, N-1] \quad (1)$$

In the FEP, the Inverse Discrete Fourier Transform (IDFT) is calculated by $IDFT(Y) = \overline{DFT(\overline{Y})}$. This leads to addition of conjugate function in the FEP, which is used for few other operations as well. As all the input vector sizes for DFT are power-of-2, and few are power-of-4 as well; the famous Radix-4 algorithm and the Split-Radix algorithm are utilized. The algorithms are based on butterfly operation and run over data samples for multiple number of stages. The number of stages is based on the input vector size. The number of stages for an input vector size of X are $log_4 X$ when input vector size is pow-of-4, and number of stages are $log_4 X/2 + 1$ when input vector size is power-of-2 but not power-of-4. The split-radix algorithm in one stage, either first or last, uses the Radix-2

butterfly algorithm. The input and output data is represented by 32 bits with both real and imaginary parts in $Q1.15$ format, while the intermediate results are stored in 50 bits with real and imaginary parts in $Q9.15$ format [7].

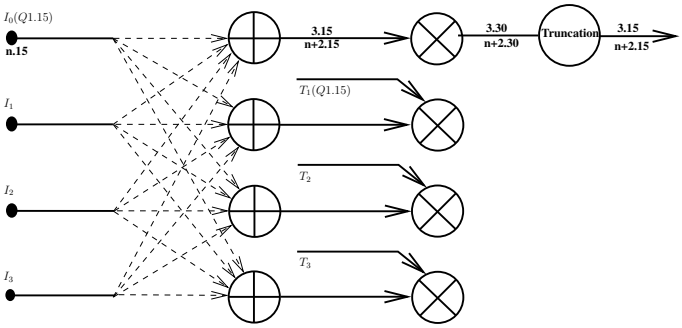

Fig. 3. Basic Radix-4 Operation

The figure 3 depicts the radix-4 butterfly operations. The four complex input vectors (I_0 to I_3) are first added in different combinations before being multiplied by twiddle factors (T_1 to T_3, T_0 is always 1). The twiddle factors are roots of unity circle and stored in the FEP memory. From the figure 3, it is evident that the addition operation over '4' complex vectors adds up '2' significant bits for each resultant, and a multiplication adds another '15' bits. The '15' LSBs of the multiplication result can be discarded without much loss of the accuracy. The truncation operation at the end limits each element of butterfly to be $Q3.15$, if input was $Q1.15$. Thus during the intermediate stages of the DFT/IDFT operation, the size of the data samples increases by '2' bits in each stage. Considering the maximum input vector size and the simulation results for a low bit error rates, the intermediate results of the DFT operation are stored as '50' bits with real and imaginary part as '25' bits [7].

The global throughput for the FEP block is set to be at least 1-sample/cycle, the processing requirements inside the FEP are based on the intended throughput and the maximum input vector size. This results in the processing of 8 samples per cycle for DFT/IDFT operations i.e. both reading 8-samples and writing 8-samples in one clock cycle. This implies that in our case two butterfly operations are to be performed in one clock cycle. The processing unit of the DFT block (with one butterfly operation) is shown in figure 4. The matrix operation is explained in the memory subsystem section of the paper.

B. Operations over sub-band level

The sub-band level operations include Energy Calculations, Maximum, arg-max Calculations, Dot Product over input vector(s) with a maximum input size of 4096. The sub-band operation over a vector is described by the number of sub-bands n_{sb}, size of each sub-band s_{sb}, and starting index of first sub-band inside the vector a. The FEP also supports the operations inside sub-band over skipped input samples (sub-sampled operation with sub-sampling factor m) , which for example may be used for pilots that appear in the symbols spaced at a particular distance. The functions are described in the following equations.

$$E_{sb}(U) = \frac{1}{s_{sb}} \sum_{i=0}^{s_{sb}-1} |U[i]|^2 \quad (2)$$

$$E(U) = E_0(U)E_1(U)\dots E_{n_{sb}-1}(U) \quad (3)$$

$$\max(U_{sb}) = \max_{0\le i\le s_{sb}} (|U_{sb}[i]|^2) \quad (4)$$

$$D_{sb} = U.V = \sum_{i=0}^{s_{sb}} U[mi+a] \times V[mi+a] \quad (5)$$

The equations describe the functionality of each of the macro block, the results over consecutive sub-bands are stored in the memory space contiguously as shown for energy calculations in equation 3.. The complex multiplications inside each of sub-band operations results in an increase of output vector size, and all the results are stored in sign-extended $Q17.15$ format to cater the maximum possible bit additions to the resultant. For each complex multiplication in these operation, the '15' LSBs are discarded without much loss of the accuracy.

C. Component-wise Operations

The component-wise operations include the four basic arithmetic operations between the two input vectors, i.e. addition, subtraction, multiplication, and division. The component-wise product and division blocks can be operated between a vector and a scalar. The component-wise division is performed by using the product module and look up table (LUT) and is given by:

$$U \div V = U[i] * \hat{V}[i] \approx U[i] \div V[i] \quad (6)$$

$\hat{V}[i]$ represents the inverse value of $V[i]$ stored in LUT. The results for all the component-wise operations are stored back in 32 bits with real and imaginary parts in $Q2.14$ format. This causes a truncation of '16' LSBs for product operations, and '1' bit in case of addition operations.

The FEP micro-controller takes care of chain of operations to be performed inside the FEP. To illustrate how the higher level functions are implemented in the FEP, the process chain for normalized channel estimation in the single antenna for OFDM based systems is shown in the figure 5. In the figure, p_{ref} and p_{rec} represent the reference and received pilot symbols respectively, $\odot$ represents the component-wise-product between the two vectors, $|\ |^2$ represents the Energy calculation module of the FEP, and $\div$ is the division of a vector by a scalar quantity.

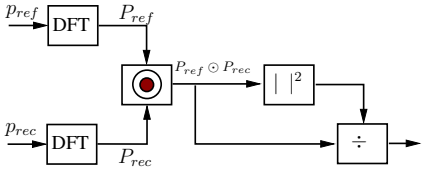

Fig. 5. Process Chain of Channel Estimation for OFDM based System

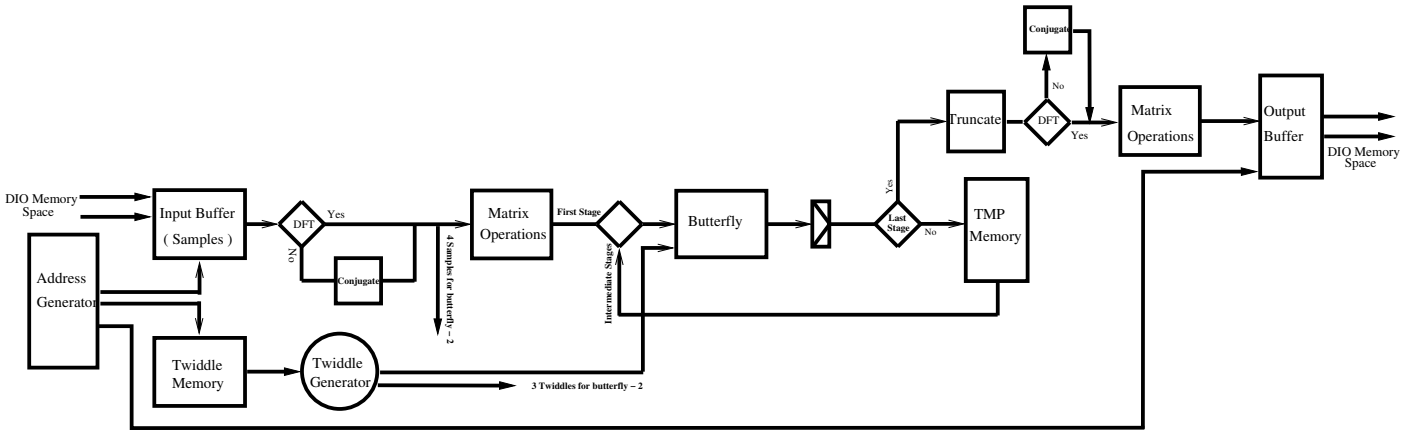

Fig. 4. The DFT / IDFT Processing Unit

IV. Memory Subsystem (MSS)

The internal memory of the FEP is used to store the input data, intermediate and final results, and the twiddle factors. From the external point of view it is a contiguous memory space accessible to all the peripherals. The memory is accessed by the VCI-Interface, DMA engine, micro-controller and by the IP core itself; hence it is a shared memory. The core is always given the highest priority followed by 6502, the DMA engine and the VCI-Interface. It is the responsibility of the control software to take care of data / result over-writings by any of the peripheries accessing the memory subsystem.

The FEP memory subsystem is composed of three main chunks, namely:

1) Input - Output data space (DIO)
2) Twiddle factors memory space (TWD)
3) Internal data processing memory space (TMP)

The FEP memory sub-system is composed of ram blocks `RAMB36`, `RAMB18` (Configurable Synchronous True Dual Port Block RAMs) available in *XilinxVirtex 5* FPGA devices [9] each of size 36 k-bits and 18 k-bits respectively. For FEP utility, these are configured as a 32-bit wide by 1-K deep (using one `RAMB36`) and 54-bit wide by 1-K deep (using one `RAMB36` and one `RAMB18`) true dual port RAMs.

The size of each of three areas of the FEP memory subsystem is based on input-output data storage, intermediate results storage, and the number of memory-accesses and processing per cycle (i.e. the over-all performance requirement of the processing block) for all the computationally different operations. The input-output (DIO) memory space is allocated in such a manner that two successive tasks for IP-core can be processed without any delay / lag between them. The memory requirement for component-wise-operations - CWO (addition, product, division, subtraction) is more than any other operation in FEP block as it requires 2 input, and 1 output vectors to be stored.

- Maximum input vector size = 4096 samples
- Memory requirement for 2 Input vectors = $2 \times 4096 \times 32 = 256$ k-bits
- Memory requirement for the output vector = 128 k-bits
- Memory requirement for one CWO = $256 + 128 = 384$ k-bits
- Total memory requirement for two CWO operations = $2 \times 384 = 764$ k-bits

Some other results like energy, maximum and dot-products over sub-bands may be required over multiple operations. Keeping this in view, we assign a total of 1 M-bits of memory for input-out data space. This is accomplished by using 32 memory blocks of `RAMB36`. The 8-bit micro-controller has an access to almost one half of this memory (byte by byte).

The size for Internal data processing memory space (called TMP) is based on the intermediate computation during different operations like DFT, Dot-product, Energy / Max calculations etc. We take the most computation intensive task i.e. DFT for maximum input vector size as the basis to decide the size of internal memory. As per algorithmic implementation of our DFT scheme, the intermediate samples (results) are stored as 50 bits instead of 32 bits (which is the case for input and output samples). The DFT implementation is a pipeline based to meet the throughput and delay requirements; thus few intermediate results are being read while some others are being written in the same clock cycle. This dictates to reserve twice the maximum intermediate samples size memory for internal results / processing. We make use of the 4 parity bits of `RAMB36` and the 2 of `RAMB18` to achieve a total efficient memory subsystem size. The 4 MSBs of the 54 bits words are not used. Therefore, each of the intermediate results of 50 bits is stored using one `RAMB36` and one `RAMB18` simultaneously.

- Maximum input vector size for DFT = 4096
- Intermediate Results Memory requirement = $2 \times 4096 \times 50 = 400$ k-bits

Thus, a total of 8 blocks each of `RAMB36` and `RAMB18` are used for Internal data processing memory.

The 'N' twiddle factors, being the roots of unity, can be computed from 'N/8' twiddle factors. Using this property of

twiddle factors, a good amount of memory space is saved. One by eight of maximum size DFT i.e. 512 twiddle factors are stored inside FEP memory. These twiddle factors are written every time FEP is initialized. The DFT operation with two butterflies requires access of '6' twiddle factors per cycle, however the implementation scheme of DFT requires an access of 3 twiddle factors in each cycle. The rest of the twiddles are calculated using the 'twiddle generator' (shown in figure 4). The twiddle factors are stored (duplicated) in three different physical rams `RAMB36` with one port access to IP-core for twiddle factors while the other port and one-half of the memory space for future usage and/or storing some vectors during debugging of IP-core. Twiddle factor memory is written by VCI-Interface and IP core can only read from it. The twiddle factors are stored either in the upper half or the lower half of all three `RAMB36` blocks, and this information is passed in the command word for DFT operations to facilitate the internal address calculations. Memory requirement for Twiddle factors = $3 \times 1024 \times 32 = 96$ k-bits. Thus total size of memory sub-system = $1024 + 400 + 96 = 1520$ k-bits = 190 k-bytes, and physically 41 blocks `RAMB36` and 8 blocks of `RAMB18` are used.

The size of the memory subsystem is a bottleneck for the future ASIC proposal. For the ASIC development, we propose to move the big chunk of the memory as external Double Data Rate Dynamic Random Access Memory (DDR DRAM).

A. Memory Access Scheme

The IP core reads out and writes in the DIO area. The DIO area is implemented as a four rows times eight columns array of 1k×32 RAMs. In the following, columns are numbered from 0 for the leftmost to 7 for the rightmost and rows are numbered from 0 for the top to 3 for the bottom. IP core accesses DIO through eight read-write channels, plus a mode indicator. The mode indicator selects between two operating modes:

- Fourier Transform mode - FT
- Pre-Post Processing Mode - PP (All Operations except FT)

When in FT mode the eight read-write channels (between the IP-Core and MSS) are configured as eight read and eight write channels, one read-write pair per column. The addresses are 32 bits words addresses in the column, that is 12 bits (4k) only. The unused MSBs are discarded. The read addresses of the eight columns are considered as equal. Only the read address of the first channel is used, the others are ignored. The write addresses of the eight columns are considered as equal. Only the write address of the first channel is used, the others are ignored. The read and the write address are not necessarily equal. Address 0 points to the topmost 32 bits word of the column, that is, the first 32 bits word of the top RAM of the column (the RAM in row number 0). Address 4095 (`0xfff`) points to the bottom 32 bits word of the column, that is, the last 32 bits word of the bottom RAM of the column (the RAM in row number 3).

When in PP mode the eight read-write channels are configured as eight read or write channels, two per row. Each channel can be used either to read or to write but not both (exclusive, read-xor-write). The read or write addresses are 32 bits words addresses in the row, that is 13 bits (8k).

Thanks to these two exclusive modes and to the fact that there are at most one read and one write channel per column (in FT mode) or two read-xor-write channels per row (in PP mode), there are at most two accesses per RAM in the DIO area. So, the two ports of the *Xilinx* block RAMs are sufficient for the IO needs.

The IP core reads and writes in the TMP area through eight read-write channels, one per 1k×50 RAM. Read and write operations are independent and have independent read and write enables and independent read and write addresses, 10 bits each (1k).

The IP core accesses the TWD area in read-only mode. It reads in each of the three 1k×32 RAMs through three dedicated read channels, one per RAM, wired to the A port of the RAMs. IP core cannot write in this area. The addresses IP core generates are 10 bits addresses (1k range).

This memory view is summarized in figure 6.

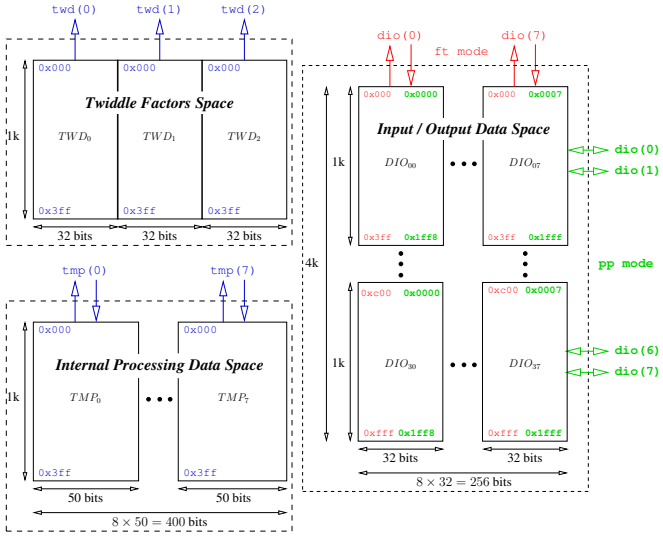

Fig. 6. The memory layout from IP core point of view

B. The Matrix Operations

The DMA engine always writes in and reads out from the FEP memory space in continuous order, and the address space access is independent of the input or output vector sizes. The variation in the operated vector sizes leads to some complexity of memory access in the DIO memory space. In case of DFT computation, the indices of samples accessed by butterfly are input-vector-size dependent, which in FEP case ranges from 8 to 4096. The indices accessed are spaced 'N/4' from each other. This, in turn, means the arrangement of input / output samples for all the possible input vector sizes in such a manner that 8 samples can be operated in one clock cycle. The IP core reads / writes eight samples per cycle in DIO, in FT

mode. These eight samples are eight consecutive components of the input/output vector. It uses a small internal cache to reorder the input samples before feeding them into its two radix-4 units, called the matrix operation. It utilizes simple 4×4 matrix (with each element as 32-bit sample) for each of the two butterflies. With the help of address generation schemes, the samples accessed are first written by IP-Core in row-wise fashion and read by the butterfly in column-wise fashion. The row-column access pattern is reversed after every 4-cycles. The similar operation starting in reverse order is also followed at the output end of the DFT macro-block. The internal cache can also be considered as few 32-bit registers and it results in avoiding the usage of any complex sample access schemes.

V. Performance Analysis

A. Results

Using the Xilinx Virtex-5 FPGA, $30DSP48E$ slices are used. This makes 16% of the total available in the FPGA, which is quite good considering the fact that the FEP is one of the most computation intensive IP of the baseband processor. The maximum achievable frequency for DFT operations is 135 M Hz, and is quite acceptable considering the throughput of the block. The number of cycles spent to calculate the different input vector sizes of the DFT are shown in the table I. The implementation of the DFT macro-block is pipelined to achieve the higher throughput and eventually higher data rates. The higher values of cycles used for the smaller pow-of-2 input sizes is due to the fact that in the Split-Radix algorithm the power-of-2 input size requires $log_4 X/2 + 1$ stages for an input vector size of X. However for the larger values, the IP performs quite better than the anticipated throughput of 1-sample-per-cycle. The throughput for the component-wise-operation and sub-band-level-operations is 2-samples-per-cycle, and is achieved in the implementation comfortably. The maximum achieveable frequency for these modules is around $150MHz$, and the number of complex multipliers used is 12.

DFT Size	# of Cycles	DFT Size	# of Cycles
8	20	16	18
32	46	64	60
128	107	256	174
512	372	1024	695
2048	1597	4096	3136

TABLE I
DFT : Number of cycles used for different Input Vector Sizes

B. Limitations of the Architecture

Though the FEP meets the functional specifications of its design and also achieves good performace; there are few limitations to its design and are listed here:

The matrix operation, small cache algorithm described earlier, puts a condition on the way the input / output vector is written / read in the DIO memory area by a DMA transfer or by a direct access through the VCIInterface: the starting sample (32 bits) address of the vector in the DIO area must be a multiple of 8. As a consequence the starting 64 bits word address used by the DMA engine or by the VCIInterface must be a multiple of 4. Symmetrically, when reading an output vector from the DIO area, the sample starting address will always be a multiple of 8. This doesn't require any specific requirement at the higher control level, but still the software has to take care of it while assigning the input and output addresses to the FEP core.

The FEP functions are mutually exclusive i.e. only one set of macro-blocks can run at the same time. e.g. In the channel estimation of OFDM systems, the component-wise product can only start once the DFT has terminated. However, this was foreseen while designing the IP and it doesn't cause any performance degradation in the overall baseband receiver design thanks to the MSS design and higher throughput of the IP. It is also worth mentioning that there is no lag or delay between the start of a task at the termination of the previous task. The new tasks (commands) can always be written in the IP and the memory space can be filled in for next task, while the current task is in progress.

VI. Conclusions

The presented Front End Processing block is capable of processing all the operations at the air-interface of all the wireless schemes. The designed hardware is flexible and the resource utilization is quite good. The maximum achieveable frequency is good enough to process the evolving applications as well. The future tasks include, the optimization of design to achieve higher frequency, benchmarking with some dedicated existing solution, and comparison of design approaches such as tool-based ASIP designs.

References

[1] J. Mitola, "The Software Radio Architecture", IEEE Commun. Mag., vol. 33, no. 5, May 1995, pp. 26-38.

[2] Yu-Wei Lin and Chen-Yi Lee, "Design of an FFT/IFFT Processor for MIMO OFDM Systems", IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS: REGULAR PAPERS, VOL. 54, NO.4, APR 2007

[3] Wei-Hsin Chang, and Truong Nguyen, "An OFDM-Specified Lossless FFT Architecture", IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS: REGULAR PAPERS, VOL. 53, NO. 6, JUNE 2006.

[4] D. Lo Iacono, J. Zory, E. Messina, and N. Piazzese, "Block processing engine for high-throughput wireless communications", 2nd International Symposium on Wireless Communication Systems, 2005.

[5] Y. Guo, J. Zhang, D. McCain, and J. R. Cavallaro, "An Efficient CirculantMIMO Equalizer for CDMA Downlink: Algorithm and VLSI Architecture", EURASIP Journal on Applied Signal Processing 2006

[6] Christophe Laot, Raphaël Le Bidan, "Low-Complexity MMSE Turbo Equalization: A Possible Solution for EDGE", IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, VOL. 4, NO. 3, MAY 2005.

[7] N.I. Muhammad, K. Khalfallah, R. Knopp, R. Pacalet, "Reconfigurable DSP Architectures for SDR Applications", 14th IEEE International Conference on Electronics, Circuits and Systems (ICECS), 2007

[8] Consortium VSIA http://www.vsia.org/

[9] Virtex-5 User Guide available at http://www.xilinx.com

[10] N.I. Muhammad, R. Rasheed, R. Pacalet, R. Knopp, K. Khalfallah, "Flexible Baseband Architectures for Future Wireless Systems", 11th EUROMICRO CONFERENCE on DIGITAL SYSTEM DESIGN (DSD) Architectures, Methods and Tools, 2008