Reconfigurable DSP Architectures for SDR Applications

Najam-ul-Islam Muhammad[†], Karim Khalfallah[†], Raymond Knopp[†], and Renaud Pacalet[‡]
[†]Institute Eurecom , 2229 routes des Cretes, F-06904 Sophia-Antipolis Cedex - France
{firstname.lastname}@eurecom.com
[‡]ENST, Institute Eurecom, 2229 routes des Cretes, F-06904 Sophia-Antipolis Cedex - France
{Renaud.Pacalet}@enst.fr

Abstract— **Addition of new bands, modes and services in mobile devices means the use of software defined radio (SDR) in the upcoming wireless devices is inevitable. To realize SDR, a common flexible hardware platform is required that supports transmission and reception of the existing radio access technologies. Air-interface and analog/digital conversions make a key part of SDR hardware both on transmitter and receiver. Front end processing block (FEP) of SDR consists of processing blocks required at the air-interface including Time/Frequency conversions, Dot-Products, Energy and Max calculations. We propose a flexible yet efficient hardware design for FEP that will furnish the front-end-functionality requirements of 2G, 3G, 4G, broadcast communication and wireless-LAN standards.**

I. Introduction

The current trend of convergence between communication and information systems contributes not only to introduction of new telecommunication products with ever-increasing functionalities, but also to the integration of several means of communication in the same system. Even a standard PC, now-a-days, has facilities like Blue-tooth connection, several standards of high-rate local area network (IEEE 802.11a, b and g), and the possibility of integrating a 2G/3G card. The different types of applications and usages have led to development of different standards being used in wireless communications systems. Though these systems have almost the same functional blocks, however the way these blocks function differs greatly from standard to standard. In wireless communication systems, radio spectrum, radio access technologies and protocol stacks vary from system to system and network to network. Moreover, the evolution of new standards has not stopped and there are no signs of it in the near future, rather there exist incompatible network technologies. These issues give rise to the need for a global design of the system that can handle most of the existing communication devices (using different wireless communication standards) that exist, if not all. The obvious choice is to have a flexible hardware architecture, that will replace existing dedicated structures for each application. This sums up the concept of Software Defined Radio (SDR). By a broad definition, SDR is a reconfigurable radio communication system that can be tuned to any frequency band, and can handle all the modulation schemes in a wide frequency range; thus serving multiple services and communication protocols [1]. SDR is controlled by software, while the programmable / reconfigurable hardware performs the mentioned tasks. Software defined radio and reconfigurability can be defined in many ways, and it is always dependent on the functional support these are providing. So before heading towards our design, we would like to define these two terms in functional design perspective. SDR, that we are addressing is capable of implementing 2G, 3G, 4G, broadcast communication and wireless LAN standards using the same HW/SW architecture. Reconfigurability denotes the capability of a system that can dynamically change its behavior, usually in response to dynamic changes in its environment. However, in the context of wireless communication reconfigurability tackles the changeable behavior of wireless networks and associated equipment, specially in the fields of radio spectrum, radio access technologies, protocol stacks, and application services. In our design, baseband processing is performed in a HW/SW co-design capable to support all the functional requirements at any air-interface and at each stage of SDR processing, from the lowest levels to higher ones. Partitioning between HW and SW follows a general cost-and-complexity versus speed trade-off. Hardware is designed in such a manner that it would support the most computation intensive task efficiently i.e. meeting the throughput and latency requirements. The hardware is also flexible enough to use the same baseband processing resources for multiple standards. The control is in software part of design, which passes the relevant paremeters to hardware for specific functionalities. The challenge in the design is to synchronize all the processings at air-interface in a efficient manner with minimum resource utilization and high accuracy.

Eurecom's research is partially supported by its industrial partners: BMW, Bouygues Telecom, Cisco Systems, France Télécom, Hitachi Europe, SFR, Sharp, ST Microelectronics, Swisscom, Thales. The research work leading to this paper has also been partially supported by the European Commission project E2R2 and by the French RNRT project IDROMEL.

II. SDR - Functional Requirements at Air-Interface

The air interfaces used by existing and future wireless communication standards include; Orthogonal frequency-division multiplexing/multiple-access (OFDM/A), Single Carrier FDMA (SC-FDMA), Wideband Code Division Multiple Access (W-CDMA), and Space-division multiple access (SDMA). The set of operations at the transmitter and receiver in the air-interface used by these schemes include Channel Estimation, Data Detection, Carrier Phase Offset (CPO) Estimation, and Synchronization. The implementations of these operations are typically tailored to the standard in question. Furthermore, these operations themselves are built up from simple macro-operations, for example Channel Estimation for OFDM systems can be implemented using FFT / IFFT, and Component-wise product of two vectors. The goal is to identify and describe the set of required operations across standards that can provide the air-interface functionality for all the standards, the said block is not necessarily efficient for all standards but is far more efficient than having different blocks for each standard supported by SDR. One such processing block performing all the air-interface operations for different standards is called Front End Processing (FEP) block in the research based SDR-Platform under progress at Institute Eurecom and ENST. The prototype being developed intends not only to fullfill current UMTS processing requirements but will also be capable of handling 3GPP Long Term Evolution (LTE) processing requirements.

The Fast Fourier Transform (FFT) has been used as building block for air-interface specific architectures both at the transmitter and receiver. Transform-based computation has traditionally been used in OFDM systems. Over the last decade, different architectures have been proposed for OFDM receivers with the FFT as the key processing block [8] [9]. For OFDM systems, FFT/IFFT block is not

only used at transmitter and receiver but can also used for Channel Estimation.
Similar to OFDM, different frequency domain computation architectures for WCDMA / HSDPA have been proposed. Frequency domain Equalization and Channel Estimation is performed in frequency domain using FFT. Iacono et al [10] [11] proposed a frequency domain block based linear equalizer for WCDMA systems, whose performance is quite similar to classical time domain equalizers. VLSI architecture for MIMO equalizer to be used for CDMA downlink using FFT was proposed by Guo et al [7]. Following the same methodology, FFT has also been utilized for MMSE turbo equalization in Global System for Mobile Communications(GSM)[6].

Though FFT makes the cardinal part of frequency domain receivers for different air-interfaces as mentioned above, there are some other blocks required to execute all the tasks. e.g. Carrier Phase Offset (CPO) estimation in OFDM system can be performed using a dot-product operation over pilot symbol position in each OFDM symbol. Dot-Product is also useful in WCDMA/Single-Carrier Systems not only for CPO but also for synchronization. Similarly, initial synchronization in 802.11x, 3G and 802.16 can be achieved using the FFT, component-wise-product, and peak detection (a max calculation over sub-bands is sufficient). Functionalities like Signal to noise ratio, automatic gain control require received signal energy, and hence an energy calculation block can be added to the functional requirements of Front End Processor (FEP). Table I lists some use cases for the macro-operations required by the different air-interfaces supported by our FEP block.

III. FEP Computational Blocks

The main functionality of FEP is discrete Direct Fourier Transform (DFT) for variable number of input samples. The processing block also offers other functionalities as conjugate, component-wise products / division, dot products, energy, max and argmax computation, rotations, rescaling, etc. Some of these functions can run in parallel, some are exclusive because they share internal resources.

A. DFT/IDFT

The Direct and Inverse Fourier Tranforms of a complex vector X of size N are defined as:

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

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

Thanks to the $\frac{1}{\sqrt{N}}$ normalization term the property $X = IFT(DFT(X))$ holds. Optionally the conjugate of the input vector may be computed before entering the DFT and the conjugate of the output of the DFT may be computed before storage in the memory. IFT is computed from $IFT(Y) = \overline{DFT(\overline{Y})}$ by using the conjugate functionality of FEP.

B. Pre-post processing

The macro-processing blocks, other than DFT/IDFT, are normally used to pre-process (or post-process) the input (output) data of the DFT/IDFT. Functional description of these processes is described below:

- Conjugate: computes the conjugate $\overline{X}$ of a vector X
- Component-wise product / division: computes the component-wise product $Z = X \odot Y$ of two vectors X and Y, this macro is also capable of division by a real vector or by a real scalar (each limited to 8-bits). The division operation is infact a multiplication by inverse of vector entities (or constant) which are stored in a look up table (LUT).
- Energy: computes the energy of a vector X:
$E(X) = \sum_{i=0}^{N-1} |X[i]|^2$
- Maximum: computes the maximum of the squares of the modules of the components of a vector, and the corresponding argmax:

$$\begin{aligned} \max(X) &= \max_{0 \le i \le N-1} (|X[i]|^2) \\ \text{argmax}(X) &: |X[\text{argmax}(X)]|^2 = \max(X) \text{ and} \\ & \forall\ 0 \le i < \text{argmax}(X); \\ & |X[i]|^2 < \max(X) \end{aligned} \quad (3)$$

- Dot product: computes the dot product of two vectors, and the square of its module:

$$\begin{aligned} D &= X.Y = \sum X_i Y_i \\ |D|^2 &= |X.Y|^2 \end{aligned} \quad (4)$$

Energy, maximum, and dot products can be computed on a whole vector or on several sub-bands of a given vector, producing as many individual results. The sub-bands are defined by their number *L*, size *M* (which is the same for all sub-bands), and starting index in the vector *I*. The different pre-post processing can be chained in several different ways. Figure 1 illustrates the different possibilities.

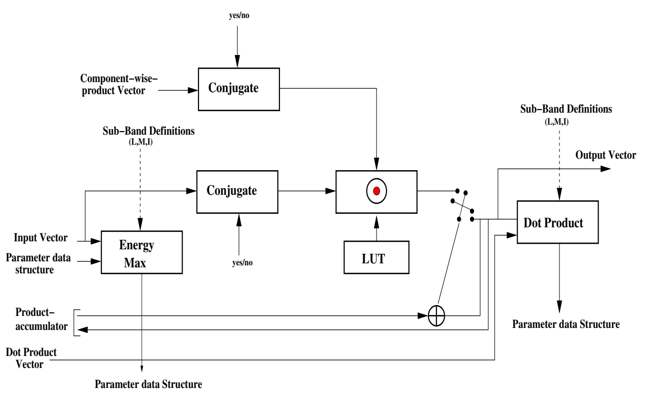

Fig. 1. Chaining of pre-post processing

IV. DSP / Hardware Selection

Once the macro-processing blocks are defined, the next step is to look for proper hardware components. A thorough analysis of computation intensive block of FEP, i.e. DFT/IDFT, was carried out to select the DSP slices for its mathematical operations. Candidate standards and ease of operations suggest that the number of input samples can be limited as powers of two between 8 and 4096. This allows to calculate DFT using simpler and efficient algorithms such as radix-4 FFT and split-radix FFT. The selection of number of bits representation of each sample is based on target technology, hardware resources, maximum achievable frequency of end product, and dynamic range of ADC converters. Therefore, all input samples to FEP block are represented in 32-bits, with real and imaginary part each of 16-bits. The data representation format chosen is Q1.15, as it matches the choice of number of bits, and also provides ease of operation in the intermediate stages of FFT calculations.

Operations	Air-Interfaces		
	OFDM/A	*SC-FDMA*	*WCDMA*
Channel Estimation	DFT + Component-wise Product	DFT + Component-wise-Product	DFT
Energy Calculations	Energy Calculations	Energy Calculations	Energy Calculations
CPO Estimation	Dot-Product over sub-bands	Dot-Product over sub-bands	Dot-Product
Data Detection	DFT + Component-wise Product	DFT + Component-wise Product	DFT + Component-wise-Product / Time-domain Dot-Product
Synchronization	Max Calculations, Filtering via DFT or Dot-Product	Filtering via DFT or Dot-Product	Dot-Product
MIMO Signal Processing	Matrix /Vector Processing (Dot-Product + Accumulation)	Matrix /Vector Processing (Dot-Product + Accumulation)	Matrix /Vector Processing (Dot-Product + Accumulation)

TABLE I
AIR-INTERFACE OPERATIONS AND RELEVANT MACRO-PROCESSING BLOCKS

To calculate N-point FFT, Radix-4 [3] and Split-Radix [2] are the most widely used and efficient schemes considering the parameters of our processing block. Radix-4 computes FFT for vector sizes that are powers of 4 in M stages (where $M = \log_4 N$), while split-radix algorithm also supports vector sizes that are powers of 2. Radix-4 is more computation efficient and is preferred over Split-Radix as long as the number of input samples (vector size) is power of 4, otherwise Split-radix is used in our design. Split-Radix performs all radix-4 stages (operations) except the first or last stage, which is a Radix-2 stage (operation). The choice of first or last stage is dependent on decimation scheme used, for Decimation in Frequency (DIF) first stage is of type Radix-2 while for Decimation in Time (DIT) last stage is of Radix-2 type.
Inputs to FFT module are input samples and twiddle factors (equi-spaced roots of unity circle, equal to number of input samples in count), and both real and imaginary parts of inputs are represented in Q1.15 data format. Radix-4 FFT algorithm implementation is based on butterfly operations [3], *N/4* butterfly operations per stage each of butterfly operation requiring Four Complex Additions and then a complex multiplication. Four additions cause an addition of 2 bits to input data resolution, so if input at any stage is represented by 'n.15' bits, the output after four additions would be '(n+2).15' bits. Then this output is multiplied with twiddle factor resulting a representation of '(n+2).30', no bit added on whole number portion of fractional number because multiplying by twiddles means rotation of input over unity circle. Least significant 15-bits can be ignored with an acceptable loss of accuracy, so a right truncation of 15 bits gives a result of '(n+2).15' bits. Thus each stage will cause an increase of two bits for each and every stage of FFT algorithm. So starting from 'n.15' bits, the output of maximum input size (4096 samples = 6 stages) will be of '(n+12).15' resolution. As the input samples are represented by '1.15', so the worst case output resolution can be '13.15' bits (28-bits). Thus the dynamic range of sample representation in DFT processing block would be between 16 and 28 bits.

The aim of the presented processing block and architecture is to come up with a research based prototyping experimental platform, and is not meant for any mass scale production; therefore for target technology selection we look FPGAs and not ASICs. Among the latest technologies available Virtex-V FPGA by Xilinx has DSP48E slices which have *25*18* bit multipliers and 48-bit adders, quite close to the requirements of FFT processing unit of FEP. The most likely input data samples to FFT operation in the FEP block are 16-QAM signal, and Zero-Mean White-Noise. Simulation analysis of these input data samples reveals that the FFT operations never exceeds the limit of 25-bits with any input samples range. This makes the Virtex-V as the obvious choice with no extra operation of saturation and/or re-scaling and also provides as good resolution of 25 bits in the intermediate stages of FFT operation. DSP48E multipliers also provide the opportunity to represent twiddle factors in Q1.17 format, thus enhancing efficiency and also reducing computation noise.

V. FEP ARCHITECTURE

Front End Processor is a part of baseband-FPGA platform which comprises of different generic processing blocks like modulator/demodulator, interleaver/de-interleaver, FEP, etc. and controlled by a central SPARC processor. These processing blocks are connected to each other and external interfaces via a common interconnect, and a generic Advanced Virtual Component Interface (AVCI) [12] compliant interface called VCIInterface in our design. The Interconnect is a crossbar and routes requests/ responses among processing blocks. VCIInterface is an AVCI wrapper for signal processing blocks and provides them a simple interface. In order to configure external devices like ADC/DAC chips, RF Rx/Tx components etc. general- purpose parallel interface modules are used. Like other processing blocks on FPGA, Front End Processor has its own Micro-controller to perform its specific tasks at its own. This not only provides efficient and effective computation but also facilitates local data transfers and low-level transactions among blocks without the intervention of central microprocessor. FEP embeds three main components as shown in figure 2 (with details of DFT macro processing block only). The host interface is responsible for the communications with the host system and for the global control and signaling; the internal memory is a local memory area, mapped on the global memory map of the system and used to store input data and output results; the FEP core is the processing unit.

Overall throughput of the most computation intensive operation i.e. DFT is set to one sample per cycle. 4096-point-DFT is the maximum value supported, with radix-4 algorithm it would take six stages while each stage would require *N/4* i.e. 1024 butterfly operations. To meet functional specification of throughput two butterfly operations per cycle need to be performed, which in turn implies that eight samples, and six twiddle factors are accessed (read from memory) in each cycle. Thanks to sophisticated memory organization and memory access algorithm, only *N/8* twiddles are stored in memory and just two twiddle factors are accessed per cycle instead of six; the details of this are out of scope of this article. To match input samples access requirements, memory banks for samples are divided into sub-memory systems to have enough memory reads / writes every cycle. Each butterfly operation requires three complex multiplications, which in turn means utilization of twelve DSP48E multipliers. Thus DFT processing block utilizes 24 DSP48E slices. Component-wise

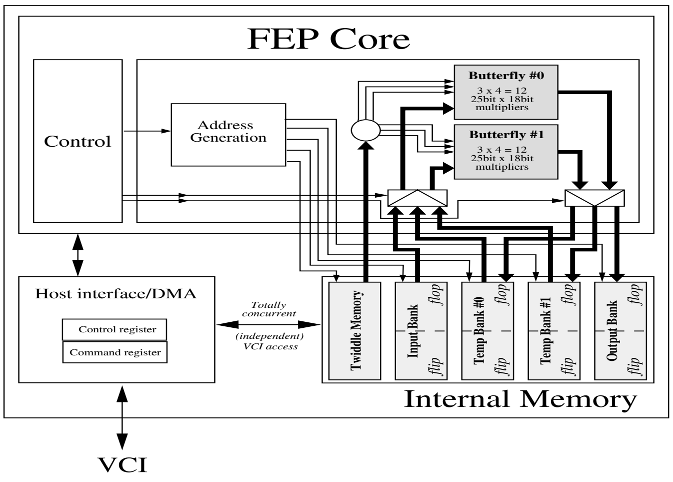

Fig. 2. FEP block diagram with description of DFT macro-processing-block

product and Dot-product processing blocks also requires multipliers, a throughput of 8 multiplications per cycle without operating these two blocks in parallel results 32 multipliers utilization. Thus in total we use 56 slices (29 %) of the total 192 DSP48E slices available in the largest virtex-V FPGA.

Initial synthesis results reveal that the critical operation in FEP, viz butterfly operation of DFT module, can achieve 230 MHz frequency (minimum). This, in turn, implies that for a 2048-point-FFT operation, a (minimum) throughput of 150 M-samples/second with the current design and can further be improved with design/code optimizations.

A. Internal Memory Organizationion

Internally memory is divided in two sub-memory systems; input memory bank, output memory bank. Furthermore Input memory consists of three input sub-memories, one each for FFT-input samples, Dot-Product input vector and Component-wise input vector. Each of these memories is of size 128 K-bits (4096 * 32 bits). Moreover, to access eight samples per cycle, input memory is divided into eight sub-banks of size 16 K-bits (512 * 32 bits) each. Twiddle-factor memory is also part of Internal Memory Subsystem, only *N/8* twiddle factors (16 K-bits = 512*32) are stored in memory and the rest of twiddles are calculated using roots-of-unity-circle property. Output memory bank stores the respective results of input banks (128*3 Kbits), and a 16-Kbits memory is used to store Energy, Max Calculation processing blocks results over sub-bands. Apart from the input and output submem-banks, to store and process intermediate stages of FFT where the data resolution is 50-bits (18*25 bits multipliers), two temp-memory banks of size 27 Kbits each is also specified.

To cater to all the standards and hence to come up with specifications, we consider 3GPP as the bench mark. The latest draft for the physical requirements for 3GPP Long Term Evolution (LTE) [4], [5] specifies generic radio frame of duration 10 ms consisting of 20 slots while two slots comprise a sub-frame. Each sub-frame consists of 14 OFDM symbols with 2048 samples each. For efficient operations, all the symbols in one subframe are loaded together in memory for DFT and other processings along with pilot symbols. Assuming 4 Receive antennas, the number of total symbols needed to be stored at the same time can safely assumed to be 64. Therefore, total memory required would be around 4 Mbits to store the input samples only. The largest Virtex-V FPGA has 10 Mbits of memory, the FEP percentage memory usage of FPGA total memory size is quite high though inside maximum limits, and if other processing blocks of baseband computation need more memory then an external double data rate dynamic random access memory (DDR DRAM) may be added to cater these requirements.

VI. Conclusions

In this article, we have proposed a global air-interface hardware processing block that can be configured to support almost all the existing and in-progress wireless communication standards. The design is based on identifying small macro-processing blocks which can be re-used for multiple standards, thus providing hardware flexibility and high efficiency. Using Virtex-V of Xilinx, the resource utilization is quite reasonable. Further the control of the block is simple, and all the parameters for specific processing of supported functionality using single command word. Future work includes, more rigorous hardware analysis to examine parmeters such as maximum achievable frequency, area, resource utilization etc, and also benchmarking Front-End-Processor and its macro-processing blocks with the existing dedicated blocks for different standards.

References

[1] J. Mitola, "The Software Radio Architecture", IEEE Commun. Mag., vol. 33, no. 5, May 1995, pp. 26-38.
[2] R. Yavne, "An economical method for calculating the discrete Fourier transform," in Proc. AFIPS Fall Joint Computer Conf. 33, 1968, pp. 115-125.
[3] Cooley, J., and Tukey, J., "An algorithm for the machine computation of the complex Fourier series", Math. Computat., 1965, 19, pp. 297-301.
[4] 3GPP LTE Activities available at http://www.3gpp.org/tb/home.htm
[5] 3GPP Release 8 Spec available at http://www.3gpp.org/ftp/Specs/html-info/36-series.htm
[6] Christophe Laot, Raphal Le Bidan, "Low-Complexity MMSE Turbo Equalization: A Possible Solution for EDGE", IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, VOL. 4, NO. 3, MAY 2005.
[7] Yuanbin Guo, Jianzhong(Charlie) Zhang, Dennis McCain, and Joseph R. Cavallaro, "An Efficient CirculantMIMO Equalizer for CDMA Downlink: Algorithm and VLSI Architecture", EURASIP Journal on Applied Signal Processing Volume 2006, Article ID 57134.
[8] 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, APRIL 2007.
[9] 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.
[10] Daniele Lo Iacono, Julien Zory, Ettore Messina, and Nicolo' Piazzese, "Block processing engine for high-throughput wireless communications", 2nd International Symposium on Wireless Communication Systems, 2005.
[11] Daniele Lo Iacono, Ettore Messina, Costantino Volpe, and Arnaldo Spalvieri, "Serial Block Processing for Multi-Code WCDMA Frequency Domain Equalization", IEEE Wireless Communications and Networking Conference(WCNC), 2005.
[12] Consortium VSIA http://www.vsia.org/
[13] Virtex-5 User Guide available at http://www.xilinx.com