Evaluation of a Heterogeneous Multicore Architecture by Design and Test of an OFDM Receiver

Sajjad Nouri, Waqar Hussain, and Jari Nurmi

Abstract—This paper presents an evaluation of a Heterogeneous Multicore Architecture (HMA) by implementing Orthogonal Frequency-Division Multiplexing (OFDM) receiver blocks as designs for the test of functionality. OFDM receiver consists of computationally intensive and general-purpose processing tasks that can provide maximum coverage to test and evaluate a massively-parallel as well as a general-purpose platform like the HMA. The blocks of the receiver are primarily designed by crafting template-based Coarse-Grained Reconfigurable Array (CGRA) devices and then arranging them in a sequence over a Network-on-Chip (NoC) structure along with a few RISC cores for complete OFDM processing. The OFDM blocks such as Fast Fourier Transform (FFT) and Time Synchronization are computationally intensive and require parallel processing. The OFDM receiver also contains tasks such as frequency offset estimation which require the processing of Taylor series and CORDIC algorithms that are serial in nature. Such a combination of serial and parallel algorithms can perform a thorough exploration and evaluation of almost all the design features of an HMA. The OFDM implementation has led to scale CGRAs to different dimensions, instantiate Processing Elements (PEs) as multiple arithmetic resources and to establish almost all possible ways of PE interconnections. It further explores time-multiplexed patterns for data placement in the CGRA memories. Nevertheless, the data can also be exchanged among different nodes over NoC structure simultaneously and independently by using direct memory access devices. In this experimental work, the performance of each CGRA, the collective performance of the whole platform and the NoC traffic are recorded in terms of the number of clock cycles and several high-level performance metrics. Today's HMAs are generally over or under resourced for the applications that they are designed for and thus not an optimal choice for the end user. Apart from the interesting comparisons to the other state-of-the-art, our experimental setup has provided important insight and guidelines that the designers can use to implement near-optimal solutions for their target applications.

Index Terms—Reconfigurable, CGRA, Network-on-Chip, Heterogeneous, Accelerator, Multicore, FFT, Time Synchronization, Channel Estimation, Frequency Offset Estimation, Receiver, OFDM.

1 Introduction

As Moore predicted that the number of transistors on a chip will double every two years, Dennard showed that the power dissipation density can be kept constant by scaling the voltage and the dimensions of the transistor ([1], [2]). It therefore allowed packing more transistors in the same chip area with ever increasing toggle rates. This trend did not last longer due to increasing leakage currents caused by near-threshold operations and thus, the enormous heat dissipation put an end to Dennardian scaling and therefore an end to Moore's law. A solution was found in multicore scaling as when a team at Massachusetts Institute of Technology presented RAW microprocessor architecture - one of the first multicores [3]. However, the recent findings from an experiment show an end also to multicore scaling. The work concludes that only 7% of a 300mm^2 chip can be operated at full frequency under a power budget of 80W [1]. It further explains that on a single chip, all cores cannot be clocked at their maximum operating frequency as the instantaneous power dissipation can be potentially very high to cause a thermal melt-down. As a preventive measure, a large fraction of the chip is switched-off (dark) or operated at a very low frequency (dim). Michael B. Taylor has indicated *Four Horsemen* as top contenders to deal with this issue, i.e., Shrink, Dim, Specialized and Deus Ex Machina [4]. Among them, the *Specialized Horseman* suggested the use of application-specific accelerators to combat the problem of Dark-Silicon. In this case, Coarse-Grained Reconfigurable Arrays (CGRAs) are advocated since they are operated at a very low frequency and can yield tremendous performance improvements. They are also reconfigurable and programmable with a higher level language. The main idea of this paper is motivated by the *Specialized Horseman* since we have used programmable CGRAs in a Heterogeneous Multicore Architecture (HMA) and special-purpose accelerators are designed for performing massively-parallel workloads of critical-priority applications.

There are a number of heterogeneous platforms that already exist and have nearly similar design philosophies. These platforms are generally structured over a Network-on-Chip architecture while many general-purpose and application-specific cores are integrated over it. A few examples are NineSilica [5], Platform 2012 [6] and MORPHEUS [8], [9]. However, the demand and efficient utility of on-chip resources has to be justified. It is observed that most of the on-chip resources are not utilized over the entire frame of execution-time [1]. Based on the recommendations published in this famous article on Dark Silicon [4], we have crafted a heterogeneous platform and subjected it to a stringent test for identifying potential architectural fallacies and pitfalls. The designs for test are

• *S. Nouri, W. Hussain, and J. Nurmi are with the Laboratory of Electronics and Communications Engineering, Tampere University of Technology, Tampere, Finland. W. Hussain is also with the Department of Computer Science, Norwegian University of Science and Technology, Norway. E-mail: (sajjad.nouri), (waqar.hussain), (jari.nurmi)@tut.fi and waqar.hussain@idi.ntnu.no*

Manuscript received XXX XX, XXXX; revised XXX XX, XXXX.

Orthogonal Frequency Division Multiplexing (OFDM) receiver blocks which contain some of the most computationally-intensive tasks, i.e., Fast Fourier Transform (FFT), Correlation, Convolution and Complex Matrix-Vector Multiplication (MVM). Selecting OFDM application as a test also provided cases for checking the competence of the platform's general-purpose processing capability for serial algorithms such as Taylor Series and CORDIC algorithms. Although a single case study would not be enough to empirically assess an architecture, we try to tackle the assessment difficulty so that our test application consists of a fine mixture of parallel and serial tasks in order to provide sufficient switching activity on almost all the architectural features that can lead to some meaningful and tangible conclusions. The computationally intensive tasks are implemented by crafting CGRAs to algorithmic constraints and the general-purpose tasks are delegated to Reduced Instruction-Set Computing (RISC) processors. The CGRAs and the programmable RISC processors cooperate over the NoC for complete OFDM implementation and testability.

The HMA platform used in this experimental work is designed for maximizing the number of computational resources to accelerate many specific algorithms of different nature and is called Heterogeneous Accelerator-Rich Platform (HARP) [10], [11]. The architecture of HARP consists of nine nodes arranged in a topology of three rows and three columns. As part of this design and test regime, our particular HARP instance contains a few RISC cores which not only act as the controllers of the system but also distribute the configuration streams and the data to be processed to all the other cores integrated to the NoC platform. After configuration and data distribution tasks, the RISC cores are available for general-purpose processing and they establish synchronization among all the cores as well as monitor their performance. All accelerators related to the OFDM receiver blocks are designed by tailoring the template-based CGRAs efficiently to the computational requirements of the algorithms as well as customizing their interconnection. The platform is evaluated as an OFDM receiver that is primarily a proof-of-concept test so a wide selection of diverse applications are available to simulate almost all the components of the platform. The simulation of the test case shows, that the platform has the architectural characteristic parameter of almost 12 Mega Operations Per Second (MOPS)/mW for a 28 nm Altera Stratix-V FPGA device.

This paper is organized as follows. Section 2 presents some of the existing state-of-the-art platforms related to HARP. Section 3 explains the architecture of HARP and the template-based CGRAs used for integration, i.e., CREMA [12] and AVATAR [13]. Moreover, the overall functionality of HARP as well as the internal structure of the NoC nodes are described in detail. In Section 4, the design and implementation of OFDM receiver blocks by using template-based CGRAs are presented. Then Section 5 explains the instantiation and implementation of OFDM receiver baseband processing on HARP as well as data distribution among different nodes. Section 6 presents measurement and estimation of several basic and advanced level performance metrics related to HARP when prototyped on a FPGA device. In section 7, the OFDM processing instance of HARP is compared with the other state-of-the-art platforms performance-wise and from an architectural point of view. Finally, the last section presents the conclusions.

2 Related Work

The main motivation behind designing HARP is to provide a platform to accelerate many specific algorithms finely tailored in hardware while the issues related to Dark Silicon can be investigated. In addition to HARP, many other state-of-the-art platforms have been presented with almost similar features which are explained in the following.

NineSilica [5] is a general-purpose homogeneous Multi-Processor System-on-Chip (MPSoC) platform composed of a 3×3 mesh of homogeneous cores connected over a NoC. A RISC processor was chosen as a computational engine in the integration with the nodes. Many Software-Defined Radio (SDR) applications have been designed and implemented on the NineSilica platform such as FFT and Correlations. Based on the performed experimental work, NineSilica requires 10.3 μs for executing 64-point FFT on a 40 nm FPGA device.

In [14], an architectured FPGA approach is presented as an integration of various general-purpose microprocessors (μP). The main idea was designing a programmable parallel processing platform on a FPGA. Some additional accelerators called Processing Units (PU) can also be linked with the microprocessors according to the user specifications while all the PUs can be either the same or different Single Instruction Multiple Data (SIMD) cores. Two separate networks (low and high latency) are provided for exchanging data and messages among the μPs and PUs. The overall system is synthesized over a 90 nm FPGA delivering a performance of 19.2 Giga Operations Per Second (GOPS).

ADRES (Architecture for Dynamically Reconfigurable Embedded System) is a reconfigurable array of 8×8 elements as a CGRA tightly integrated with a Very Long Instruction Word (VLIW) processor [15]. Each of the processing elements employed in ADRES contains Functional Units (FUs) and Register Files (RFs) connected in a mesh topology. There are also routing resources composed of wires, buses and networks. The connection among FUs is made by a multi-port global Data Register File (DRF) with the data bus width of 32 bits. High data level parallelism is provided by using the ability of FUs to support SIMD processing. The RFs are used for storing intermediate data where the number of words is equal to 16 and 64 in local an global RF, respectively. The ADRES specific instances can be generated using an XML-based architecture specification language. ADRES is implemented using 90 nm CMOS technology and shows a performance of 4 MOPS/mW [16].

Platform 2012 is an area- and power-efficient multi-core computing accelerator which is defined as locally synchronous and globally asynchronous processor clusters [6]. Each of four clusters is comprised of 16 general-purpose RISC processors with independent instruction streams. Clusters are communicating with each other using a high-performance fully-asynchronous NoC. For synchronization and power management, a specialized hardware as a multi-channel DMA engine is employed in P2012. There is also a heterogeneous extended version of P2012 which is presented in [7] and is called He-P2012. The platform delivers 40 MOPS/mW performance using a 28 nm CMOS technology.

MORPHEUS ([8], [9]) is a complex and dynamically reconfigurable SoC which is the most recently published heterogeneous multiple accelerator platform. It is composed of three different types of reconfigurable devices classified according to the level of granularity: fine-grained, middle-grained and coarse-grained. FlexEOS as an embedded FPGA is well suited for fine-grained algorithms. FlexEOS is composed of 4K Multi-Function logic Cells (MFC) based on SRAM 1-bit Lookup-Tables (LUT) which is a re-programmable FPGA fabric and can be programmed by using standard description languages such as VHDL and Verilog. DREAM [17] is a middle-grain reconfigurable Digital Signal Processor (DSP) core composed of a 32-bit RISC core processor and PiCoGA-III reconfigurable datapath (matrix of reconfigurable logic cells). It promises to deliver a 0.2 GOPS/mW performance using a 90nm CMOS technology. The coarse-grained one is XPP-III which is integrated with the datapath of a VLIW processor. Its main purpose is providing highly parallel processing performance for streaming applications. All the aforementioned devices along with the system modules (Heterogeneous Reconfigurable Engines (HREs), memory units and I/O peripherals) communicate with each other over a NoC. The overall MORPHEUS chip is synthesized on a 90 nm CMOS technology providing 0.02 GOPS/mW performance with an average dynamic power of 700 mW.

Compared to all above mentioned state-of-the-art platforms, the design of HARP presents more processing resources and integrated accelerators which results to high computational power.

3 Platform Architecture

The experimental platform is HARP template that generally allows to instantiate nine NoC nodes at maximum. The central node is usually integrated with a RISC core called COFFEE [19]. It is essential to allow programmability and constant supervision of the platform while it can also be used for general-purpose processing. CREMA- and AVATAR-generated accelerators can be integrated with other nodes in order to accelerate computationally intensive tasks. In the following sections, the architecture of template-based CGRAs used to build HARP as well as the general program flow are explained in detail.

3.1 Coarse-Grained Reconfigurable Arrays

CREMA and AVATAR have almost the same architectural features, shown in Fig. 1. The only difference is related to their sizes which are adjusted based on the proposed applications. CREMA is equipped with R rows × 8 columns of PEs, where R is application dependent, and two 32-bit local memories each of size 16 columns × 256 rows. AVATAR is a scaled-up version of CREMA which is composed of R rows × 16 columns of PEs as well as 32 columns × 512 rows for each local memory. The data can be interleaved between local memories and the PEs by using the I/O buffers which are made of sixteen 16 × 1 multiplexers and sixteen 32-bit registers for CREMA and twice the size for AVATAR, accordingly. During an operation, the data to be processed over the PE array is loaded and stored sequentially into the local memories by utilizing the Direct Memory Access (DMA) device [20].

Each PE can receive two operands at its inputs and contains a 32-bit Arithmetic and Logic Unit (ALU) in order to perform both integer and floating-point operations in IEEE-754 format. The components of PE core can be separated into two main parts: Functional Units (FU) and configuration control blocks. A PE is composed of a LUT, adder, multiplier, shifter, immediate register and floating-point logic which will be selectivity instantiated at design-time according to the processing requirements of an application. As it can be seen from Fig. 1, each PE in a CGRA has interconnections with neighboring PEs in a point-to-point fashion with multiple routing possibilities, i.e., local, interleaved and global. Local interconnections are only with the nearest neighboring PEs while the global interconnections are used among the PEs that are relatively farther from each other.

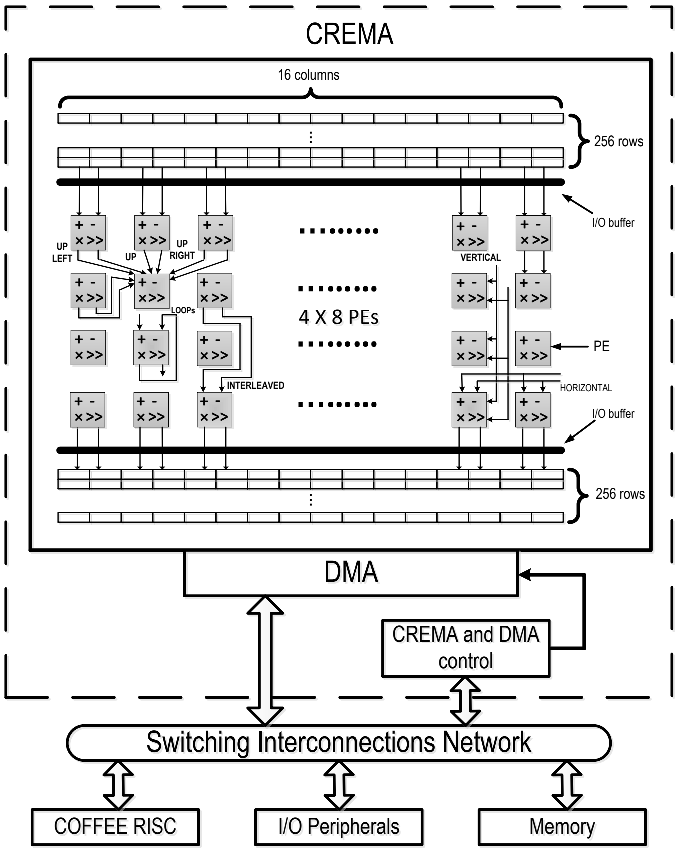


Fig. 1. The architecture of scalable template-based CGRA used for integration over NoC.

For each particular application, the suitable and most efficient placement and routing have to be performed based on its algebraic expressions. At any clock cycle, the pattern of interconnections among all PEs and the set of operations to be performed by each PE in the CGRA is called *context*. The contexts are designed by using custom graphical user-interface tool at the system design-time and can be enabled at run-time for an execution step. The contexts can be switched at different instances of execution time based on the application scenario. Each PE has its own configuration memory where the configuration words are stored during the system startup time. The configuration words are sequentially injected into the PE arrays by the DMA device using a pipelined infrastructure [21]. Each configuration word is composed of an address and operation field which are used to determine the task of each PE and its destination address, respectively. The control flow of CREMA- and AVATAR-generated accelerators is generally programmable in C language while COFFEE RISC core performs the control operation by writing control words to the control registers of the CGRA accelerators. The contexts can be changed based on the configuration data in the CGRA. Thus, reconfiguration is performed when there is a need to replace an already existing configuration stream. The execution flow can be listed as follows:

1) The configuration data is loaded in the CGRA with the help of a DMA device at the system start-up time.
2) The data to be processed is loaded into the local memories

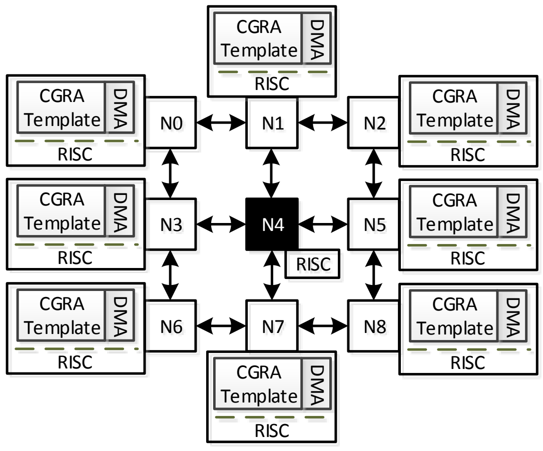


Fig. 2. Heterogeneous Accelerator-Rich Platform (HARP) Template.

of the CGRA.

3) Enabling a context to configure the functionality of the PEs as well as routing between them.
4) The data is processed over the PE array.
5) The CGRA is reconfigured by changing the context as required.
6) A new set of data can be loaded for processing and the processing continued from Step 3.

These phases can be iterated until the algorithm completes its execution and then the results will be transferred from the local memory of the CGRA to the RISC processor or the local memory of another CGRA for further processing. It has to be noted that steps 4 and 5 can not be interleaved and should be performed in order.

3.2 Heterogeneous Accelerator-Rich Platform

The HARP platform is written in parametrized VHDL. It is constructed over a NoC of nine nodes arranged in a 3×3 mesh topology. As it is shown in Fig. 2, the central node is integrated with COFFEE RISC core while the rest of the nodes contain a template-base CGRA, a data memory and a DMA device with supervisor and slave interface. Furthermore, some other nodes can also be integrated with COFFEE and act as supervisor nodes based on the application requirements. Since HARP is a template-based architecture, the user can integrate a CGRA of a specific dimension to any of the existing nodes or even leave the node as a data routing resource.

A detailed view of master and slave nodes of HARP is shown in Fig. 3. As it is depicted, each node of NoC has one master and two slave interfaces. The master interface is responsible for writing to the network, performing the data transfer within the node and is therefore integrated with the RISC core. However, the master interfaces of other nodes are connected to the master side of the DMA device while their slave interfaces are integrated to the local memory and slave part of the DMA device. The supervisor node containing a RISC processor is responsible for transferring data among its own data memory and the other slave nodes data memory. The RISC core(s) establish synchronization for data transfer between two different nodes. For this purpose, a shared space is allocated in the data memory of RISC core(s) for setting and resetting ’read’ and ’write’ flags. The shared space can be written by the other nodes as well.

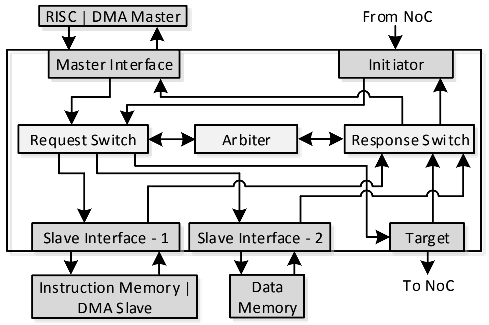


Fig. 3. An overview of the Supervisor and Slave nodes of HARP.

During the system start-up time, the supervisor node of the NoC (N4) starts the data transfer by sending the configuration stream and the data to be processed from its data memory to the data memories of the slave nodes. Both data and configuration stream are transferred over the NoC in the form of packet transmission. The packet is composed of two parts: the routing information in the header and the data and configuration words in the rest of the packet. The packet is received first at the initiator and then forwarded to the request switch of the destination node. A node can be contacted by using the initiator module while the node’s arbiter resides between the request and response switch in order to establish connections among different modules. The targeted slave device then gets selected by the request switch based on the address field of the routed packet, which can point to the data memory of the respective node or the DMA slave. The data can also be written to the NoC through the target module. In the case of RISC core which requires to read/write data from/to the instruction or data memory, a node’s master should contact the request switch in order to get connected to one of its local slave devices. Once the transport route is specified, the configuration stream as well as data to be processed can be loaded into the local memory banks of the template-based CGRA by using the DMA devices in the slave nodes. It has to be considered that the next control word might be sent to the DMA device once the previous data transfer task has been completed after several cycles. Within this period, the RISC core may perform the same operations for the other nodes as well. As it is mentioned above, synchronization has to be established between supervisor and slave nodes by using an allocated shared memory space. The size of the shared memory space depends on the number of slave nodes. The synchronization is established in such a way that the RISC core writes ’1’ in the shared memory location corresponding to the destination node. The packet is then sent to the destination node by the RISC core that also requests the DMA slave to start the data transfer. As the next step, the DMA master starts fetching the stream from the node’s data memory and distributes it to the configuration memories or one of the local data memories of CGRA. As soon as the DMA completes the data transfer, an acknowledgment will be sent by DMA’s master and RISC core writes ’0’ over the NoC which results to reset the shared memory location corresponding to the data memory of the supervisor node. During this process, the RISC core should not send any new packet and activation control word to the same node’s DMA. Thus, the sequence of data transfers can be retained.

Once the configuration stream and the data to be processed are loaded into the local memories of slave nodes and also when the internal data transfer of the template-based CGRAs is performed

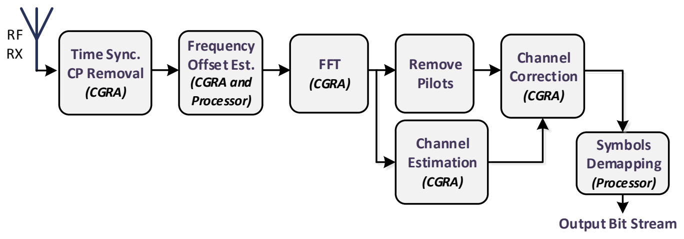

RF
RX
Time Sync. CP Removal (CGRA)
Frequency Offset Est. (CGRA and Processor)
FFT (CGRA)
Remove Pilots
Channel Correction (CGRA)
Channel Estimation (CGRA)
Symbols Demapping (Processor)
Output Bit Stream

Fig. 4. A simplified block diagram of the IEEE 802.11a/g receiver.

completely by the DMA device, the control words can be sent to the CGRAs by the RISC core for cycle-accurate processing. The slave nodes containing a CGRA can work in parallel and independent of each other. However, some data dependencies in the program flow might happen which requires exchanging data between CGRA nodes. In other words, the stored results in one of the local memories of CGRA can be transferred to the other slave nodes for further processing. Furthermore, the same process of synchronization has to be performed by the RISC core as it allocates the location of shared memory corresponding to the receiver node and then targets the DMA device of the transmitter node. In the next section, the whole mechanism of HARP as well as sharing computational resources of the CGRA are explained in detail. It can be seen that various execution and data transfer programming schemes can be applied to the platform by the users according to the requirements of the specific application.

4 Design of IEEE 802.11a/g Receiver Blocks

In this experimental work, OFDM receiver baseband processing is designed onto the HARP platform, based on IEEE 802.11a/g specifications. OFDM is a promising technique for achieving higher data rates while it includes some advantages such as resilience to Inter-Symbol Interference (ISI) and to frequency selective fading caused by the multipath propagation [22]. The simplified physical layer of the IEEE 802.11a/g receiver and the components implemented by CGRA or processor are depicted in Fig. 4. Once the entire OFDM symbol is transmitted across the channel and Analog to Digital Conversion (ADC) is implemented, the receiver is responsible for performing Time Synchronization, Frequency Offset Estimation, FFT, Channel Estimation and Symbols Demapping. The received packet is composed of short and long training symbols which are called preamble, signal field which contains one OFDM symbol and data field with variable number of OFDM symbols. In general, the preamble is composed of predefined samples which are known to the receiver and used for synchronization purposes. According to the IEEE 802.11a/g standard specifications, the first seven segments of short training symbols can be utilized for packet detection while the last three parts are used for coarse frequency offset estimation. Each segment contains 16 subcarriers based on specific repetition. The long training symbols contain 64 identical samples of long OFDM symbols which are also used for channel estimation and fine frequency offset estimation. In the following subsections, the design and implementation of application-specific accelerators for the aforementioned OFDM receiver blocks using template-based CGRA is presented in detail. In this paper, all designed CGRAs are scaled up or down in order to have near optimal implementation (employing most of the PEs) of application mapping in terms of performance, resource utilization, power dissipation and execution time.

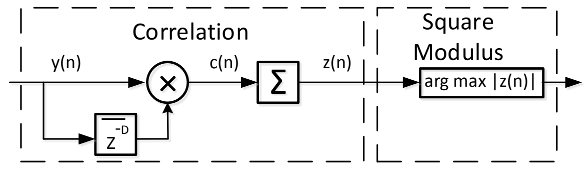


Fig. 5. Signal flow structure of the delay and correlation algorithm for Time Synchronization.

4.1 Time Synchronization

Time Synchronization is the first block of OFDM receiver after analog to digital conversion. This task is required for detecting the precise moment of arrival of received OFDM symbols as well as finding the right position of FFT window. One of the methods for performing Time Synchronization is Cyclic Prefix (CP) correlation based method [23]. The CP refers to the copy of the end part of OFDM symbol as a prefixing of a symbol in order to cope with ISI and enable the OFDM signal to operate reliably. As it is shown in Fig. 5 based on CP correlation method, the received signal y_n is correlated with its delayed version. In general, correlation is required in different applications to determine the similarity between two signals. The length of delay z^{-D} is equal to 16 which is the length of CP based on IEEE 802.11a/g standard specifications. The correlation algorithm will produce outputs c_n and z_n expressed in Eq. 1 and Eq. 2, respectively. Here the $*$ stands for the complex conjugate.

$$c_n = y_n y^*_{n-D} \tag{1}$$

$$z_n = \sum_{i=0}^{L-1} c_{i+n} \tag{2}$$

At this step, the above equations must be mapped over the PE array of AVATAR. The reason behind choosing AVATAR is to achieve higher processing speed and reduce the execution time. AVATAR is also further scaled up to 5×16 PE array for this specific design. In order to perform placement and routing in more efficient way, Eq. 1 can be simplified as Eq. 3 where R and I stand for Real and Imaginary parts of the received signal, respectively.

$$\begin{aligned} c_n = &\underbrace{((y_{n(R)} \times y_{n-D(R)}) + (y_{n(I)} \times y_{n-D(I)}))}_{Real} \\ &+ \underbrace{((y_{n(I)} \times y_{n-D(R)}) - (y_{n(R)} \times y_{n-D(I)}))}_{Imaginary} \end{aligned} \tag{3}$$

The mapping of an 80-point correlation algorithm on AVATAR based on Eq. 3 is performed in two consecutive contexts depicted in Fig. 6. The first context which is not shown here is loading immediate values to the PEs for shift operation after each multiplication. It is required in order to prevent any overflows caused by multiplication. It has to be mentioned that some PEs can be switched off by disabling the specific parts of I/O buffers. As the first step, the received OFDM data symbols are loaded into the first local memory of AVATAR. Two different tasks are performed by the second context. The first one is related to the multiplication among the received data symbols and the complex conjugation of its delayed version (the first row of PEs) which is mapped over

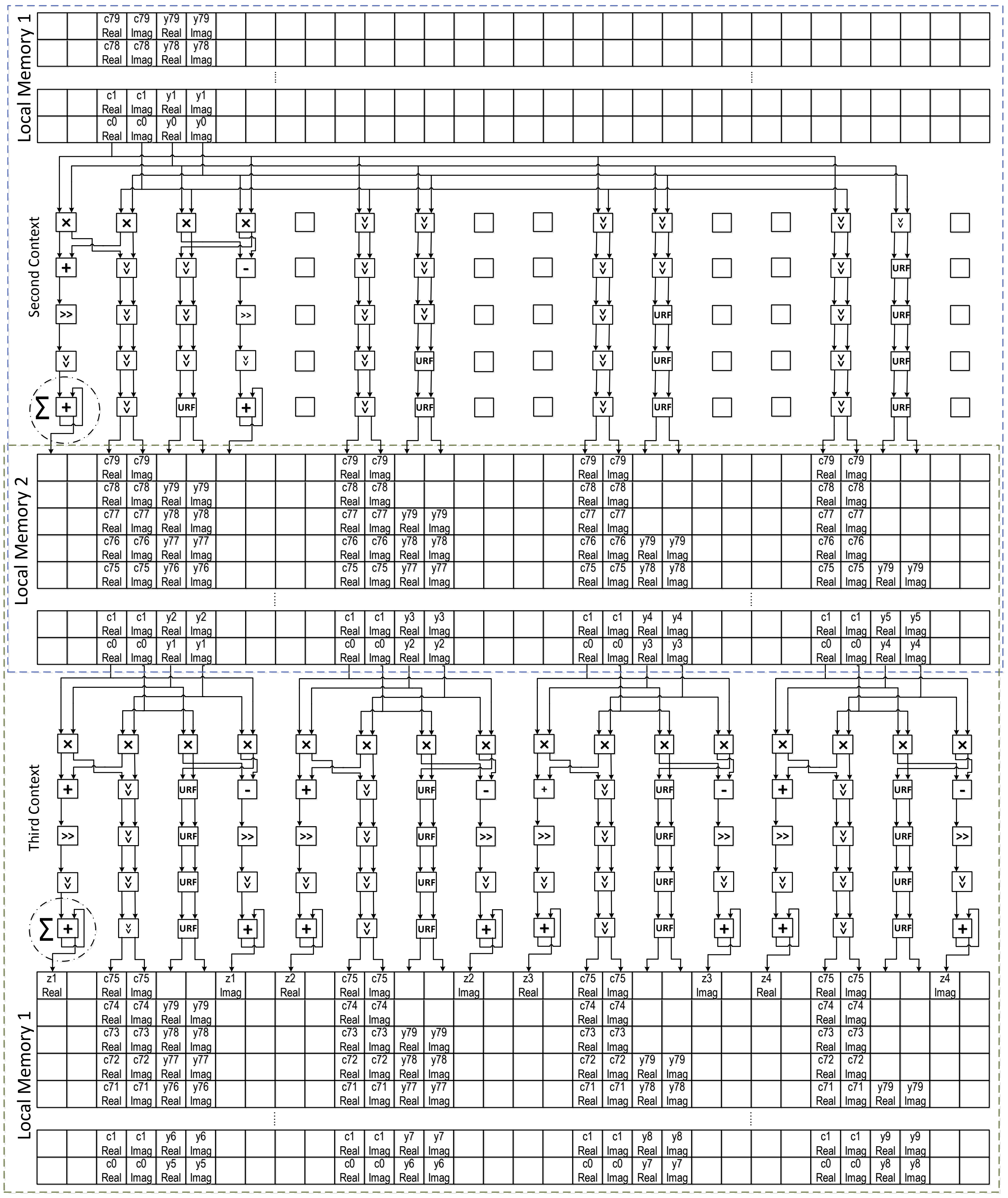


Fig. 6. Second & Third Contexts for the Calculation of the Correlations ©IEEE 2015 [24]. c_n and y_n stand for the original received signal and the complex conjugate of the delayed version of the signal, respectively.

the PE array based on Eq. 3 (addition and subtraction can also be observed from the second row of PEs based on the mentioned equation). The data indicated by the indexes from 0 to 79 belong to 80-point correlation. 80 correlations are required in order to implement time synchronization for 80 data symbols. The second task is distributing the data to the other columns of local memory in order to maximize the parallel usage of resources (columns 6, 10 & 14 of the second context). The second context can only be used for performing the first correlation as well as data distribution. Then the stored results in the second local memory will be used by changing the direction of data flow to the first local memory for executing four correlations in parallel. As it can be observed from the last row of PEs (third context) in Fig. 6, a sum-of-products of the results of complex multiplication is performed for calculating the final result of each correlation based on Eq. 2. The summation operation is performed by using feedback operation while the number of iterations is equal to the number of existing data symbols. Furthermore, the data symbols have to be delayed after each correlation by one cycle which can be performed by using Unregistered-Feed Through (URF) feature of the PE. In the third context, four URFs are used for shifting the delayed version of data symbols by four positions during each iteration. It allows accomplishment of the next step by transmitting the shifted version of data to the second local memory. Thus each four correlations and data shifting are implemented in parallel using the third context until all 80 correlations complete their execution.

The next step after completion of all the 80 correlations is seeking the maximum value which can be performed by the RISC processor (N3). This largest value is corresponding to the index of the time offset specifying the edge of the first FFT window and can be calculated by using Eq. 4. It should be mentioned that since the results of correlations are complex, the Square Modulus (SM) is required to calculate the magnitude of complex numbers.

$$\begin{aligned}\hat{\tau}_s &= \underset{n}{\operatorname{argmax}} \mid z_n \mid \\ &= \underset{n}{\operatorname{argmax}} \mid z_{n(R)} \times z_{n(R)} + z_{n(I)} \times z_{n(I)} \mid\end{aligned} \tag{4}$$

Here $\hat{\tau}_s$ stands for the largest value among the 80 results of the performed correlations (z_n) where R and I represent the Real and Imaginary parts. Once the computation of square modulus is completed, the largest value can be found by performing search algorithms in the RISC core. At the end, the index of time offset should be transmitted to the Master node (N4) for further processing.

4.2 Frequency Offset Estimation

Despite all the advantages of OFDM, it suffers from sensitivity to Carrier Frequency Offset (CFO) because of the device impairments [22]. Therefore, the received baseband signal will be centered at f_Δ (stands for the frequency offset) after down-conversion at the receiver side instead of zero. CFO may cause ISI and also rotation of demodulated symbols in the constellation. One of the methods for estimating the CFO is using special training symbols which are added in the transmitter [22]. As it is mentioned above, the last three segments of the short training symbols can be used for this purpose. Let us assume that x_n and r_n are transmitted and received signal, respectively. Then, the down-conversion of the received signal r_n can be expressed in Eq. 5.

$$r_n = x_n e^{j2\pi f_\Delta n T_s} \tag{5}$$

CFO can be estimated by using delay and correlation method applied on the received data symbols and its delayed version which

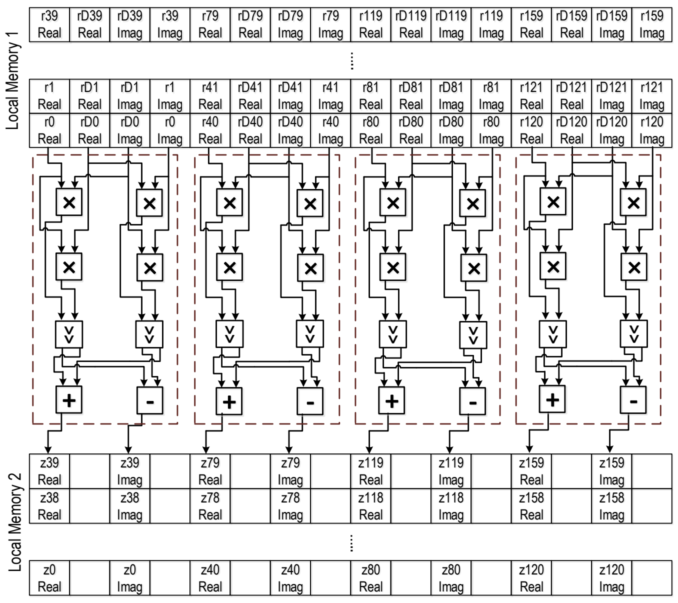


Fig. 7. Context for the multiplication between a signal and its complex conjugation ©IEEE 2015 [24].

can be expressed as

$$\begin{aligned} z &= \sum_{n=0}^{L-1} r_n r_{n+D}^* \\ &= \sum_{n=0}^{L-1} \big(\underbrace{((r_{n(R)} \times r_{n+D(R)}) + (r_{n(I)} \times r_{n+D(I)}))}_{Real} \\ &\quad + \underbrace{((r_{n(I)} \times r_{n+D(R)}) - (r_{n(R)} \times r_{n+D(I)}))}_{Imaginary} \big) \end{aligned} \tag{6}$$

where the value of delay D can be calculated by multiplying the period of short training symbols and frequency space (0.8 μs $\times$ 20.0 MHz). R and I also stand for the real and imaginary part of the signal, respectively. The short training symbols utilized for frequency offset estimation are composed of 48 predefined data symbols added prior to transmitting the signal through the channel. Accordingly, 48 complex multiplications are required which are designed as a context shown in Fig. 7. Here r and rD stand for the last 48 received short training symbols and its delayed version, respectively. Once the complex multiplication is performed, the frequency offset can be achieved using the following equation

$$\hat{f}_\Delta = -\frac{1}{2\pi D T_s} \angle z, \tag{7}$$

where T_s is the sampling period and $\angle$ takes the angle of z. In order to find the phase angle of a complex number, the following equation can be expressed.

$$\hat{\theta} = atan(\frac{y}{x}) \tag{8}$$

The required division between the imaginary part y and the real part x can be performed in processor software by using CORDIC algorithm [25] which is one the most popular algorithms due to the simplicity of its hardware implementation. It has to be mentioned that the algorithms with complex algebraic equations and high number of iterations such as CORDIC are not efficient to be mapped on CGRA in terms of execution time. Moreover, the designed CGRA is simple and not capable to do complex algebraic and trigonometry related operations. Therefore, they can be performed by means of processor software with shorter execution time at the cost of more energy and power. Once the result of division is calculated for all 48 complex numbers, the phase angle can be computed in processor software by expanding Taylor series for the arctangent ratio based on Eq. 9.

$$arctan\ x = \sum_{n=0}^{N} \frac{-1^n}{2n+1} x^{2n+1} \tag{9}$$

The value of N depends on the required accuracy, it is equal to 4 in this specific case. The last step of this block is frequency offset correction which can be performed by multiplying the estimated frequency offset and the received signal based on Eq. 10.

$$r_n{}' = r_n \times e^{-j2\pi f_\Delta \frac{n}{N}} \tag{10}$$

Here, $r_n{}'$ is the corrected signal, n is the sample index and N is the number of samples in a symbol. In order to calculate the exponent of $-j2\pi f_\Delta \frac{n}{N}$ function required for frequency offset correction, the algorithm of Taylor series should be applied according to the following equation.

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} \tag{11}$$

Considering the complex numbers for the received data symbols, the above equation should be modified as

$$e^z = e^x(cos(y) + isin(y)) \tag{12}$$

where z is composed of the real x and imaginary y parts. Thus, *cos* and *sin* functions can also be expanded by using Taylor series which are depicted in Eq. 13.

$$cos\ y = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} y^{2n} \quad , \quad sin\ y = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} y^{2n+1} \tag{13}$$

Once all of the aforementioned steps have been performed in processor software, the data can be transferred again to the local memory of CGRA in order to perform the complex multiplication related to Eq. 10. It can be implemented by using almost the same context as shown in Fig. 7.

4.3 Fast Fourier Transform

Subsequent to the correction of the received signal in terms of the frequency offset, the data symbols have to be converted from time domain to frequency domain which is called demodulation. It can be performed by using FFT as a special case of Discrete Fourier Transform (DFT). Compared to the other blocks in the receiver side, FFT is among the most time consuming and computationally intensive ones. Mathematically, the DFT of a finite length sequence *x(n)* can be defined as

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \tag{14}$$

where the finite sequence of N complex numbers is converted into an equivalent-length N-periodic sequence of complex numbers and $W_N^{nk} = exp(-j2\pi \frac{nk}{N})$ is a twiddle factor. DFT can be efficiently processed using FFT of radix-2^m structures [26], where $m \in Z^+$ and its structural unit is called a butterfly. Based on the IEEE 802.11a/g specifications, demodulation is performed by a 64-point FFT within 3.2 μs. In this case study, the CGRA accelerator for processing 64-point FFT has been implemented in radix-4 scheme using AVATAR-template and is explained in detail in [13].

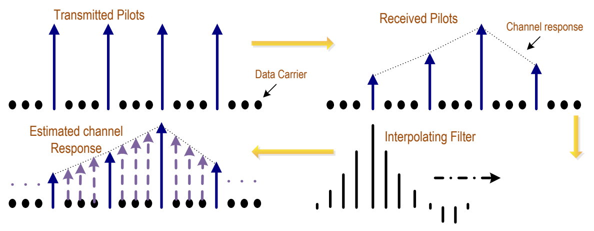


Fig. 8. Channel Estimation based on Pilot-Assisted Linear Interpolation.

4.4 Channel Estimation

Transmitted data symbols may get distorted after passing through the wireless channel and prior to reaching the receiver antenna due to the various impairments. Channel estimation is the task of estimating channel frequency response which has to be performed subsequent to recovering data symbols in the demodulator block. One of the methods for executing the channel estimation is pilot-assisted linear interpolation algorithm (depicted in Fig. 8) by using the pilots which are added in the transmitter side and known for the receiver.

Let's assume the Y_n is the received data symbols, it can be expressed as

$$Y_n = X_n H_n + N_n \tag{15}$$

where n, H_n and N_n stand for the number of subcarriers, channel impulse response and additive noise, respectively. As the first step, the H_n has to be estimated and then, Y_n should be corrected according to X_n [27]. Based on the IEEE 802.11a/g specifications, there are four pilots inserted between data subcarriers while the place as well as their values are known for the receiver. The channel impulse response can be calculated using the following equation

$$\tilde{H_k} = M^{-1} P_{Rx} \tag{16}$$

where k, P_{Rx} and $\tilde{H_k}$ are representing the number of pilots, the received noise-impaired pilots and the channel impulse response of the received pilots, respectively. M also stands for a diagonal matrix formed from transmitted pilots which is expressed as Eq. 17.

$$M = \begin{bmatrix} M_{1,1} & 0 & \cdots & 0 \\ 0 & M_{2,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & M_{k,k} \end{bmatrix} \tag{17}$$

The complex multiplication between Received Pilots (RP) and the Inverse of Transmitted Pilots (ITP) can be mapped on AVATAR as a second context of channel estimation shown in Fig. 9. The first context is related to loading immediate values for shift operations. As it can be observed, only four columns of PEs are instantiated for this purpose. Once the channel response $\tilde{H_k}$ of the received pilots is computed and stored in the second local memory, the frequency response of the adjacent subcarriers requires to be estimated by means of Linear Interpolation, since in order to perform the channel equalization, the receiver requires to have the frequency response of all the subcarriers. Linear Interpolation

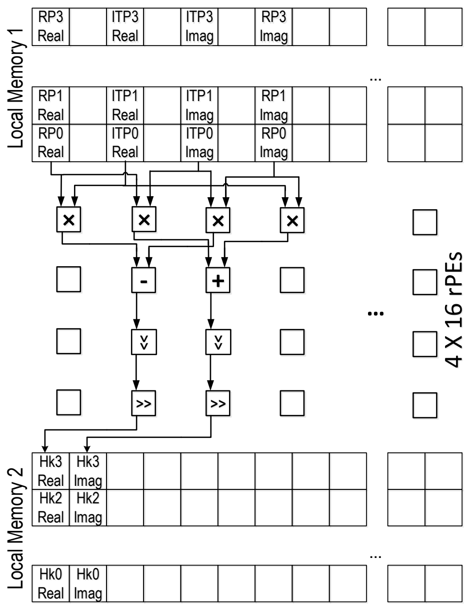


Fig. 9. Second Context for the Channel Estimation.

is a method for approximating the value at each position between two adjacent known values which can be expressed for this case study based on the Eq. 18.

$$\hat{H}_n = \sum_{i=1}^{N_p-1}\sum_{j=1}^{N_s}\tilde{H}_k(i) \underbrace{+}_{3} ((\tilde{H}_k(i+1) \underbrace{-}_{1} \tilde{H}_k(i)) \underbrace{\times}_{2} \underbrace{\frac{j-1}{N_s}}_{\mu}) \tag{18}$$

Here, the channel frequency response for four received pilots $\tilde{H}_k$ will get expanded by using the step size μ in order to estimate the channel frequency response for other subcarriers located around pilots $\tilde{H}_n$. The number of pilots and samples are also represented by N_p and N_s, respectively. The above equation can also be mapped on AVATAR as a third context shown in Fig. 10. First of all, both real and imaginary parts of the pilots have to be loaded into the first local memory along with the step size which has 16 different fixed values calculated by using RISC processor software. Eq. 18 requires three steps to be computed completely which are instantiated by dashed border circles in Fig. 10. Subsequent to the completion of linear interpolation, the calculated channel frequency response of all subcarriers should be stored in the second local memory for further processing.

As the next step of channel estimation block, channel equalization is required to be performed which is the task of refining the received noisy data symbols Y_n as close as possible to X_n. For this purpose, the received data symbols should be divided by the computed channel frequency response as expressed in Eq. 19.

$$\hat{Y}_n = \frac{Y_n}{\hat{H}_n} \tag{19}$$

The division operation can be performed by using an iteration algorithm called Newton-Raphson method [28]. Originally, this method can be used for seeking the root of an equation. As an example, the first approximation of the root of a given function *f(x)* can be obtained by using the following equation

$$x_{n+1} = x_n + \frac{f(x_n)}{f'(x_n)}, \tag{20}$$

where n is the number of iteration, and x_n is the initial guess of the root and $f'(x_n)$ is derivative of a function $f(x_n)$. Newton-Raphson method can also be modified to be used for other purposes such as division operation based on the following equation.

$$x_{n+1} = x_n.(2 - Dx_n) \tag{21}$$

Let us assume that the result of $\frac{1}{D}$ is going to be found by using a function *f(x)* which has a zero value at $x = \frac{1}{D}$. The function can be written as $f(x) = \frac{1}{x} - D$ and expressed as Eq. 21 where D and x_n stand for the denominator and initial guess, respectively. In this case study, the denominator would be a complex number since it is affected by the noisy channel. Thus, it has to be simplified in order to map on the template-based CGRA based on the following equation.

$$\frac{x+iy}{a+ib} \times \frac{a-ib}{a-ib} = \frac{(x+iy)\times(a-ib)}{a^2+b^2} \tag{22}$$

Here $x+iy$ is representing received data symbols from FFT block and $a+ib$ and $a-ib$ stand for estimated channel response and its complex conjugate, respectively. As it can be seen from the left part of Fig. 10, the first step of Newton-Raphson method related to the computation of $\frac{1}{a^2+b^2}$ can be mapped on AVATAR where D is equivalent to $(a^2 + b^2)$ based on (21) and a and b stand for the real and imaginary values, respectively. The first two rows of PEs are specified for computing the square values of the real and imaginary parts of the channel frequency response as well as performing the addition among the achieved square value results. As it is depicted in Fig. 10, the initial guess should be loaded into the local memory of the third context subsequent to the computation of linear interpolation along with every column due to line readability of local memories in CGRAs which makes them simpler and faster. The right part of the fourth context is just used for transferring the results of linear interpolation for further processing over a fifth context.

As the next step shown in Fig. 11, the constant 2 is loaded into the local memory of the fifth context along with the results of the first part of Newton-Raphson method. The left part of the fifth context is composed of the preprocessing of data by using latency, the required shift operations, subtractions and multiplications which are used for performing Eq. 21 completely. At the right side of the fifth context, the demodulated data symbols are multiplied by the complex conjugation of estimated channel frequency response in order to execute the channel equalization. Then during the last context of channel estimation block, the results obtained from Newton-Raphson method are multiplied by the results of calculated complex multiplication. Within this stage, the shift operation is required for preventing data overflow as well as producing the final product. Here Res_i stands for the equalized received data symbols which are ready for the extracting data bits from them within the symbols demapping block which can be implemented by using the RISC processor software.

5 Implementation of Baseband Processing on HARP

The instantiation and implementation of OFDM receiver baseband processing on HARP is for the proof-of-concept, to demonstrate its design and technical capabilities as well as the functionality. All designed accelerators can be mapped on HARP in a way that both master and slave nodes can exchange data with each other. Table 1 shows the size of each template-based CGRA mapped on

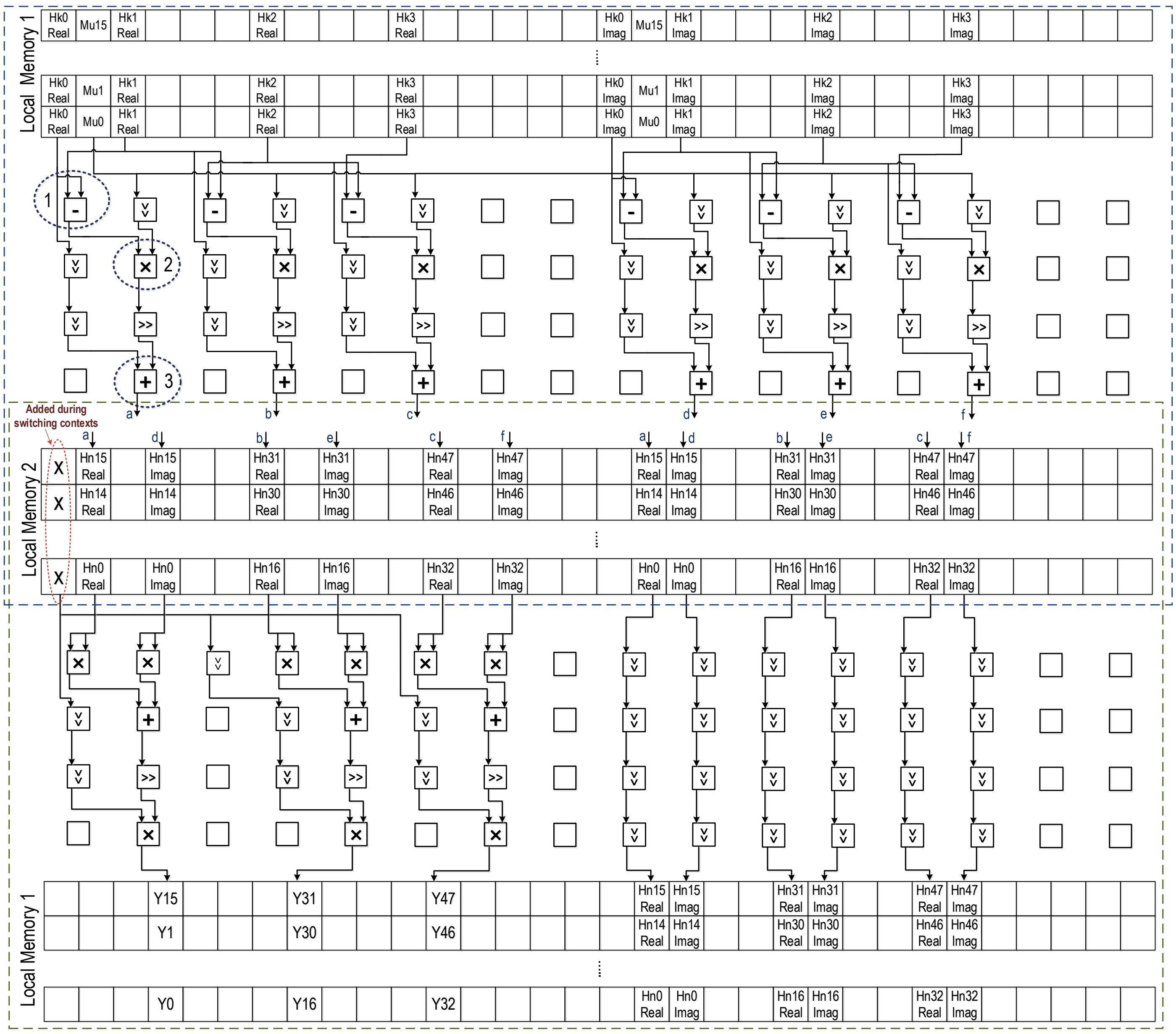


Fig. 10. Third & Fourth Contexts for the Calculation of Linear Interpolation and Newton-Raphson Method.

HARP used for generating application-specific accelerators for the receiver blocks. After mapping the kernels on the designed CGRA accelerators and integration over HARP, the overall architecture processing OFDM can be depicted as shown in Fig. 12. In the proposed platform, the distributed control for transferring the data as well as execution can be written by the designer in software which has to be compiled for the three instantiated RISC processors.

The order of exchanging the results and data symbols among the master and slave nodes is specified with the numbers and dashed arrows. The RISC cores are responsible for transferring the configuration stream and data to be processed to CGRAs in a way that node N3 RISC is responsible for N0 and N6 CGRAs while N4 RISC is responsible for N1 and N7 CGRAs and N5 RISC is responsible for N2 and N8 CGRAs. Subsequent to the system start-up and transfer of the configuration stream by the three RISC cores, the received data symbols are loaded and then transported to the N0 for performing Time Synchronization block. The Clock Cycles (CC) required for data transfer from data memory of a node to the data memory of another node and to CGRA's local memory as well as processing at different stages are shown in Table 1. As it can be seen, 1982 CC are required for total data transfer from N3 RISC to N0 CGRA specified in Fig. 12 by dashed arrow number 1. Correlation can be performed in 1120 CC. Once the correlation algorithm is performed by N0, the results will get back to N3 for computing the index of the time offset which requires 7590 CC for transferring the data from CGRA to data memory. Calculating the SM and finding the index of time offset can also be executed in 2345 CC by using RISC processor software. At the end of the first stage, the calculated time offset is transmitted in 47 CC to N4 for further processing shown with dashed arrow number 2.

Then the short training symbols have to be loaded to the local data memory of N1 for executing complex multiplication. Loading the data from data memory of N4 to the data memory of N1 and

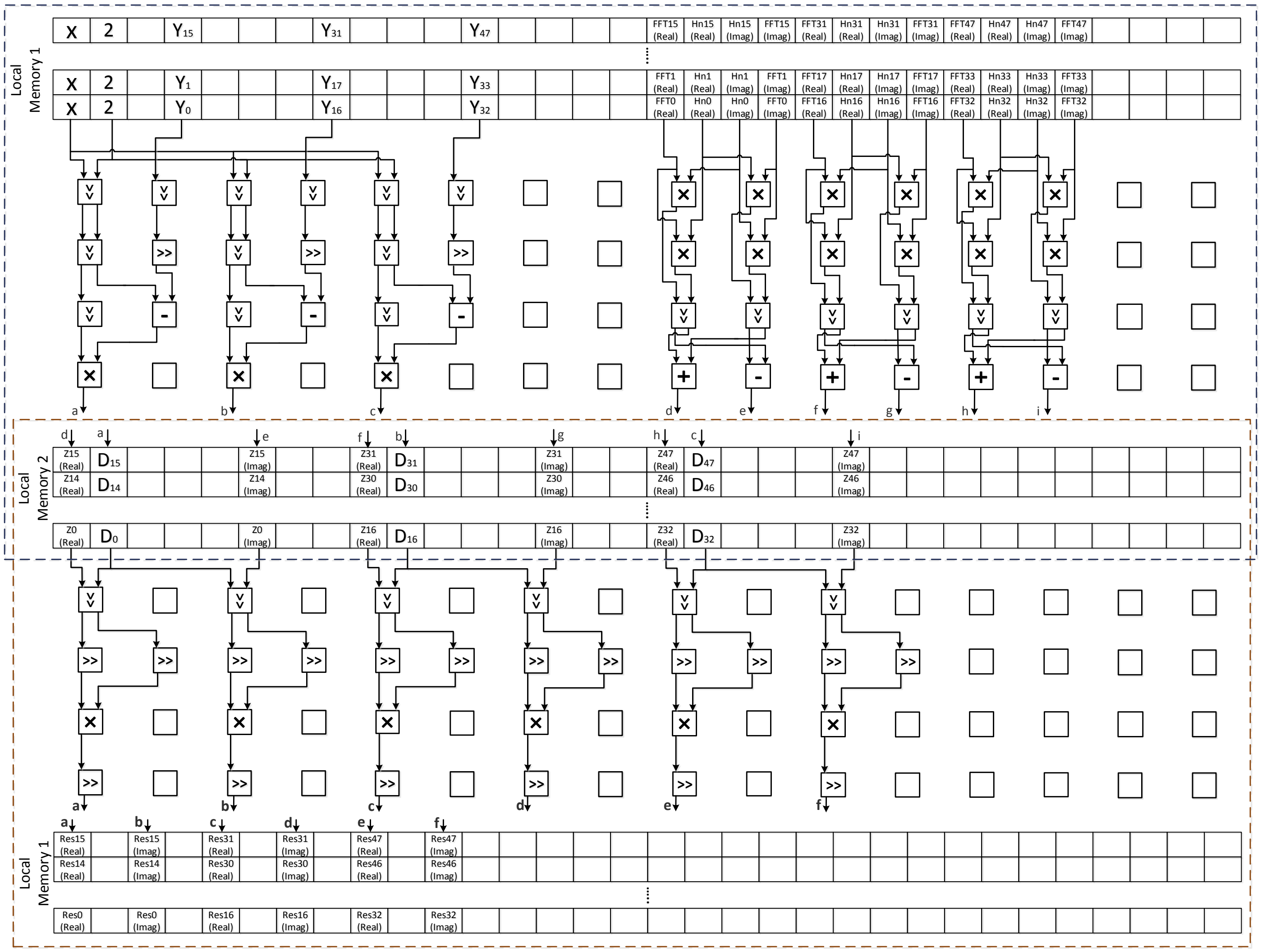
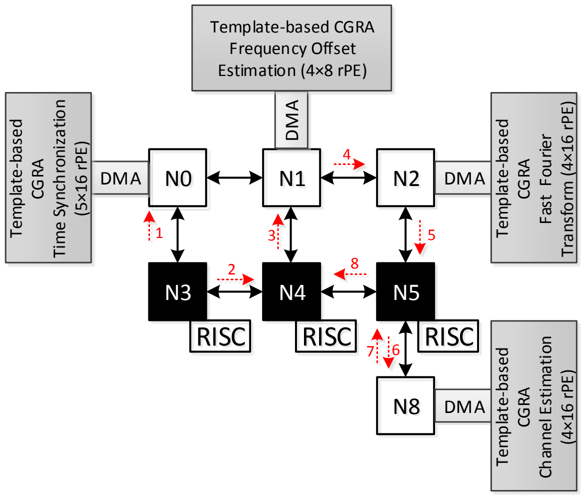


Fig. 11. Fifth & Sixth Contexts for the Calculation of Newton-Raphson Method and Channel Equalization.



Template-based CGRA Frequency Offset Estimation (4×8 rPE)
Template-based CGRA Time Synchronization (5×16 rPE)
Template-based CGRA Fast Fourier Transform (4×16 rPE)
Template-based CGRA Channel Estimation (4×16 rPE)
DMA
N0
N1
N2
N3
N4
N5
N8
RISC
1
2
3
4
5
6
7
8

Fig. 12. Abridged general view of IEEE 802.11a/g receiver on HARP platform. In this figure, the red colored numbers specify the order in which the nodes should be executed. The black and white colored nodes are the supervisor and slave nodes, respectively.

then to the CGRA takes 4664 CC depicted with dashed arrow number 3. Furthermore, complex multiplication can be executed in just 44 CC by using the designed template-based CGRA. As it was mentioned earlier, some parts of the frequency offset estimation block should be performed by using RISC processor software which requires data exchanging between N1 and N4 twice. Thus, the results of complex multiplication have to be returned to the data memory of N4 for performing some parts of frequency offset estimation by the N4 RISC processor. In total, the required execution time for performing frequency offset estimation can be divided into 12634 and 74 clock cycles (40 CC plus 30 CC related to Eq. 10) to be executed by the N4 RISC core and the template-based CGRA mapped on N1, respectively.

Subsequent to the completion of Eq. 10, utilizing the designed context located at N1, the corrected data symbols in terms of the added carrier frequency offset can be transported from the local memory of the N1 to the data memory of N2 (shown with dashed arrow number 4) and then to the local data memory of CGRA by the DMA device in 570 CC and 504 CC, respectively. N2 then computes a 64-point, radix-4 FFT on the results from N1 in 328 CC.

Once the FFT is performed completely, the DMA of N2 transfers the results from the local memory of CGRA to the data

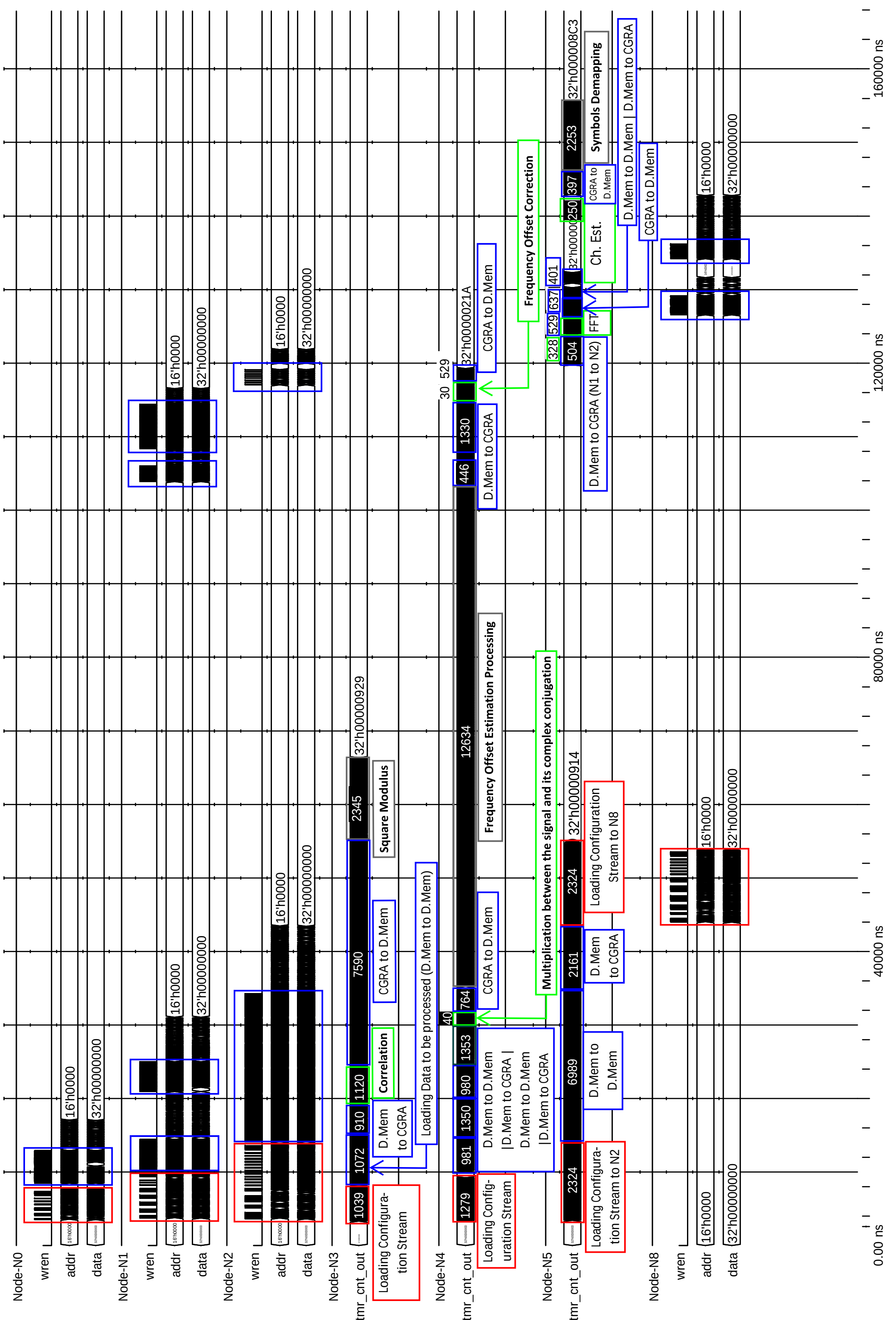


Fig. 13. Digital Waveforms related to the Configuration Streams (Red Lines), Data Transfers (Blue Lines) among the Data Memories of the CGRA-Nodes and the implementation of the receiver blocks (Green and Gray Lines related to the tasks performed by CGRAs or RISC processors, respectively). The signals 'wren', 'addr' and 'tmr_cnt_out' stand for write-enable, address ports of the Data Memory and the counter for enumerating the number of clock cycles, respectively. The numbers inside the digital waveforms of N3, N4 and N5 are also mentioned in Table I.

memory of N5 (shown with dashed arrow number 5) which acts as a supervisor node for exchanging the data between N2 and N8 in 529 CC. Then the data symbols should be transported to the local data memory of N8 (depicted with dashed arrow number 6) for executing the channel estimation which takes 1038 CC in total. Subsequent to the completion of channel estimation by N8

TABLE 1
Clock cycles required for data transfer and processing at different stages. In the table, D. Mem, Trans and Exe. stand for Data memory, Transfer and Execution, respectively. Clock cycles with * sign represent data transfer from CGRA to Node's data memory.

Node-to -Node	D. Mem to D. Mem	D. Mem to CGRA	Trans. Total	Exe. Total
N3-N0 (Correlation)	1072	910	1982	1120
N0-N3 (SM)	-	7590*	-	2345
N3-N4	47	-	-	-
N4-N1	1961	2703	4664	12634 (+74)
N1-N2	-	570* + 504	1074	328
N2-N5	-	529*	-	-
N5-N8	637	401	1038	250
N8-N4	-	397*	-	2253

in 250 CC, the final results have to be transferred back to the data memory of N4 (dashed arrows 7 & 8) for performing symbols demapping as a last stage which is to be performed by RISC processor software.

In order to speed up the execution time of the overall platform as well as to demonstrate the simultaneous execution by the nodes of HARP, the configuration streams of all slave nodes are loaded in parallel. However, due to the data dependency between different blocks of the receiver, each node has to wait until the other one completes its execution. In the current instance of the platform, the nodes N6 and N7 are not instantiated with any CGRA as they are not required by our particular test-case. Although a CGRA can be reconfigured to perform more than one task, it will make the critical path longer and therefore reducing the operating frequency. Moreover, the target platform for testing was highly dense FPGA device and we preferred to increase resources and not to compromise the speed.

The number of clock cycles required for implementing each block by CGRA in addition to the data transfer between the data memories is shown in Table 1. Three kinds of data transfers can be observed from the table. The first one is related to the data transfer from the data memory of RISC cores to the data memory of the CGRAs. The second one is about the data transfer from the data memory of CGRA to the local memory banks. The third one is data transfer within a slave node by using a DMA device. The total execution time for time synchronization, frequency offset estimation, FFT and channel estimation blocks are equal to 3809, 12708, 420 and 250 clock cycles, respectively. The symbols demapping can be performed by RISC processor software in 2253 clock cycles. Furthermore, the percentage of the time for the useful computation of the nodes N0, N1, N2 and N8 which consist of CGRAs without considering the time for transferring the data or configuration streams is equal to 3.61%, 0.22%, 1.05% and 0.8%, respectively.

The time frame in which the NoC bandwidth is utilized for transferring the configuration stream and data between the data memories of the CGRAs can be observed from Fig. 13. The configuration transfer is shown in red, data exchange between CGRAs in blue and execution is shown in neon colored boxes. The reference measurements are in terms of COFFEE RISC core clock cycles. The operating frequency at which the RISC cores and the entire platform is simulated is 200.0 MHz. Since there are three RISC cores on the platform as can be observed from Fig. 12, each RISC core has its own counter for general-purpose measurements and in the waveform shown with signal name `tmr_cnt_out`.

The waveforms in Fig. 13 show a very detailed view of the implementation and functioning of OFDM kernels in a fixed time frame. Apparently, the frame shows a large space without any signal activity and Frequency Offset Estimation Processing as the largest time consuming block (Worst-Case). Considering this situation, the configuration and data transfer frequency can be scaled down with reference to the worst-case without compromising system throughput. This would save instantaneous power dissipation as the frequency of memory accesses will reduce - a major factor to overall power consumption by the system. In cases, where there are several kernels running on different components of the system, self-aware computing can play a key role in identifying the worst-case and mitigating the operating frequencies of the all the system components so that the computational workload is uniformly balanced [29]. The power dissipation of current version of HARP can be mitigated by using self-aware computing to dynamically control the voltage and operating frequency of each core with an adaptive feedback control system, as presented in [30]. Efforts will be made to replace the dark part of the chip with reconfigurable accelerators which can be clocked at relatively lower frequency, therefore reducing the on-chip power dissipation density.

The waveform also shows a very computationally intensive task, i.e., FFT being computed in a very small fraction of time relatively. A designer may want to employ a large number of computational resources to demanding algorithm like FFT and eventually find it over-performing relatively in the whole design space. In this particular case of implementation, a large CGRA like AVATAR containing 5×16 PEs and processing FFT appears to be an expensive choice.

Designed CGRAs for performing OFDM receiver are crafted based on the algebraic equations in the most optimum way since most of the PEs are employed in each context. Optimal mapping of an application is important at design-time in order to improve the performance, area utilization, power and development time which requires scaling up or down a CGRA. Besides the mapping of the applications on the CGRAs, near optimal solution require some proceedings such as maintaining self-aware Dynamic Voltage and Frequency Scaling (DVFS), reconfiguration, scalability in computational resources, modularity and regularity in the architecture.

6 Measurements and Estimations

The overall platform is synthesized for a Stratix-V (5SGXEA4H1F35C1) FPGA device for prototyping purposes. The operating conditions are selected for 0°C and 85°C as low and high junction temperature, respectively and 23 mm heat sink with 200 LFPM airflow as a preset cooling solution. The achieved operating frequencies after placement and routing are equal to 182.32 MHz and 170.3 MHz at 0°C and 85°C, respectively for slow timing model (900 mV). The fast timing model (900 mV) shows 258.73 MHz at 0°C and 235.85 MHz at 85°C. The overall platform works on a single clock source.

Node-by-node breakdown of resource utilization summary for the proposed platform is depicted in Table 2 in terms of the number of Adaptive Logic Modules (ALMs), Registers, Memory Bits and DSP elements. Around 62% of the resources are consumed by the design. Furthermore, the total number of 18-bit DSPs resources

TABLE 2
Node-by-node Breakdown of Resource Utilization Summary for Stratix-V (5SGXEA4H1F35C1) FPGA device.

Node	ALMs	Registers	Memory Bits	(32-bit Multipliers) DSPs
N0	22,616	9,471	2,633,472	(20) 40
N1	8,171	7,985	2,365,736	(16) 32
N2	22,809	11,583	2,635,136	(28) 56
N3	5,436	5,648	3,145,728	(6) 12
N4	5,505	5,716	3,145,728	(6) 12
N5	5,442	5,650	3,145,728	(6) 12
N8	25,908	16,399	2,633,144	(33) 66
NoC	2,842	4,371	-	-
Total	98,729 62%	66,823 11%	19,704,672 51%	(115) 230 90%

utilized is 230 (90%) which depends on the number of 32-bit multipliers instantiated. Each 32-bit multiplier instantiated in a PE requires two 18-bit DSP elements to be synthesized on the FPGA. The breakdown of the number of 32-bit multipliers used for each block of the receiver and a RISC on an NoC node is also given on Table 2.

The platform's power dissipation is estimated based on post placement and routing (post P&R) information using PowerPlay Power Analyzer Tool of Quartus II 15.0 at an ambient temperature of 25°C and at an operating frequency of 200.0 MHz. The overall estimation process yielded 'HIGH' level of confidence. The power estimates were achieved by simulating the gate-level netlist of the platform and then generating the Value Change Dump (VCD) file by using ModelSim software. The VCD file contains signal transition activity information during the execution of OFDM receiver [31]. The dynamic power is because of the signal switching activity of the design for the entire run-time duration while the static power, the power of the whole FPGA chip, is required to keep the device in the ON state. The tool estimated 1243.84 mW, 2623.72 mW and 27.37 mW as static, dynamic and I/O power dissipation, respectively. Therefore, a total power dissipation of the FPGA with the OFDM platform was 3894.93 mW. Node-by-node breakdown of dynamic power and energy consumption of the system is shown in Table 3. It can be observed from the table that the dynamic power dissipation increases as the size of CGRA increases and vice-versa. In this case study, AVATAR integrated in N0, N2 and N8 requires almost 1.5X-2X dynamic power consumption compared to the CREMA used in N1. Although the static power will be increased for just a few mW by scaling up the CGRA, it appears to be almost the same for all CGRAs since the large portion of the FPGA chip remains unused, thus adding a large offset to all the static power estimations. The static power is essentially characteristic to a particular FPGA chip. The energy consumption for each node is also calculated as a product of power dissipation and execution time.

The current instance of the platform is composed of 240 PEs. By considering the operating frequency of 200 MHz and total power dissipation of 3894.93 mW, this instance of HARP delivers a performance of 48 Giga Operations Per Second (GOPS) and 0.012 GOPS/mW for Altera Stratix-V chip in 28 nm. A few instances of HARP when instantiated with CGRAs of varying sizes and different number of cores also yielded 0.012 GOPS/mW of performance ([10], [11]). We can consider this as an **architectural constant** for the HARP template on Stratix-V FPGA and establish comparisons with other platforms shown in Table 4. It is the scalability and regularity in HARP architecture that ensures application-independent figure of merit. Regarding the utilization rate of PEs, in order to implement the whole OFDM receiver, 2914 operations have been performed by 240 instantiated PEs in 9.845 μs which results to 295.98 MOPS. Therefore, the utilization rate of PEs would be equal to 0.61% (295.98 MOPS / 48 GOPS). This makes the platform an excellent candidate for alleviating Dark Silicon issues.

7 Evaluation and Comparisons

Since each platform has been synthesized for a specific technology, cross-technology comparisons have to be conducted with acceptable accuracy by scaling the process sizes as well as analyzing the performance gaps. It should be mentioned that a direct comparison between two different technologies is difficult and the following results are only indicative. In Stratix devices, scaling from 40 nm and 90 nm to 28 nm will increase the synthesis frequency by 20% and 40%, respectively [32]. Regarding the performance gap between FPGAs and ASICs, a 90 nm ASIC implementation shows a speed-up of 4X on average compared to a 90 nm FPGA implementation while requiring 14.0X lower dynamic power [33]. However, the factor of 14 for the total power dissipation can be decreased to almost 2 considering the worst-case scenario if the static power is almost equal to the dynamic power dissipation [33]. In order to estimate the performance from 90 nm ASIC to 28 nm FPGA, the platform's value should be multiplied by a scaling factor of 60%.

As it can be observed, the homogeneous MPSoC called NineSilica [5], requires 10.3 μs in order to execute 64-point radix-4 FFT while the same task can be computed in HARP in 2.1 μs which shows 3.9X speed-up after performance scaling. Regarding the resource utilization, each of NineSilica and HARP requires 71,679 ALUTs on Stratix-IV device and 102,637 ALMs on Stratix-V device, respectively. Since each ALM on Stratix-V device is equivalent to two ALUTs on Stratix-IV device, HARP shows the cost of 2X logic resources against homogeneous MPSoC.

Another homogeneous MPSoC synthesized on a 90 nm FPGA device [14] delivers a performance of 19.2 GOPS. Subsequent to scaling the performance from 90 nm FPGA to the 28 nm FPGA by a factor of 40%, the value is increased to 26.88 GOPS. Compared to the scaled platform's value, HARP platform shows a gain of 1.78X.

The platform P2012 [6] has been synthesized using 28 nm CMOS technology. Considering the equality between the static and dynamic power as the worst case, the scaled performance of P2012 on a 28 nm FPGA is estimated to be 0.005 GOPS/mW where HARP shows a performance gain of 2.4X.

Each of ADRES [15] and MORPHEUS [8] platforms present the performance of 0.004 GOPS/mW and 0.02 GOPS/mW at 90 nm CMOS which can be scaled to a 90 nm FPGA as shown in Table 4. After scaling to a 90 nm FPGA and multiplying by a 60% speed-up, the HARP's performance shows a gain of 15X and 3X in comparison to the scaled performance of ADRES and MORPHEUS, respectively.

Regarding the throughput of our OFDM implementation by using HARP platform and based on 200 MHz clock frequency,

TABLE 3
Dynamic power and energy estimation of each CGRA node and the NoC.

Node	Accelerator Type	Dynamic Power (mW)	Active Time (μs)	Dynamic Energy (μJ)
N0	Time Synchronization	414.35	7.32	3.03
N1	Frequency Offset Estimation	272.23	0.37	0.1
N2	FFT	526.07	1.64	0.86
N3	General Purpose Processing, Synchronization, Control	114.47	141.17	16.15
N4		113.82		16.06
N5		114.52		16.16
N8	Channel Estimation	448.01	1.25	0.56
NoC	-	10.10	-	-
Integration Logic	- -	609.07	- -	-
Total	-	2623.72	-	53.16

TABLE 4
Performance comparisons based on GOPS and GOPS/mW with HARP implementation at 200.0 MHz on 28 nm FPGA. SU_fTfs, SD_aTfs and Ps stand for Speed-Up for FPGA to FPGA scaling, Speed-Down for ASIC to FPGA scaling and Power scaling, respectively.

Platform Technology	Performance Metric	Platform's Value (PV)	Scaled PV	HARP's Value	Gain
NineSilica [5] FPGA 40 nm	FFT Execution Time (μs)	10.3	Exe. Time - Exe. Time × 20% SU_fTfs = 10.3 - 10.3 × 0.2 = 8.24	2.1	3.9×
[14] FPGA 90 nm	GOPS	19.2	PV×40% SU_fTfs = 19.2×1.4= 26.88	48	1.78×
P2012 [6] CMOS 28 nm	GOPS/mW	0.04	PV×SD_aTfs×Ps = 0.04×1/4×1/2= 0.005	0.012	2.4×
ADRES [15] CMOS 90 nm	GOPS/mW	0.004	(PV×SD_aTfs×Ps)×60% SU_fTfs = (0.004×1/4×1/2)×1.6= 0.0008	0.012	15×
MORPHEUS [8] CMOS 90 nm	GOPS/mW	0.02	(PV×SD_aTfs×Ps)×60% SU_fTfs = (0.02×1/4×1/2)×1.6= 0.004	0.012	3×

the achieved value is equal to 17 Mbit/s which comes from the number of data bits extracted at the output of the receiver (192 data bits) and the execution time of Symbols Demapping block. The required throughput (data rate) in IEEE 802.11a/g for 16-QAM modulation is 48 Mbit/s, including the coding bits. Therefore, as a result of having rather low clock frequency in the employed FPGA device used for synthesizing the HARP platform, we have not met the required throughput. However, by changing the device to one with at least 3.0X higher clock frequency, we can meet the required throughput set by the standard. It should be noted that the minimum clock frequency required for us to meet the throughput defined by the standard is not high and can be found even in commercial off-the-shelf low cost components.

8 Conclusion

In this paper, Orthogonal Frequency-Division Multiplexing (OFDM) receiver blocks are implemented as designs for test to completely verify the functional and architectural characteristics of a heterogeneous multicore platform like HARP. The signal processing required by OFDM receiver contains computationally-intensive tasks of both parallel and serial nature, hence are potentially competent candidates to explore almost all the design features of HARP. The designed receiver over HARP is prototyped for a Field Programmable Gate Array (FPGA) device at an operating frequency of 200.0 MHz and at the room temperature of 25°C. HARP delivers a performance of 48 GOPS and 0.012 GOPS/mW which considered as its architectural constant due to the scalability and regularity in HARP architecture that ensures application-independent figure of merit. It shows a speed-up of 3.9X for 64-point radix-4 FFT at the cost of 2X additional resource utilization in comparison to NineSilica which is a general-purpose homogeneous Multi-Processor System-on-Chip (MPSoC). HARP performs at 48 GOPS which showing a gain of 1.78X in comparison to the scaled performance of the MPSoC designed with a Multiple-Instruction Multiple-Data approach. Furthermore, HARP when compared against the scaled performance levels of P2012, ADRES and MORPHEUS in terms of GOPS/mW, shows a performance gain of 2.4X, 15X and 3X, respectively. The simulation and synthesis results and also comparisons against other state-of-the-art platforms demonstrate the benefits of maximizing the number of reconfigurable processing resources on a platform as the integrated CGRAs can be scaled to very large dimensions. The exploration has provided important insights into the architecture and based on those, we recommend that, (1) in a heterogeneous multicore architecture like HARP, mitigate the signal transition activity over the entire platform with reference to the worst-case performing core for power conservation. In this context, self-aware computing models can be helpful, which will most likely result in the investigation of Dark Silicon. (2) At design time, allocate computational resources such that a core does not over or under perform relative to the overall execution time frame as in case of Fast Fourier Transform processing in this experimental work. (3) Metrics like Operations per Watt (OPS/W) are architectural constants and do not change by scaling the computational resources of a platform.

Acknowledgment

This research work is conducted at the Laboratory of Electronics and Communications Engineering, Tampere University of Technology, Finland. This work was partially funded by the Academy of Finland under contract # 258506 (DEFT: Design of a Highly-parallel Heterogeneous MP-SoC Architecture for Future Wireless Technologies), TUT Graduate School, Tuula and Yrjö Neuvo fund, HPY Research Foundation. Tekniikan Edistämissäätiö (TES) Foundation and HiPEAC Collaboration Grant.

References

[1] G. Venkatesh, J. Sampson, N. Goulding, S. Gracia, V. Bryksin, J. L. Martinez, S. Swanson and M. B. Taylor, "Conservation cores: reducing the energy of mature computations", ASPLOS 10, pp. 205218, 2010.

[2] A. Pedram, S. Richardson, S. Galal, S. Kvatinsky and Mark Horowitz, "Dark Memory and Accelerator-Rich System Optimization in the Dark Silicon Era," in IEEE Design & Test , vol.PP, no.99, pp.1-1 doi: 10.1109/MDAT.2016.2573586.

[3] M. B. Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffman, P. Johnson, Jae-Wook Lee, W. Lee, A. Ma, A. Saraf, M. Seneski, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, A. Agarwal, "The Raw microprocessor: a computational fabric for software circuits and general-purpose programs," Micro, IEEE, vol. 22, no. 2, pp. 25,35, Mar/Apr. 2002.

[4] M.B. Taylor, Is dark silicon useful?: harnessing the four horsemen of the coming dark silicon apocalypse. In proceedings of the 49th Annual Design Automation Conference (DAC 12) (pp. 11311136). NY, USA: ACM.

[5] R. Airoldi, F. Garzia, O. Anjum, and J. Nurmi, "Homogeneous MPSoC as baseband signal processing engine for OFDM systems", International Symposium on System on Chip (SoC), 2010, pp. 26-30, Sept. 2010, doi: 10.1109/ISSOC.2010.5625562.

[6] D. Melpignano, L. Benini, E. Flamand, B. Jego, T. Lepley, G. Haugou, F. Clermidy and D. Dutoit, "Platform 2012, a Many-Core Computing Accelerator for Embedded SoCs: Performance Evaluation of Visual Analytics Applications", in Proc. 49th Annual Design Automation Conference (DAC '12). ACM, New York, NY, USA, 1137-1142.

[7] F. Conti, C. Pilkington, A. Marongiu and L. Benini, "He-P2012: architectural heterogeneity exploration on a scalable many-core platform", in Proc. of the 24th edition of the great lakes symposium on VLSI (GLS-VLSI '14), pp. 231-232, ACM, New York, NY, USA.

[8] N. S. Voros, M. Hbner, J. Becker, M. Khnle, F. Thomaitiv, A. Grasset, P. Brelet, P. Bonnot, F. Campi, E. Schler, H. Sahlbach, S. Whitty, R. Ernst, E. Billich, C. Tischendorf, U. Heinkel, F. Ieromnimon, D. Kritharidis, A. Schneider, J. Knaeblein, and W. Putzke-Rming, "MORPHEUS: A Heterogeneous Dynamically Reconfigurable Platform for Designing Highly Complex Embedded Systems", ACM Trans. Embed. Comput. Syst. 12, 3, Article 70, 33 pages, April 2013.

[9] F. Thoma, M. Kuhnle, P. Bonnot, E. M. Panainte, K. Bertels, S. Goller, A. Schneider, S. Guyetant, E. Schuler, K. D. Muller-Glaser, and J. Becker, "MORPHEUS: Heterogeneous Reconfigurable Computing", International Conference on Field Programmable Logic and Applications, FPL 2007, pp. 409-414, 27-29 Aug. 2007.

[10] W. Hussain, R. Airoldi, H. Hoffmann, T. Ahonen and J. Nurmi, "Design of an Accelerator-Rich Architecture by Integrating Multiple Heterogeneous Coarse Grain Recongurable Arrays over a Network-on-Chip", in Application-specific Systems, Architectures and Processors (ASAP), 2014 IEEE 25th International Conference on, pp.131-138, 18-20 June 2014.

[11] W. Hussain, R. Airoldi, H. Hoffmann, T. Ahonen and J. Nurmi, "HARP2: An X-Scale Reconfigurable Accelerator-Rich Platform for Massively-Parallel Signal Processing Algorithms", in the Springer's Journal of Signal Processing Systems, 2015.

[12] F. Garzia, W. Hussain and J. Nurmi, "CREMA, A Coarse-Grain Reconfigurable Array with Mapping Adaptiveness", in Proc. 19th International Conference on Field Programmable Logic and Applications (FPL 2009). Prague, Czech Republic: IEEE, September 2009.

[13] W. Hussain, F. Garzia and J. Nurmi, "Designing Fast Fourier Transform Accelerators for Orthogonal Frequency-Division Multiplexing Systems", Journal of Signal Processing Systems, Springer, ISSN 1939-8018, Vol 69, pp. 161-171, December, 2012.

[14] P. Bonnot, F. Lemonnier, G. Edelin, G. Gaillat, O. Ruch, P. Gauget, "Definition and SIMD implementation of a multi-processing architecture approach on FPGA". In Proc. of Design, Automation and Test in Europe (DATE '08). ACM, New York, NY, USA, 610-615.

[15] B. Mei, S. Vernalde, D. Verkest, H. D. Man, and R. Lauwereins, "ADRES: An architecture with tightly coupled VLIW processor and coarse-grained reconfigurable matrix", Field-Programmable Logic and Applications, vol. 2778, pp. 61-70, ISBN 978-3-540-40822-2, September 2003.

[16] F. Bouwens, M. Berekovic, A. Kanstein, G. Gaydadjiev, P. Diniz, E. Marques, K. Bertels, M. Fernandes, and J. Cardoso, "Architectural Exploration of the ADRES Coarse-Grained Reconfigurable Array", Reconfigurable Computing: Architectures, Tools and Applications, Lecture Notes in Computer Science, Springer Berlin / Heidelberg, vol.4419, pp. 1-13, ISBN: 978-3-540-71430-9, 2007.

[17] F. Campi, A. Deledda, M. Pizzotti, L. Ciccarelli, P. Rolandi, C. Mucci, A. Lodi, A. Vitkovski, L. Vanzolini, "A dynamically adaptive DSP for heterogeneous reconfigurable platforms". In Proc. of Design Automation and Test in Europe (DATE '07). EDA Consortium, San Jose, CA, USA,9-14.

[18] T. Ahonen and J. Nurmi, "Hierarchically heterogeneous network-on-chip", in Proceedings of the 2007 International Conference on Computer as a Tool (EUROCON07), pp. 25802586, IEEE, 9-12 September 2007. ISBN: 978-1-4244-0813-9, DOI:0.1109/EURCON.2007.4400469.

[19] J. Kylliäinen, T. Ahonen, and J. Nurmi, "General-purpose embedded processor cores - the COFFEE RISC example", In Processor Design: System-on-Chip Computing for ASICs and FPGAs, J. Nurmi, Ed. Kluwer Academic Publishers / Springer Publishers, ch. 5, pp. 83-100, ISBN: 978-1-4020-5530-0, June 2007, DOI: 10.1007/978-1-4020-5530-0_5.

[20] C. Brunelli, F. Garzia, C. Giliberto, and J. Nurmi, "A Dedicated DMA Logic Addressing a Time Multiplexed Memory to Reduce the Effects of the System Buss Bottleneck", in Proc. 18th International Conference on Field Programmable Logic and Applications, (FPL 2008), Heidelberg, Germany, pp. 487-490, 8-10 September 2008.

[21] F. Garzia, C. Brunelli and J. Nurmi, "A pipelined infrastructure for the distribution of the configuration bitstream in a coarse-grain reconfigurable array", in Proceedings of the 4th International Workshop on Reconfigurable Communication-centric System-on-Chip (ReCoSoC'08). Univ Montpellier II, July 2008, pp. 188-191, ISBN:978-84-691-3603-4.

[22] J. Heiskala and J. Terry, "OFDM Wireless LANs: A Theoretical and Practical Guide", Copyright ©2002 by Sams Publishing, SAMS, 201 West 103rd St., Indianapolis, Indiana, 46290 USA.

[23] J.-J. van de Beek, P.O. Borjesson, M.-L. Boucheret, D. Landstrom, J.M. Arenas, P. Odling, C. Ostberg, M. Wahlqvist, and S.K. Wilson, "A time and frequency synchronization scheme for multiuser OFDM", IEEE Journal on Selected Areas in Communications, vol.17, no.11, pp.1900-1914, Nov. 1999.

[24] S. Nouri, W. Hussain and J. Nurmi, "Design and Evaluation of Correlation Accelerator in IEEE-802.11a/g Receiver using a Template-based Coarse-Grained Reconfigurable Array," in IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP & International Symposium on System-on-Chip (SoC), 2015, pp.1-6, 26-28 Oct. 2015.

[25] J. E. Volder, "The CORDIC Trigonometric Computing Technique", IRE Transactions on Electronic Computers, pp. 330-334, September 1959.

[26] Cooley, J. W., & Tukey, J. W. (1965). An algorithm for the machine calculation of complex Fourier series. Mathematics of Computation, 19, 297-301.

[27] Man-On Pun, M. Morelli, and C-C Jay Kuo, "Multi-carrier techniques for broadband wireless communications : a signal processing perspective", copyright ©2007 by Imperial College Press, December 2007.

[28] V. S. Ryaben'kii and S. V. Tsynkov, "A Theoretical Introduction to Numerical Analysis", CRC Press, p. 243, 2006.

[29] H. Hoffmann and et al.,"Self-aware computing in the Angstrom processor", 49th IEEE Design Automation, 2012.

[30] W. Hussain, H. Hoffmann, T. Ahonen and J. Nurmi, "Power Mitigation by Performance Equalization in a Heterogeneous Reconfigurable Multicore Architecture", Journal of Signal Processing Systems, 2016. doi=10.1007/s11265-016-1142-5

[31] "Design Implementation and Optimization Quartus-II Handbook Version 13.1", Vol. 2, Altera Corporation, May 2013.

[32] "Altera Product Catalog. 2015", Release Date: July 2014, Version 15.0, page 2, www.altera.com.

[33] K. Ian and J. Rose, "Measuring the Gap Between FPGAs and ASICs", Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on , vol.26, no.2, pp.203,215, Feb. 2007.

Sajjad Nouri is working as a Researcher and studying as a Doctor of Technology student in Laboratory of Electronics and Communications Engineering, Tampere University of Technology (TUT), Tampere, Finland. He has a B.Sc degree in Software Engineering from the University of Guilan, Iran. He received his M.Sc degree with distinction in Information Technology in June 2015 from TUT. His research work contains design and implementation of accelerators specialized for Software Defined Radio (SDR) applications by using different template-based Coarse-Grained Reconfigurable Arrays (CGRAs) and mapping their VHDL model onto the FPGA for evaluating the designed accelerators in terms of different performance metrics. He is also working on Accelerator-Rich Architectures and Heterogeneous Multicore Platforms in order to maximize the number of computational resources for exploiting the un-utilized part of the chip which is known as Dark Silicon.

Waqar Hussain is currently working as Principal Researcher at ARM, Norway and as a Visiting Researcher at the Department of Computer Science, Norwegian University of Science and Technology, Trondheim, Norway. Previously, he has worked as a Researcher and later as Postdoctoral Researcher at the Department of Electronics and Communications Engineering, Tampere University of Technology, Finland. He also has been a Visiting Scientist at the Department of Computer Science, University of Chicago, IL, USA and also at the Chair for Integrated Signal Processing Systems, Rheinisch-Westfaelische Technische Hochschule (RWTH), Aachen, Germany. His research interests include design and development of homogeneous and heterogeneous multicore and manycore systems that are specialized for application-specific, reconfigurable and general-purpose processing. Dr. Hussain is in the technical program committee of several high-level conference and reviews a number journal articles every year. He is the first co-author of the book, Computing Platforms for Software-Defined Radio published by Springer International Publishing. He hold a Doctor of Technology degree from Tampere University of Technology, Finland.

Jari Nurmi D.Sc.(Tech) Jari Nurmi works as a Professor at Tampere University of Technology, Finland since 1999, in the Faculty of Computing and Electrical Engineering. He is working on embedded computing systems, System-on-Chip, wireless localization, positioning receiver prototyping, and software-defined radio. He held various research, education and management positions at TUT since 1987 (e.g. Acting Associate Professor 1991-1994) and was the Vice President of the SME VLSI Solution Oy 1995-1998. Since 2013 he is also a partner and co-founder of Ekin Labs Oy, a research spin-off company, now headquartered in Silicon Valley as Radiomaze, Inc. He has supervised 19 PhD and over 130 MSc theses at TUT, and been the opponent or reviewer of 33 PhD theses for other universities worldwide. He is a senior member of IEEE, and member of the technical committee on VLSI Systems and Applications at IEEE CAS. In 2004, he was one of the recipients of Nokia Educational Award, and the recipient of Tampere Congress Award in 2005. In 2011 he received IIDA Innovation Award, and in 2013 the Scientific Congress Award and HiPEAC Technology Transfer Award. He is a steering committee member of four international conferences (chairman in two). He has edited 5 Springer books, and has published over 350 international conference and journal articles and book chapters.