HW/SW Co-design of an IEEE 802.11a/g Receiver on Xilinx Zynq SoC using High-Level Synthesis

Sajjad Nouri*, Jens Rettkowski†, Diana Göhringer†† and Jari Nurmi*
*Laboratory of Electronics and Communications Engineering, Tampere University of Technology, Tampere, Finland
†Application-Specific Multi-Core Architectures (MCA) Group, Ruhr-Universität Bochum, Bochum, Germany
††Chair for Adaptive Dynamic Systems,Technische Universität Dresden, Germany
Emails: sajjad.nouri@tut.fi, jens.rettkowski@rub.de, diana.goehringer@tu-dresden.de, jari.nurmi@tut.fi

Abstract—**This paper presents an implementation of an Orthogonal Frequency-Division Multiplexing (OFDM) receiver using the high-level synthesis tool, from Xilinx called Software Defined System-on-Chip (SDSoC). The Zynq SoCs containing an ARM processor besides a Field Programmable Gate Array (FPGA) are introduced to improve the system performance and power efficiency. SDSoC provides an embedded C/C++ application programming interface for developing heterogeneous embedded systems. Thus, the OFDM receiver is written in C/C++ code to be realizable on the FPGA and the ARM processor. The OFDM receiver is composed of computationally intensive tasks which requires a HW/SW co-design to fulfill system requirements. In this work, the implementation of OFDM receiver blocks are evaluated on ZC706 board and compared against the Heterogeneous Accelerator-Rich Platform (HARP). Based on the achieved results, the complete execution of the IEEE 802.11a/g receiver shows an overall speed-up 3.49X compared to the HARP platform.**

1. Introduction

The demand for executing various sets of operations by embedded systems as well as the increasing computational performance and rapid development in semiconductor technology have led to the use of Heterogeneous Multicore Architectures (HMAs) that use accelerators. Additionally, Internet-of-Things (IoT) and its future, Internet-of-Everything (IoE), increase significantly the need for heterogeneous platforms since the wide range of technologies belonging to IoT are used in a large spectrum of application domain with its own data models [1]. HMAs can also be a worth candidate for 5G baseband in terms of exploiting parallelism in order to support a large number of connected devices communicating at multiple different standards and different types of data transmission schemes. Another reason which has led to the growth and importance of HMAs is the Dark Silicon issue [2]. It states that on a single chip, all cores cannot be clocked at their maximum operating frequency because of the given thermal design power constraint. Therefore, a large fraction of chip has been forced to be powered-off (dark) or operated at a very low frequency (dim). One of the suggested solutions to combat with this issue is the use of application-specific accelerators instantiated in a HMA for performing computationally intensive kernels.

In this context, Field-Programmable-Gate-Arrays (FPGAs) can be employed for integrating multiple processing elements (PEs) on a single chip to build heterogeneous Multi-Processor Systems-on-Chip (MPSoCs), to be consistent with Section II. Since programming of FPGAs as well as designing application-specific accelerators by using hardware description language might be expensive and complicated, High-Level Synthesis tools such as VivadoHLS [3] and SDSoC (Software-Defined SoC) [4] have been introduced. SDSoC provides an embedded C/C++ application programming interface for developing heterogeneous embedded systems on the Zynq MPSoC and SoC platform. It includes a C/C++ compiler which can compile each thread of an application by a processor in software or in hardware.

The main contribution of this paper is a novel HW/SW codesign of an Orthogonal Frequency-Division Multiplexing (OFDM) receiver using high-level-synthesis. The OFDM receiver contains algorithms of parallel and serial nature, i.e., Fast Fourier Transform (FFT), Correlation, Convolution and Complex Matrix-Vector Multiplication (MVM) which requires hardware/software co-design. This mixture of parallel and serial tasks are computationally intensive and time consuming. The computationally intensive and parallel tasks are implemented into hardware by compiling the functions written in C/C++ code. The functions are optimized by pragmas to be efficiently implemented in the FPGA. Furthermore, in this work, the system is compared to the Heterogeneous Accelerator-Rich Platform (HARP) [5] in terms of performance. The system presented in this paper shows a speedup of 3.49x compared to the HARP system.

This paper is organized as follows. In Section II, related work about OFDM receivers implemented in HMA is presented. Section III explains the algorithms employed at different blocks of the OFDM receiver. Section IV presents the instantiation and implementation of the OFDM receiver baseband processing on the ZC706 evaluation board. In Section V, the experimental results are discussed and compared to the HARP system. In the last section, the conclusion is given.

2. Related Work

Until now, several state-of-the-art heterogeneous multicore platforms have been developed in order to accelerate many specific algorithms and investigate Dark Silicon issue. In the following, some of the most promising heterogeneous platforms are described briefly.

Platform 2012 (P2012) is a homogeneous MPSoC consisting

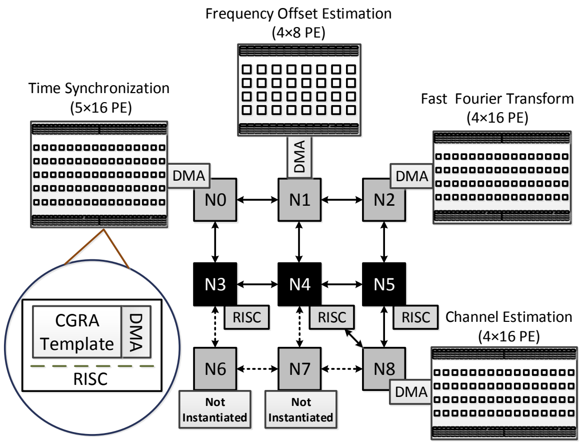

Figure 1. HARP Template, the black colored N3, N4 and N5 are the supervisor nodes containing the RISC processor, the rest of the nodes are slave and can be either RISC or CGRA as a coprocessor; General view of IEEE 802.11a/g receiver on HARP platform.

of four clusters connected by a Network-on-Chip (NoC) architecture [6]. Each cluster contains 16 general-purpose Reduced Instruction Set Computing (RISC) processors which are locally synchronous but globally asynchronous. In addition to the homogeneous version, there is also a heterogeneous extended version of P2012, called He-P2012 [7].

MORPHEUS ([7], [9]) is a heterogeneous dynamically reconfigurable platform comprising fine-grained, middle-grained and coarse-grained reconfigurable devices communicating over a 64-bit NoC. FlexEOS as an embedded FPGA is employed for fine-grained algorithms and constructed from multi-function logic cells. Besides FlexEOS, MORPHEUS has DREAM platform as a middle-grained reconfigurable Digital Signal Processor (DSP) core in order to exploit instruction level parallelism. DREAM consists of a 32-bit RISC core processor and PiCoGA-III reconfigurable datapath. The coarse-grained one is XPP-III which is composed of a dataflow array and Very Long Instruction Word (VLIW) processor as a heterogeneous platform optimized for streaming applications. Software oriented approach was also provided for ease of development and implementation.

HARP is designed as a template-based architecture written in parametrized VHDL and programmable in C language for maximizing the number of computational resources [10]. HARP is composed of nine nodes connected to each other over a NoC in a 3×3 mesh topology, shown in Figure 1. The CGRAs are template-based and scalable. Each CGRA is equipped with R rows × C columns of reconfigurable Processing Elements (rPEs) where the values of 8 or 16 can be assigned to C while the value of R is based on the application requirements and algorithms. The functionality of rPE as well as interconnection among them can be implemented at design time. HARP is designed in a way that designers can instantiate each node either with a CGRA of a specific dimension or a RISC core or even leave it as a data routing resource. The central node is usually considered as a supervisor node which is responsible for transferring data between its own data memory and data memories of the slave nodes. Slave nodes contain a template-based CGRA or RISC core, a data memory and a Direct Memory Access (DMA) device [11]. An example

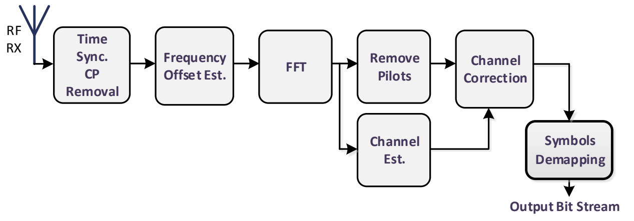

Figure 2. A simplified block diagram of the OFDM receiver.

of application mapping on HARP is also depicted in Figure 1 where an OFDM receiver is designed by crafting template-based CGRAs and then arranged over a NoC structure. As it is reported, HARP delivers a performance of 48 Giga Operations Per Second (GOPS) and 0.012 GOPS/mW for Altera Stratix-V chip in 28 nm. Compared to other state-of-the-art platforms, HARP shows a performance gain of 2.4X against the scaled performance of P2012 on a 28 nm FPGA which is estimated to be 0.005 GOPS/mW. Moreover, HARPs performance shows a gain of 3X in comparison to the scaled performance of MORPHEUS which has the value of 0.02 GOPS/mW at 90 nm CMOS.

3. OFDM Receiver Algorithms

OFDM, as a special case of multicarrier modulation, is introduced and developed to combat with Inter-Channel Interference (ICI), Inter-Symbol Interference (ISI), frequency selective fading and also multipath channel fading in wireless communications. In an OFDM system, a transmitted data stream is divided into several parallel streams over different parallel subcarriers to achieve higher data rates [12]. The OFDM receiver, shown in Figure 2, is composed of several blocks to perform the inverse operation of the transmitter. In the following sections, each block of the receiver as well as the employed algorithms will be explained.

3.1. Time Synchronization

Time Synchronization is the first block of the OFDM receiver. It finds the exact moment of individual OFDM symbols and the starting point of the FFT window. One approach is a Cyclic Prefix (CP) correlation based method. Each of the received data symbols (y_n) should be correlated with its delayed version and thereafter the maximum values have to be found [13]. The following two steps perform a Time Synchronization:

1) Correlation between y_n and conjugation of y_{n-D}. The length of delay (D) is equal to the length of CP (16).
$$c_n = y_n y^*_{n-D} \tag{1}$$
2) Calculating the square modulus of produced outputs from the last step and finding the maximum value which is also the right position of the FFT window.
$$z_n = \sum_{i=0}^{L-1} c_{i+n} \tag{2}$$

3.2. Frequency Offset Estimation

The next block of the OFDM receiver is Frequency Offset Estimation where the amount of Carrier Frequency Offset (CFO) caused by device impairments should be estimated [12]. In other words, the received baseband signal after downconversion will be centered at $f_\triangle$ instead of zero. This can cause ISI and inaccuracy of demapping the data symbols. To estimate the CFO, special training symbols added in the transmitter can be employed. Each received packet contains short and long training symbols in addition to the data symbols which are predefined samples known for the receiver. At the receiver side, the down-conversion of the signal r_n can be expressed as

$$r_n = x_n e^{j2\pi f_{\triangle} n T_s} \tag{3}$$

where x_n is the transmitted signal. By applying the delay-and-correlate method on the output of Equation (3) and its delayed version, CFO can be estimated as depicted in Equation (4) and (5). The amount of delay, *D*, is equal to 16 which is calculated by multiplying the period of short training symbols and frequency spacing (0.8 μs $\times$ 20.0 MHz).

$$z = \sum_{n=0}^{L-1} r_n r_{n+D}^{*} \tag{4}$$

$$\hat{f_{\triangle}} = -\frac{1}{2\pi D T_s} \angle z \tag{5}$$

In Equation (5), T_s is the sampling period and $\angle$ takes the angle of z. Subsequent to finding the CFO, frequency offset correction has to be performed as the last step of this block by multiplying the estimated CFO and the received signal based on the following equation where ${r_n}'$ is the corrected signal, n is the sample index and N is the number of samples in a symbol.

$${r_n}' = r_n \times e^{-j2\pi f_{\triangle} \frac{n}{N}} \tag{6}$$

3.3. Fast Fourier Transform

FFT is the next block of the OFDM receiver which is required for converting the corrected received signal from time domain to frequency domain. FFT is the special case of the Discrete Fourier Transform (DFT) which can be performed efficiently by using radix-2^m structures where $m \in Z^{+}$ [16]. By increasing the value of m, the number of execution stages of the FFT will be decreased. However, the arithmetic resources required by the FFT structural unit as well as its complexity will be increased.

3.4. Channel Estimation

Before the OFDM data symbols arrive at the receiver, they may get distorted during the transmission. Therefore, channel estimation is required for estimating the channel frequency response to resolve correctly the symbols. Frequency domain channel estimation can be performed using pilot subcarriers and interpolation. According to the IEEE 802.11a/g specifications, there are four regular pilot subcarriers which include constant known data. At the receiver side, the received noisy data symbols, Y_n, can be written as Equation (7) where *n*, H_n and N_n stand for the number of subcarriers, channel response and additive noise, respectively.

$$Y_n = X_n H_n + N_n \tag{7}$$

The process of channel estimation is composed of two steps: estimation of H_n and correction of Y_n as close as possible to X_n [14]. The channel response can be computed by multiplying the inverse of the diagonal matrix formed from transmitted pilots, M^{-1}, and the received noise-impaired pilots, P_{Rx} shown in Equation 8.

$$\tilde{H_k} = M^{-1} P_{Rx} \tag{8}$$

Here, k and $\tilde{H_k}$ are representing the number of pilots and the channel response of the received pilots, respectively. As the next step, the channel response of the remaining subcarriers requires to be computed by using Linear Interpolation which is a method for approximating the value at each position among two adjacent known values. It can be mathematically expressed as the following equation:

$$\hat{H_n} = \sum_{i=1}^{N_p-1} \sum_{j=1}^{N_s} \tilde{H_k}(i) + ((\tilde{H_k}(i+1) - \tilde{H_k}(i)) \times \frac{j-1}{N_s}) \tag{9}$$

where $\tilde{H_n}$, N_p, N_s and μ stand for the extension of channel frequency response of the pilots for the remaining subcarriers, number of pilots and samples and the step size, respectively. Subsequent to channel estimation, channel equalization is required to be implemented as an effort to correct the received noisy data symbols. Therefore, the received data symbols are needed to be divided by the estimated channel response according to the following equation.

$$\hat{Y_n} = \frac{Y_n}{\hat{H_n}} \tag{10}$$

3.5. Symbols Demapping

Subsequent to performing all required synchronization operations, the last step is the demodulator. This is required to determine the transmitted data bits for each received data symbol. In general, there are two decision methods for this purpose: hard and soft. In this research work, the first method is used for performing symbols demapping for 16-QAM (Quadrature Amplitude Modulation) constellation points. At the transmitter side of the OFDM, data bits are modulated with one of the schemes such as 16-QAM which comprises four bits per symbol. Moreover, since QAM changes both the amplitude and phase of the carrier, transmitted data symbols are complex and composed of Quadrature and In-phase carriers. Thus, at the receiver side, the complex plane should be divided into decision boundaries which include the set of points. Each point includes Quadrature and In-phase parts which are equivalent to real and imaginary parts of the received data symbols, respectively. As the complex plane is divided into In-phase and Quadrature areas, there are four zones for each leftmost and rightmost two bits. Therefore, the received data symbols can be mapped to data bits according to the decision boundaries and the closest constellation points to them. In other words, a data symbol in each of 16 decision areas will be mapped to the bits assigned to that constellation point [12].

4. Implementation of Baseband Processing on the ZC706 Evaluation Board

In this section, the instantiation and implementation of the OFDM receiver baseband processing on the Xilinx Zynq SoC using the high-level synthesis tool, SDSoC development environment, as an application for a Linux host is presented. SDSoC provides an embedded C/C++ application programming interface for implementing heterogeneous embedded systems. Each block of the OFDM receiver is written in C++ code based on the aforementioned algorithms. The overall platform for the OFDM receiver can be customized with application-specific hardware accelerators which are connected to the platform through a data motion network. The entire architecture of the OFDM receiver on the ZC706 evaluation board is depicted in Figure 3. It has to

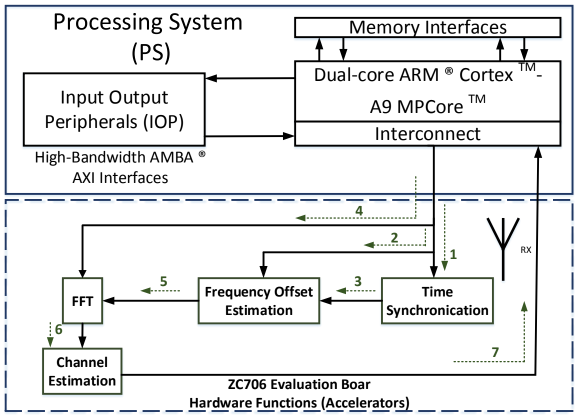
Processing System (PS)
Memory Interfaces
Dual-core ARM ® Cortex TM-A9 MPCore TM
Input Output Peripherals (IOP)
Interconnect
High-Bandwidth AMBA ® AXI Interfaces
4
2
1
RX
5
3
FFT
Frequency Offset Estimation
Time Synchronization
6
7
Channel Estimation
ZC706 Evaluation Boar
Hardware Functions (Accelerators)

Figure 3. General view of IEEE 802.11a/g receiver on ZC706 evaluation board.

be mentioned that the platform for the ZC706 development board is the default provided by the SDSoC tool. The order of executing the hardware accelerators as well as exchanging the results and data symbols among the different blocks is specified with the green colored numbers and dashed arrows. Moreover, the hardware/software connectivity and the timeline for programmed hardware functions to be called during the process of the OFDM receiver are shown in Figure 4. As the first step, the data symbols are loaded from the ARM processor to the FPGA hardware by a DMA (dashed arrow number 1). SDSoC provides different DMA configurations. In this case, for array arguments, `axi_dma_sg` (scatter-gather DMA) data mover is required. In general, the data motion network in the SDSoC environment is composed of the following components: hardware interface for the accelerators, data movers between the Processing System (PS) and the FPGA exchanging data through AXI ports from the ARM processor. As it can be seen from Figure 4, the CPU establishes each hardware function and the data transfer for each function call. Once all calls and transfers are completed and the inputs of hardware functions become available, accelerators can start their actual computation based on the algorithm data flow controlled by the software. It has to be mentioned that setting up Data Mover (DM) for each new array can be performed in parallel with the data transfer of the previous one. As an example, setting up DM for the first input of TS which is an array with the size of 80 will take 1005 Clock Cycles (CC). Afterward, this array can be transmitted to the hardware function in 1317 CC while at the same time, setup DM for the second array can be established. In other words, the total process of transferring data from the CPU to the FPGA hardware functions will take 6642 CC, expressed in Equation (11).

$$6642\,CC = 5334\,CC + (5334\,CC - 4026\,CC) \quad (11)$$

Subsequent to performing the previous stages, the correlation algorithm as well as seeking the index of the time offset can be implemented in 9549 CC. The simplified algorithm written for performing the Time Synchronization block in SDSoC development environment based on Equation (1) and (2) is depicted in Algorithm 1. As it can be observed from the highlighted parts, loop pipelining and loop unrolling can be employed. The `pragma` *pipeline* improves the performance of the hardware function by implementing the loop in a concurrent manner. Loop pipelining transforms the sequential execution of operations into parallel operations. Subsequent to applying pipelining, the cycles to execute all operations are reduced. The efficiency of this `pragma` is limited mainly by data dependency. The `pragma` *loop unrolling* improves also the performance by exploiting the parallelism between loop iterations. Unrolling creates copies of the loop body which are executed concurrently. The programmer can determine how many copies are created by inserting a factor. The available hardware resources and data dependency give an upper limit of how many copies can be created. In this case, HLS unroll factor is chosen to be 4 as the most effective factor by using trial and error. Furthermore, since the design was too large, it was not synthesizable for the larger unroll factors.

Algorithm 1 CP correlation based method for TS

```
1:  Initialize Location to 0, and Max to $-1$
2:  for i:=1 to 80 step 1  do
3:      #pragma HLS PIPELINE enable_flush rewind off
4:      for j:=1 to 80 step 1  do
5:          #pragma HLS PIPELINE II=1
6:          #pragma HLS unroll factor=4
7:          $\mathbf{c} \leftarrow \mathbf{y}(j)\mathbf{y}^*_D(j+i) + \mathbf{c}$
8:      $\mathbf{z} = real(\mathbf{c}) * real(\mathbf{c}) + imag(\mathbf{c}) * imag(\mathbf{c})$
9:      if $(z > Max)$ then
10:         $\mathbf{Max} = \mathbf{SM}$
11:         $\mathbf{TimeOffsetIndex} = \mathbf{i}$
```

As the next step, the short training symbols as well as the index of time offset have to be transferred from the ARM processor to the hardware accelerator for performing frequency offset estimation and correction in 10280 CC (dashed arrows number 2 & 3). The computation of Frequency Offset Estimation block can be executed in 4747 CC. In parallel with the computation of Frequency Offset Estimation, twiddle factors can be loaded from CPU to the FPGA hardware functions which are required for implementing the FFT block. In the next stage, FFT is performed by the use of the radix-4 algorithm on the results of the Frequency Offset Estimation block which are exchanged directly among the accelerators in 6487 CC. Once the FFT is performed completely, the data is transferred to the next block (dashed arrow 6) in order to implement the Channel Estimation which is the last hardware function. Then the results of the Channel Estimation block should be transferred back to the PS in order to convert the data symbols to data bits. As it is mentioned above, the hard decision method is employed for Symbols Demapping which is just based on the comparison among the received data symbols and the constellation points located in the same decision boundaries. This task can be performed completely in software without requiring any accelerator.

5. Experimental Results, Comparison and Discussion

The number of clock cycles required for implementing each block in hardware on the FPGA and in software on the ARM processor in comparison with the HARP platform are shown in Table 1. The speed-up for each accelerator is calculated based on the execution time (since the platforms compared are running at different clock speeds) which is the product of the total number of

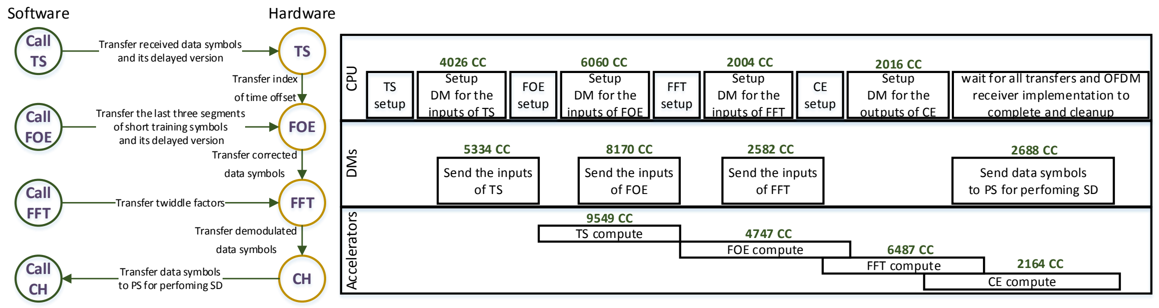
Software
Hardware
Call TS
Transfer received data symbols and its delayed version
TS
Transfer index of time offset
Call FOE
Transfer the last three segments of short training symbols and its delayed version
FOE
Transfer corrected data symbols
Call FFT
Transfer twiddle factors
FFT
Transfer demodulated data symbols
Call CH
Transfer data symbols to PS for perfoming SD
CH
CPU
TS setup
4026 CC
Setup DM for the inputs of TS
FOE setup
6060 CC
Setup DM for the inputs of FOE
FFT setup
2004 CC
Setup DM for the inputs of FFT
CE setup
2016 CC
Setup DM for the outputs of CE
wait for all transfers and OFDM receiver implementation to complete and cleanup
DMs
5334 CC
Send the inputs of TS
8170 CC
Send the inputs of FOE
2582 CC
Send the inputs of FFT
2688 CC
Send data symbols to PS for perfoming SD
Accelerators
9549 CC
TS compute
4747 CC
FOE compute
6487 CC
FFT compute
2164 CC
CE compute

Figure 4. HW/SW connectivity with direct connection and timeline for hardware function calls. DM stands for Data Mover.

TABLE 1. Clock cycles required for processing in two platforms at different stages. In the table, TS, FOE, CE and SD stand for Time Synchronization, Frequency Offset Estimation, Channel Estimation and Symbols Demapping. Moreover, * and † stand for those algorithms which have been executed by SW instead of HW and data transfer from HW to SW, respectively.

Platform	HARP (CC) at 200 MHz			SDSoC (CC) at 667 MHz					Gain	Gain
Accelerator	Data transfer	HW	Execution Time (μs)	SW	Execution Time (μs)	Data transfer	HW	Execution Time (μs)	(SDSoC HW vs HARP)	(SDSoC HW vs SW)
TS	9,619	3,465	17.33	14,082	21.11	6,642	9,549	14.32	1.21X	1.69X
FOE	4,664	12,708	63.54	14,366	21.53	10,280	4,747	7.12	8.92X	3.16X
FFT	2,677	328	1.64	9,282	13.91	3,160	2,503	3.75	0.44X	3.71X
CE	1,435	250	1.25	5,276	7.91	2,016†	2,164	4.74	0.38X	4.86X
SD	-	2,253*	16.27	3,724	5.58	-	-	-	4.03X*	-
OFDM Receiver	-	31,521	157.6	46,730	70.1	-	26,677	40	3.94X	1.75X

clock cycles and the clock period. Accordingly, the overall speed-up is the ratio of the old execution time to the new execution time for a system. The clock frequency of HARP and SDSoC are equal to 200 and 667 MHz, respectively. The calculated speed-up of the SDSoC HW accelerators compared to HARP is given in Table 1. As it can be observed, the implementation of Time Synchronization, Frequency Offset Estimation and Symbols Demapping on the ZC706 evaluation board shows overall speed-ups with the values of 1.21X, 8.92X and 4.03X, respectively. However, in the case of the FFT and channel estimation, HARP has better performance which shows the ability of CGRAs for executing parallel tasks. Although the FPGA is also a worthy platform in executing parallel tasks, however by the use of CGRA, the designers can implement near-optimal solutions for their target applications since the placement and routing can be performed manually. One of the reported problems of CGRAs is a fixed set of PEs and interconnections which are not optimal for various applications in terms of cost and performance. In order to tackle this issue, several Design Space Exploration (DSE) techniques have been proposed which are concerned with the interconnection between PEs [15]. Therefore, the users of CGRAs can design near-optimal applications in terms of cost and performance by employing the DSE techniques. Another difference between CGRA and FPGA is related to the level of granularity of the employed reconfigurable devices. Fine-grained devices can be characterized by PEs with granularity of 2 or 4 bits. However, FPGAs can potentially accept any bit-size while their concept is based on LUTs not PEs. Compared to coarse-grained devices, fine-grained devices require more PEs and also a combination of PEs to perform a single operation which results to large resource demanding. On the other hand, fine-grained devices might be better in terms of flexibility for performing new tasks. In CGRAs, each cell performs 16-bit or 32-bit operations. CGRAs provide high level of data parallelism and throughput because of the symmetry in their structure. However, they occupy an area of a few million gates, which makes them expensive and can have a potentially high transient power dissipation. The algorithms related to TS, FFT and CE are parallel in nature. Based on the amount of computations required by each of them as well as their algorithms, they may have better performance on FPGA or CGRA. Regarding the Symbols Demapping block, it was implemented by SW in both platforms instead of HW due to its faster implementation. Moreover, the estimated speed-up gained by hardware accelerators compared to the software-only measured clock cycles can be observed from the rightmost column of the Table 1. As it was predicted, considerable speed-up can be achieved by using the hardware accelerators. For the whole OFDM receiver, ZC706 evaluation board could give speed-up of 3.94X compared to the HARP platform due to the different clock frequency and NoC used for exchanging the data.

TABLE 2. Synthesis results of the proposed accelerators on ZC706 evaluation board and also resource utilization summary of HARP (Stratix-V (5SGXEA4H1F35C1) FPGA device) against ZC706. Acc stands for Accelerator.

Acc	Resource Utilization					
	ZC706				HARP [5]	
	BRAM	FF	LUT	48-bit DSP	ALMs	18-bit DSP
TS	0	1,019	563	34	22,616	40
FOE	0	41,506	47,838	459	8,171	32
CE	24	29.664	5,143	153	22,809	56
FFT	4	26,900	24,894	103	25,908	66
SD	0	1,875	9,101	0	0	0
Total %	28 (5.14)	100,964 (23.09)	117,539 (53.77)	749 (83.22)	98,729 (62)	230 (90)

Table 2 presents the resource utilization summaries of each of the designed accelerator on Zynq®-7000 all programmable SoC ZC706 evaluation kit. Table 2 also presents the comparison of the resource utilization of HARP and ZC706 evaluation board. It has to be mentioned that a single ALM in Stratix-V FPGA device is composed of two LUTs. The total number of 18-bit DSP resources utilized in the case of HARP 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. In the case of the ZC706 evaluation board, a 48-bit DSP block is an arithmetic logic unit used by hardware functions and composed of an add/subtract unit and a multiplier connected to a final add/subtract/accumulate engine. It means that against HARP platform where DSP was just used for multiplication, the employed DSP blocks in the ZC706 evaluation board can act as an arithmetic logic unit. Regarding the power consumption, it is estimated by using Xilinx Power Estimator (XPE) tool for the whole OFDM receiver on the ZC706 evaluation board. It shows 3.171 W for total on-chip power at an ambient temperature of 25°C which includes 0.243, 2.18 and 0.748 for static, Processor System (PS) + dynamic and I/O power, respectively. In this context, Zynq-7000 all programmable SoC ZC706 evaluation board shows an overall improvement in terms of total power dissipation compared to the HARP platform with the value of 3.9 W (1.22X). Regarding the comparison among HARP platform and Xilinx Zynq SoC, as the clock frequencies are different, largely based also on the different FPGA platforms used, the comparison is not a completely fair. Probably HARP would achieve a higher clock frequency on the ZC706 board, but we do not know as we do not have such an implementation of HARP reported. Even if HARP would run at the same clock frequency as implemented design on ZC706, we would achieve a slight speed-up over HARP. Therefore, a direct comparison is difficult and the results are only indicative.

6. Conclusion

In this work, a novel HW/SW codesign of IEEE 802.11a/g baseband processing using a high-level synthesis tool is presented. The HW/SW codesign is mapped to an ARM processor and an FPGA. The IEEE 802.11a/g receiver has computationally intensive parallel and serial tasks that have a high potential to be accelerated by the FPGA using a high-level synthesis tool. Therefore, important insights into the explored platform can be provided from the implementation results as well as a comparison with another state-of-the-art heterogeneous platform (HARP). It can be concluded that the implementation of inherently parallel algorithms such as FFT is more efficient by employing Coarse-Grained Reconfigurable Architectures (CGRAs) instead of using high-level-synthesis for FPGAs. Moreover, the users of CGRAs can implement near-optimal solutions for their target applications by employing various design space exploration techniques. On the other hand, for the mixture of serial and parallel tasks, Xilinx Zynq SoC platform is more powerful since all parts of an algorithm can be mapped into hardware/software. In contrast to the HARP platform and also software implementation of the receiver on Zynq, the presented approach shows an overall speed-up of 3.49X and 1.75X, respectively.

Acknowledgment

This research work is jointly conducted by the Laboratory of Electronics and Communications Engineering, TUT, Tampere, Finland and the Application-specific Multi-Core Architectures Research Group, RUB, Bochum, Germany. 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.

References

[1] D. Pizzolli et al., "Cloud4IoT: A Heterogeneous, Distributed and Autonomic Cloud Platform for the IoT," 2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Luxembourg City, 2016, pp. 476-479. doi: 10.1109/CloudCom.2016.0082

[2] 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. 1131-1136). NY, USA: ACM.

[3] "Introduction to FPGA Design with Vivado High Level Synthesis", UG989 (v1.0), July 2, 2013. Available at: http://www.xilinx.com

[4] "SDSoC Environment Tutorial", UG1028 (v2016.3) November 30, 2016. Available at: http://www.xilinx.com

[5] S. Nouri, W. Hussain and J. Nurmi, "Evaluation of a Heterogeneous Multicore Architecture by Design and Test of an OFDM Receiver", IEEE Transactions on Parallel and Distributed Systems, 2016, submitted and under review.

[6] D. Melpignano and et. al., "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 and et. al., "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 and et. al., "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 and et. al., "MORPHEUS: Heterogeneous Reconfigurable Computing", International Conference on Field Programmable Logic and Applications, FPL 2007, pp. 409-414, 27-29 Aug. 2007.

[10] W. Hussain and et. al., "HARP2: An X-Scale Reconfigurable Accelerator-Rich Platform for Massively-Parallel Signal Processing Algorithms", in Journal of Signal Processing Systems, Springer, vol. 85, issue 3, pp. 341-353, 2016.

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

[12] 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.

[13] J.-J. van de Beek and et. al., "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.

[14] Man-On Pun and et. al., "Multi-carrier techniques for broadband wireless communications : a signal processing perspective", copyright ©2007 by Imperial College Press, December 2007.

[15] Y. Kim and et. al., "Design Space Exploration for Efficient Resource Utilization in Coarse-Grained Reconfigurable Architecture," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 18, no. 10, pp. 1471-1482, Oct. 2010. doi: 10.1109/TVLSI.2009.2025280

[16] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series", Math. Comp., vol. 19, pp, 297-301, April 1965.