EURASIP Journal on Embedded Systems
a SpringerOpen Journal

RESEARCH Open Access

Embedded reconfigurable synchronization & acquisition ASIP for a multi-standard OFDM receiver

Mahmoud A Said*, Omar A Nasr and Ahmed F Shalash

Abstract

Embedded reconfigurable architectures are currently attracting increasing attention in the wireless communications industry due to the escalating number of wireless standards in today's market. Application specific instruction-set processors (ASIPs) present a reconfigurable solution that offers a compromise between programmability and low power consumption. In this article, the design and implementation of an embedded synchronization and acquisition ASIP for OFDM based systems is proposed. The engine architecture is presented and the programming model is explained in details. The proposed engine is scalable and it can be configured to support a multitude of synchronization algorithms and OFDM standards. While applicable to many OFDM systems, the proposed architecture was successfully verified on long term evolution (LTE Rel. 8) and WiMAX 802.16e systems. A partial list of synchronization and acquisition algorithms are tested on the engine for the two standards, and the results highlight the capabilities of the engine. The processor has been synthesized with 0.18μm standard cell CMOS library. It is estimated to occupy 1.1 mm^2 and the projected power consumption is 7.9mW at 120 MHz, which meets the speed requirements of the tested standards. More results are included within the article.

Keywords: reconfigurable ASIP, embedded processors, baseband processors, low-power design, OFDM synchronization

1 Introduction

Contemporary wireless standards allow for the radio to have connectivity with more than one technology at the same time. For example, the radio can be connected to a Wi-Fi hotspot when a signal exists, or to a WiMAX base station if the Wi-Fi signal is weak or does not exist. The ability to connect to more than one technology increases the reliability and the use of the radio's connectivity. It also enables applications that require constant connectivity, such as remote health care and remote industrial automation, which cannot tolerate any loss of connectivity at any time. Moreover, there are still competition, enhancements, regional variants, and new versions of the wireless standards that emerge with time. For example, in the field of 4G and beyond, the marginal competing standards and the need to have an easy migration path between different systems increases the need for configurability without sacrificing throughput, area or power consumption. This line of thinking gave a boost to the concept of the software defined radio (SDR) [1]. SDR, in general, is based on general purpose digital signal processors (DSPs). Thus, it suffers from limitations in throughput and power consumption. However, the need for programmability and configurability is unabated due to the proliferation of wireless standards. Another approach to achieve configurability without sacrificing power consumption is to use application specific instruction-set processors (ASIP) [2-4]. In ASIP technology, a core unit is programmed using a specific instruction set that configures the core unit to perform multiple functionalities.

For most of today's and emerging standards, orthogonal frequency division multiplexing (OFDM) was the modulation scheme of choice in systems such as high performance LAN type 2 (HIPERLAN/2) [5], IEEE 802.11a [6], IEEE 802.16 family [7] and 3GPP long term evolution (LTE). OFDM's main advantage is its ability

* Correspondence: mahmoudabdelall2005@yahoo.com
Center for Wireless Studies, Faculty of Engineering, Cairo University, Cairo, Egypt

to alleviate the inter-symbol interference (ISI) caused by multi-path fading channels, even for large

channel delay spreads. Hence, at the receiver, there is no need to design complex channel equalizers, which reduces the complexity and the power consumption of the receiver. On the other hand, OFDM systems are very sensitive to synchronization errors [8]. Therefore, there is a need to design, and efficiently implement high accuracy synchronization algorithms using embedded reconfigurable engines that can support the increasing number of OFDM-based standards. The concept of reconfigurable engines for wireless applications has been previously explored in the literature. Configurable radio architectures that can support multiple standards were proposed in [9,10], where the engine core consists of an array of reconfigurable units. Poon [11] uses five different configurable units to perform all tasks for the digital part of the radio. Application specific processor architecture was proposed in [4] for OFDM channel estimation. In our previous study [3], an ASIP architecture is proposed to support synchronization tasks in OFDM systems. In [3], we proposed only the architecture of reconfigurable engine architecture to achieve a compromise between powerful dedicated hardware implementations and very flexible general DSP processors, but with limited programming capabilities.

Expanding on [3], an embedded reconfigurable ASIP-based engine that can efficiently carry out OFDM synchronization and acquisition tasks is presented. The main building block of the engine is a core unit that was designed to efficiently carry out synchronization tasks. The core unit can be programmed with a special instruction set to optimize the usage of the hardware resources. Memories for data and instructions, registers for intermediate data storage, and an instruction decoder are all parts of the the engine. The engine and the instruction set are optimized for vector instructions, which are frequently used in synchronization and acquisition algorithms. The results show that the hardware multiplexing in this ASIP solution reaches a smaller implementation area than the solution of multiple dedicated implementations. In addition, it allows a higher degree of hardware reuse between different algorithms in different standards.

The organization of the article is as follows: Section 2 introduces the OFDM system model. The detailed engine architecture is proposed in Section 3. Section 4 discusses the algorithm selection and analysis of the processing tasks, while the programming model is discussed in Section 5. Results of the proposed engine are presented in Section 6. Section 7 concludes the article.

2 OFDM system model

A typical OFDM receiver is shown in Figure 1 The used transmission model is described in detail in [8]. The resulting time domain signal $s(t)$ is composed of successive symbols where Symbol l is formed from N subcarriers, $a_{l,k}$ (transmitted data), where l denotes the symbol index and k is the subcarrier index. The subcarrier spacing $\Delta = \frac{F_s}{N}$ where F_s is the sampling frequency. The sampling time of the OFDM signal is $T_s = \frac{1}{F_s}$ The N subcarriers are divided between data, pilots and guard bands according to the used OFDM standard. Pilots are reference signals known at the receiver which are used in data-aided estimations for synchronization or channel estimation purposes. Guard bands are used in order to limit the bandwidth of the transmitted signal to be less than $\frac{1}{T_s}$ A guard interval of length N_g samples is added before each OFDM symbol to combat the multi path fading channel effects. The total number of time samples in one OFDM symbol is $N_s = N + N_g$.

The received signal, when the transmitted signal passes through a channel with an impulse response $h(t)$ is

$$r(t) = \sum_i h_i(t)s(t - \tau_i) + n(t) \tag{1}$$

where delays τ_i are channel tap delays and $n(t)$ is the complex-valued additive white Gaussian noise (AWGN). Sampling the signal at time instants nt_s, and removing the guard interval yielding

$$r_{l,n} = r((lN_s + n)t_s) \tag{2}$$

Demodulation of the subcarriers via a Fast Fourier Transform (FFT) yields the received data symbols:

$$X_{l,k} = \sum_{n=0}^{N-1} r_{l,n} e^{-j2\pi nk/N} \tag{3}$$

This is equivalent to

$$X_{l,k} = a_{l,k}.H_{l,k} + n_{l,k} \tag{4}$$

where $H_{l,k}$ is the channel frequency response at subcarrier k in symbol l and $n_{l,k}$ is the additive noise samples at subcarrier k in symbol l.

Three major synchronization problems result in increasing the error rates at OFDM receivers:

- Inaccurate frame beginning detection.
- Carrier frequency offset (CFO) [8].
- Sampling clock frequency offset (SCFO) [8].

Figure 2 summarizes the synchronization processes in a typical OFDM receiver. First, the incoming signal passes through a packet detection block that will, besides detection of packet existence, give a rough estimate of the symbol beginning. Once a signal is detected, exact frame boundary detection and compensation for

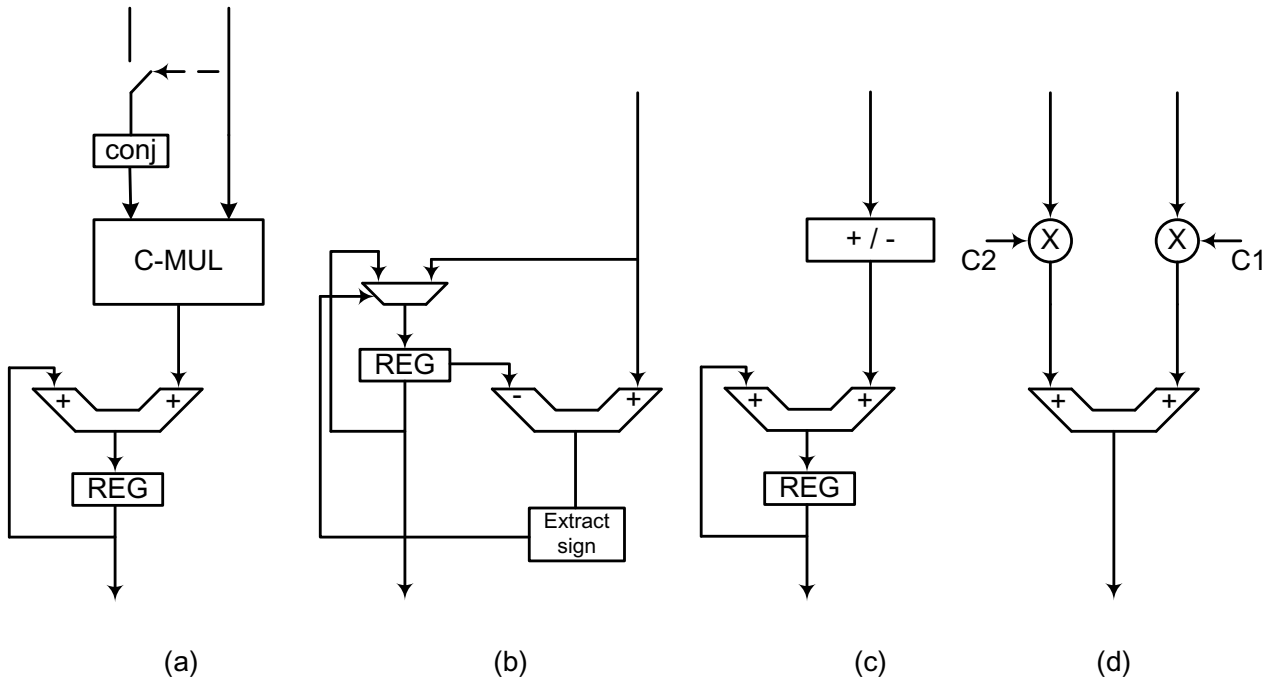

Figure 1 Typical OFDM receiver and processes of our baseband processor.

timing offset are performed. To maintain subcarrier orthogonality, CFO is estimated, refined and corrected after determination of frame boundaries. The signal then passes through the FFT block to estimate the transmitted symbols.

Tracking the variations of the CFO and SCFO is critical in OFDM systems due to their sensitivity to frequency offsets. SCFO and the residual part of the CFO (RCFO) are estimated and corrected in a tracking phase.

The synchronization functions are divided into two main phases:

1. Acquisition phase. Four processes are performed in this phase: symbol timing (frame boundary detection), initial fractional CFO (FCFO) estimation, cell-search (CS) and ICFO estimation. Correction of the estimated errors is shown in Figure 2

2. Tracking phase. In this phase RCFO and SCFO are estimated and corrected.

From the implementation point of view, a significant amount of baseband processing takes place in the synchronization sub-system. Optimized architectures that fulfill the needs of the synchronization sub-system with a high degree of configurability will have the advantage in terms of area and power.

3 Design of the proposed engine

3.1 Engine architecture

The proposed application specific instruction set (ASIP) synchronization engine achieves a compromise between

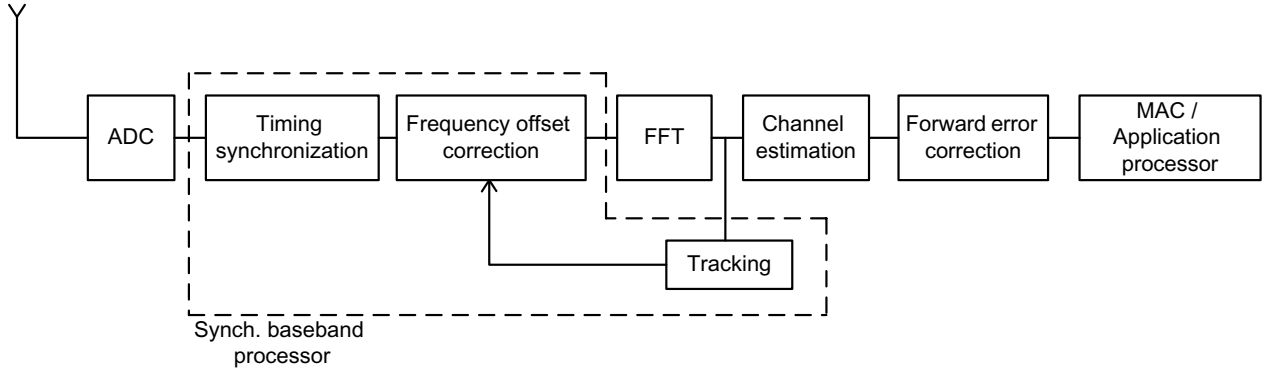

Figure 2 synchronization processes in a typical OFDM receiver.

powerful dedicated hardware implementations and very flexible general DSP processors. All of the used units are grouped together in one pipelined configurable unit (CU). This core unit is optimized for synchronization purposes as well as many other algorithms and allows the execution of many complex operations. We enabled a high degree of hardware reuse, and this resulted in less area and removed many control overheads while a lower degree of parallelism was attained. The choice of a single unit without external accelerators is based on a careful study of synchronization tasks involved within OFDM systems and throughput requirement in the supported standards. Most of the commonly used accelerators like COordinate Rotation DIgital Computer (CORDIC) [12], maximum likelihood (ML) [11] and other accelerators are included with the ordinary complex multiply-accumulate (C-MAC) unit to be executed on CU. This will keep power and area levels as low as possible. The general architecture of the proposed engine is shown in 3 It consists of the main CU, two dual-port memory banks, embedded ROM for any used reference sequences, an output updater (bank of registers) with a simple controller, two register files and the control part (instruction decoder (ID), program memory (PM), specialized control registers).

3.1.1 Configurable unit (CU)

The CU is connected to the output of a CU input generator and controlled directly by the instruction decoder (ID) with a 26-bit control vector, which identifies the working operation and the used elements inside the CU. Figure 4 shows the internal design of CU. Many operations are executed on the same CU. CU configuration for major CU supported operations and the resulting operation are listed in Table 1 Even though the inputs to the CU are complex most of the time, the engine uses real data buses. This design was chosen to remove the limitations imposed by complex data buses on real data operations, especially on phase calculations.

The main operation of the CU is the Complex Multiply ACcumulate (C-MAC). Mathematically, it can be implemented in two ways, using either three or four real multipliers. To limit the number of multipliers, three multipliers are used, although five adders are required as opposed to four adders in the four multipliers scheme. An extra adder is added to allow the summation of eight different real inputs or four complex inputs. In addition to the two large accumulators, the CU uses internal multiplexers to configure the running operation according to the control vector (CV).

The CU consists of six 12-bit real adders, three 13-bit real multipliers followed by two 12-bit rounders, two 24-bit accumulators, two two's complement operations, ten 13-bit multiplexers (MUX) and two 24-bit shifters. The eight ports (I1 ... I8) in Figure 4 are intended for operations on real data while the six ports (I1 ... I6) can be used alone to implement the complex multiplication process. Real ADD/SUB operations are executed with two different precisions (12-bit and 24-bit).

The CU is optimized by pipelining into three pipeline stages. The first stage is an addition stage used for normal and vector complex multiplications. This adds the benefit of having a first stage capable of adding eight

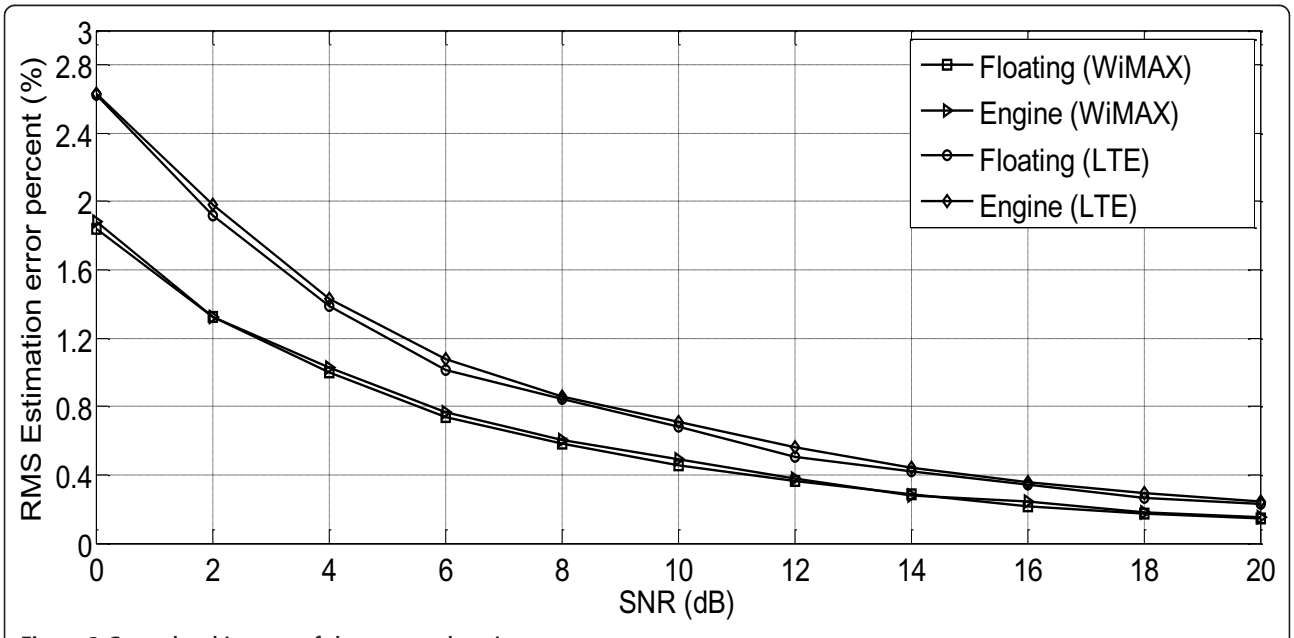

Figure 3 General architecture of the proposed engine.

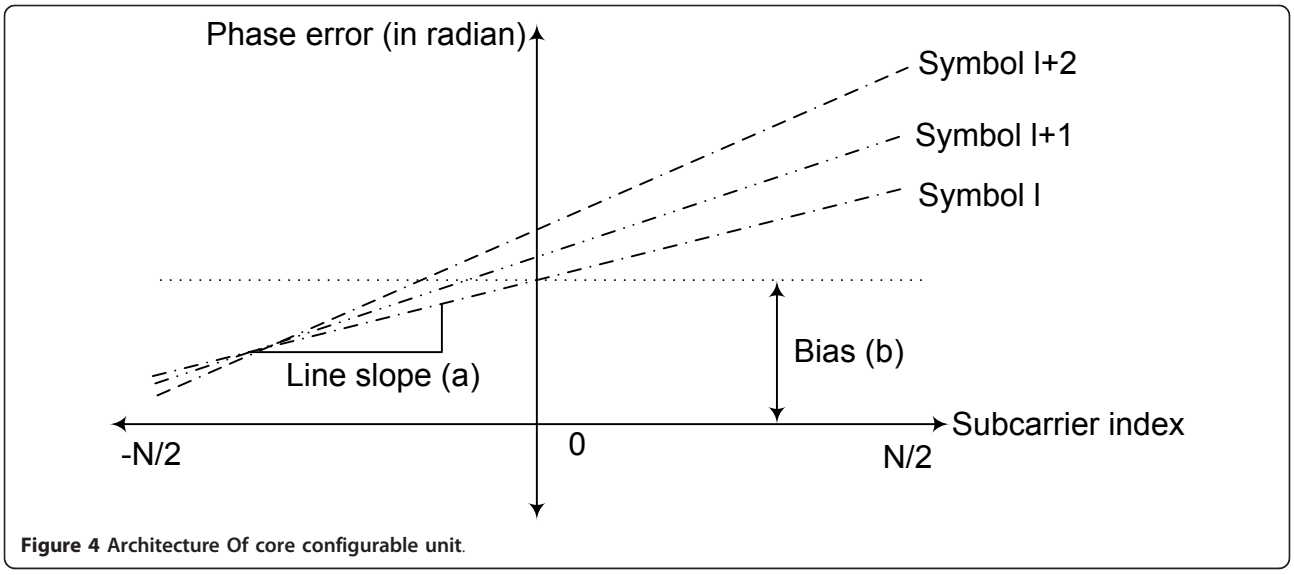

Figure 4 Architecture Of core configurable unit.

real numbers before passing its output to the next addition stage (stage 3). The second stage is the multiplication stage. It has only one multiplier between two registers to minimize the critical path of the overall unit. The third stage is the second addition stage like the first stage but it has only two adders instead of four.

One cycle of latency is achieved when pipelining the CU in normal instructions. Vector instructions are executed on a time multiplexing manner on the CU with a maximum vector length of 256 elements. Among different supported operations, the controlled accumulation (CACC) operation needs the largest number of simultaneous complex input signals. CACC operation adds or subtracts four complex words every cycle. This puts a constraint on the memory system to supply the unit with a maximum of four words every cycle. However in this mode, no write operations can be executed. To work in CACC mode, hardware configuration of the CU with respect to the control vector is done. M1, M2, and M3 multipliers of Figure 4 are bypassed while a running configuration of the Add/Sub operations (A1 ... A8) are controlled via a stored control sequence.

Table 1 Major CU supported operation

CU-configuration	Resulting operation
C-MAC	Auto-correlation
	Cross-correlation
	Euclidean distance calculation
	Vector complex multiplication
Real multiply-add	ab + cd
Controlled C-ACC	BPSK preamble correlation
CORDIC	Vectoring mode
	Rotation mode
Maximum likelihood	On-line comparison

Time multiplexing of operations running on the engine core limits its multi-process/cycle capabilities. A maximum of one operation/cycle can be executed on the engine core, no matter wither this operation is simple like addition or computationally complex like complex-multiplications. Although the architecture has one CU, the engine is scalable via adding multiple CU units connected with each other by the two ports I9 and *I*10 to support larger systems.

3.1.2 Memory system

Memory is divided into two 286 word dual-port banks (24-bit). Memory size is dominated by the maximum supported correlation length of 256 in addition to the free space needed to store any internal outputs. The choice of the maximum correlation length was based on the required performance in 802.16e and 3GPP-LTE release 8. Inputs to the memory system is connected to a Memory Input Generator in Figure 3, which is controlled by the instruction decoder. Memory could accept inputs from the external ports, Register File 2, main CU output or the memory itself in a MOV operation. Memory controller handles the write operations and prevent any racing conditions. The two banks are running on the same operating frequency of the core unit. No special addressing modes are required, and hence, address generators are basically counters. Time sharing between different tasks running on the processor allowed further optimization in the memory system by increasing the memory reuse option.

Two general purpose register files, Register File 1 and Register File 2, in Figure 3 are used with a register input generator controlled by the instruction decoder directly. Register File 1 is of size 12-bit and holds 8 general purpose registers to facilitate data flow operations, counting, set outputs and many other useful operations.

Register File 2 is of size 24-bit but it consists of four general purpose registers only. The first advantage of them comes when dealing with movements of complex data inside the engine. These optimization methods beyond the traditional one fixed size register file allows faster execution of real and complex data operations. For example, moving a complex word from the memory system as two (real, imaginary) parts would take double latency beside the complexity in dealing with the two parts as a one word in the executed program.

Reference correlation sequences are stored in 3072 byte ROM for the both of 802.16e and LTE release 8. The ROM takes its address directly from the instruction decoder with an internal counter for its address only. All data can be transferred between different parts of the engine through a data bus of four complex words maximum.

3.1.3 Input/output interface

The engine interfaces with the outer world through a set of input control signals (*IC*) and output control signals (*OC*) beside two external ports for data transfers. Four *IC* signals are connected directly with the instruction decoder and used for acknowledgment about a certain event. Another four *OC* signals output from the controller of the output updater to identify the state of the engine at any stage. The two external data transfer ports are 24-bit wide each (12-bit real, 12-bit imaginary). One port is connected to the time domain side, while the other port is connected after the FFT operation. Any read operation is carried out through one of these two ports and with the two addresses external read address 1 and external read address 2. The accuracy of the chosen number of bits is verified in Section 6.

The output updater holds the same output value on the same port, until a control signal comes from the instruction decoder to its controller to update the output with a newer value in a certain register. The output registers are general registers used to set any value as an output. Here, we give output registers restricted names to clarify the engine operation. The five output registers holds the starting address, FCFO, CELL-ID, ICFO, and SCFO. All of the five registers are 12-bit each. Typically, Starting Address goes to the input buffer that holds the FFT window to identify the first sample in the incoming frame. The estimated FCFO is used by the CFO correction complex-multiplier to derotate the input samples. CELL-ID is transferred to higher layers. ICFO is added to the fractional part of the CFO to guarantee correct reception with time. Estimated value of SCFO is considered as the seed for the ROB/STUFF correction algorithm in [13].

While transferring data from any port to the internal memory banks, no execution of any other instruction is carried out. This control mechanism is achieved when the controller holds the instruction inside the instruction register by re-entering the same instruction to the instruction register till the end of the transfer process. The same mechanism works for vector instructions, where the controller re-enter the vector instruction to the instruction register till the end of the execution phase.

3.1.4 CORDIC algorithm

The CORDIC algorithm [12] can carry out many trigonometric operations and is used here only for angle measurements. The CORDIC stage is composed of adders and shifters, as shown in Figure 5 The precision of the output depends on the number of stages used. Each additional stage adds one bit of precision. There are three methods of implementation to the CORDIC algorithm [12]; spatial multiplexing, time multiplexing and joint spatial-temporal multiplexing. The time multiplexing method perfectly meets our engine design with the addition of the two shifters in Figure 4 to implement the CORDIC stage. The associated memory required with it is called CORDIC look up table (LUT) of size 14 byte. Another control register is required to control the sign of ADD/SUB module used in the next CORDIC stage. The output angle here has a precision of 12 bits and executes in 18 cycles.

3.2 Engine programming

Programming of the proposed embedded ASIP includes three types of instructions. The first type is the ordinary classes like program flow instructions (conditional and unconditional jumps), move instructions, real, and complex ADD/SUB instructions, interfacing control instructions (external reads, output set). The second type is optimized instructions to facilitate the implementation of synchronization subsystem tasks as well as other algorithms in different parts of the engine. The third type is vector instructions.

Ordinary instructions operate on single data points stored in registers (RF1 & RF2), and the result is automatically stored in another register. Most of instructions of this type take one cycle to complete. All control instructions belong to this simple class of instructions. Optimized instructions are special instructions for special purposes like the *ANGLE* instruction and the *BPCACC* (discussed later). This kind of instructions operates on single point or vector of complex numbers stored either in the memory like *BPCACC* or in registers like *ANGLE*, and the result stored also in either memory or registers. Execution of these instructions always consist of multiple execution stages. Vector instructions operate on vectors of complex numbers stored in memories. The output is either stored in another memory if there is no accumulation associated with it, or in a register from register file 2 if there is an

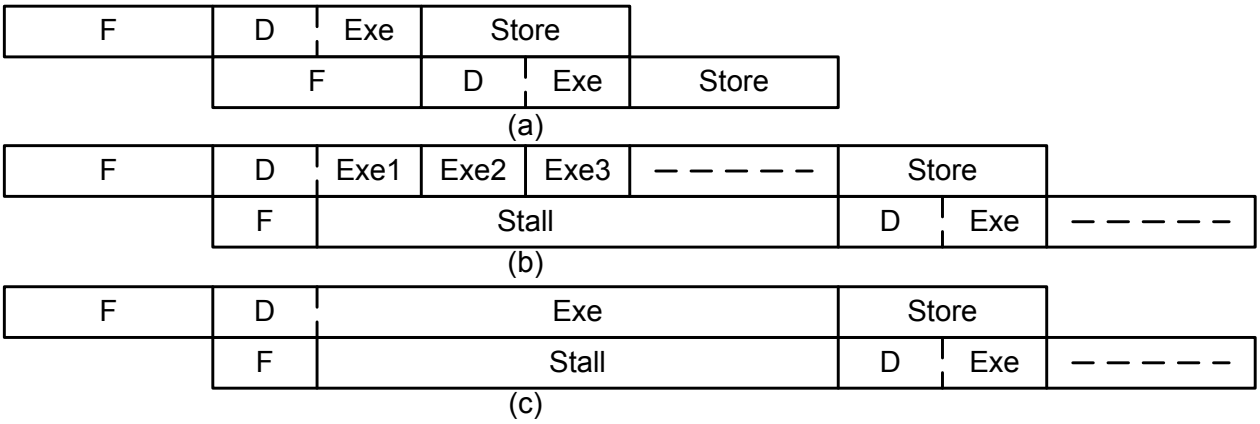

Figure 5 Architecture of a single CORDIC stage.

accumulation. The number of cycles needed for vector instructions depend on the vector length.

In normal operation, one instruction is fetched while another one is decoded and executed as shown in Figure 6a In the execution of special type instructions like the *ANGLE* instruction that measures the angle of a complex number, the pipeline is stalled till the end of the CORDIC subroutine. This operation is shown in Figure 6b When a vector instruction is fetched, a program flow control mechanism is activated and the pipeline initiates a counter with a control value and enter a stall state till the end of execution before fetching the next instruction as shown in Figure 6c

To illustrate the difference in execution of various types of instructions, Table 2 shows an example of the units used and units bypassed in the execution of an instruction of each type. Each instruction is 20-bit wide and the whole size of the required program memory is 512 instructions.

The CORDIC subroutine is executed when fetching the *ANGLE* instruction (special type) in 18 cycles for a precision of 12-bits (as stated before). Shifters included with the engine core are not general purpose shifters that accept arbitrary inputs; they are used only in the execution of the iterative CORDIC algorithm and are controlled by a counter attached with the instruction decoder.

Although most of the instructions are either control instructions or instructions that operate on single data (ADD, SUB,...), the processor operates most of the time on vector data. Hence, the processor is optimized for operations on vectors of complex data or specialized operations associated with many supported tasks. The assembly program becomes relatively long for control or single data instructions compared to what it fulfills.

The engine is programmed via a script (contain the entire program) enters a primitive compiler. The compiler outputs a (.hex and.mif) memory initialization file for the program memory. The output file containing the

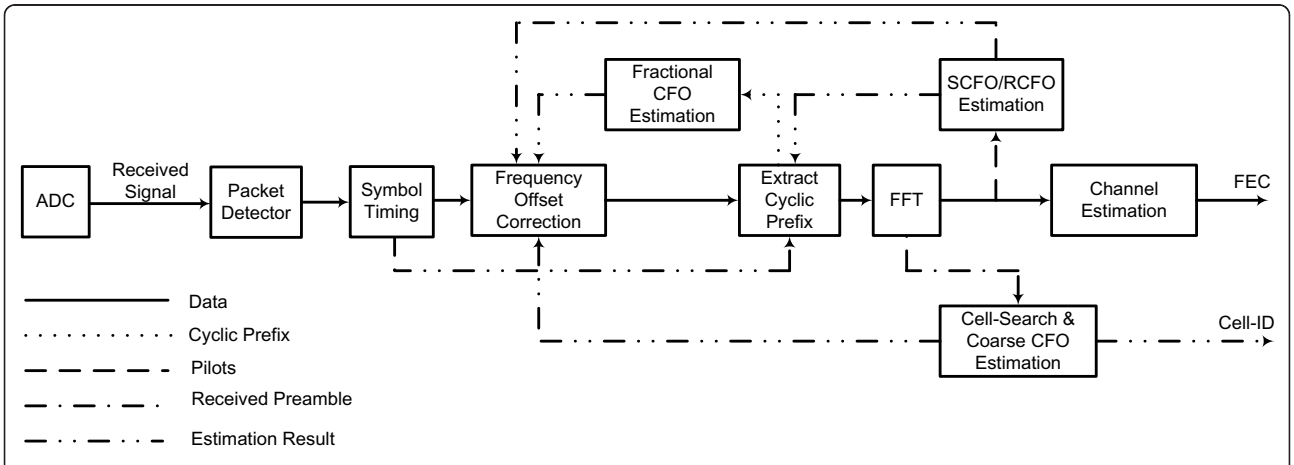

Figure 6 Instructions Pipeline in case of (a) normal executions, (b) special executions and, (c) vector executions.

Table 2 Execution of three operations of different types

Operation	Inputs source	DM1, DM2	M1, M2, M3	A1,..., A6	A7, A8	CORDIC control	CORDIC LUT	shifters	ROMS	output updater
CMAC	Data path (DM)	active	active	active	active	off	inactive	inactive	inactive	inactive
Angle	Register file 2	inactive	bypassed	bypassed	active	on	active	active	inactive	inactive
Read	Data path (External)	active	inactive	inactive	inactive	off	inactive	inactive	inactive	inactive

binary vector is downloaded into the program memory to begin the program fetching.

4 Example algorithms analysis

In this section, we explain a set of typical synchronization tasks usually performed in OFDM receiver. Typically, the receiver perform the following tasks:

1. Obtain coarse frame boundaries with a packet detector.

2. Initiate a search over the samples selected from step (1) to obtain fine frame boundaries. This step with step (1) are called Frame boundary determination (FBD).

3. With the first sample in the frame known, estimate and correct fractional CFO.

4. After FFT, estimate the ICFO and update the value of the frequency of offset estimated in step (3).

5. For OFDM cellular standards, estimate the CELL-ID from the given reference sequence.

6. Track and update the residual CFO (RCFO) and SCFO with the non-preamble OFDM symbols.

For testing and evaluation purposes of the proposed architecture, high level Matlab floating/fixed point models of IEEE802.16e and LTE Rel. 8 have been created to apply the chosen algorithms in [8,13-15]. The proposed embedded ASIP can be configured to implement the chosen synchronization algorithms. It is capable of supporting not only the chosen algorithms but it can support other algorithms as well.

4.1 Packet detection

Detection of the correct boundaries of the incoming OFDM packets has a major impact on the performance of all post FFT sub-systems. Therefore, good timing synchronization algorithms in the acquisition stage will allow early locking on the incoming signal. Using correlation-based algorithms as a detection method directly as in [16] and [17] will cost more energy in the ideal state (no transmitted signals). On the other hand, using a simple, but not accurate detection method, like single or double sliding window (DSW) algorithms [18] to detect the packet and then refine the estimate using correlations will cost us lower energy.

In DSW algorithm, a decision variable is measured and compared with a chosen threshold which depends on the target probability of miss detection and the probability of false alarm. Assume that the decision variable is m_n and the chosen threshold is th. A packet is detected if $m_n > th$ at any sample instant n

$$a_n = \sum_{m=0}^{M-1} |r_{n-m}|^2 = a_{n-1} + |r_n|^2 - |r_{n-M}|^2 \tag{5}$$

$$b_n = \sum_{l=0}^{L-1} |r_{n+1}|^2 = b_{n-1} + |r_{n+L}|^2 - |r_n|^2 \tag{6}$$

$$m_n = \frac{a_n}{b_n} \tag{7}$$

where a_n, b_n, and M, L are the energies and sizes of window A and window B, respectively.

The energy contained in any sample $|r_n|^2$ can be measured with a complex multiplication unit in Figure 7a with a conjugate flag at the second port. The comparison process is executed with a simple subtraction flag check. The size of each window is chosen to be 64 samples, which gives a good performance at low signal to noise ratio (SNR). From the implementation point of view, a total of 384 bytes of memory are required here with ADC resolution of 12-bit (12-bit real, 12-bit imaginary); accuracy is verified by the engine results in Section 6. The packet detector should re-evaluate the decision parameter m_n every new sample, so a decimation in the incoming signal is carried out to leave room for the execution of the required computations.

4.2 Symbol timing

In [14], a maximum likelihood (ML) symbol timing estimator based on cyclic prefix (CP) correlation was proposed. The estimated timing of the first sample, $\hat{t}$, is

$$\hat{t} = \arg\max_n \left[2 \left| \sum_{m=0}^{L-1} r_{n+m} * r^*_{n+m+N} \right| - \rho \sum_{k=0}^{L-1} |r_{n+m}|^2 + |r_{n+m+N}|^2 \right] \tag{8}$$

where ρ is the correlation coefficient between r_k and r_{k+N}

$$\rho = \left| \frac{E(r_n * r^*_{n+N})}{\sqrt{E(|r_n|^2)\, E(|r_{n+N}|^2)}} \right| = \frac{{\sigma_s}^2}{{\sigma_s}^2 + {\sigma_n}^2} = \frac{SNR}{SNR + 1} \tag{9}$$

```
MOV     Vector_Length,236;
VCMAC   DM1,R1,DM2,R2,LR1,LR2;
VEDAC   DM1,R1,LR2;
SUBL    LR1,LR2,LR1;
VEDAC   DM2,R1,LR2;
SUBL    LR1,LR2,LR1;
MOV     Vector_Length,1;
VCMAC   DM1,R1,DM2,R1,LR2,LR3;
SUBL    LR1,LR2,LR3;
VCMAC   DM1,R8,DM2,R8,LR2,LR3;
ADDL    LR3,LR2,LR3;
VEDAC   DM1,R1,LR2;
SUBL    LR3,LR2,LR3;
VEDAC   DM2,R1,LR2;
SUBL    LR3,LR2,LR3;
VEDAC   DM1,R8,LR2;
ADDL    LR3,LR2,LR3;
VEDAC   DM2,R8,LR2;
ADDL    LR3,LR2,LR3;
ADDI    R1,1,R1;
ADDI    R8,1,R8;
ADDI    R3,1,R3;
CMPL    LR3,LR1;
CJMP    011,36;
MOV     LR1,LR3;
ADDI    R7,1,R7;
CMPI    R3,21;
CJMP    '011',17;
SET     BEGINNING_ADDR,R7,R1;
SET     out_ctrl_1,1,R1;
```

Figure 7 Architecture of: (a) standard C-MAC unit, (b) ML unit, (c) controlled accumulation unit and, (d) standard multiply-add unit.

The boundaries of the search window come from the packet detector. To prevent inter-symbol interference, a reasonable shift inside the cyclic prefix is done. Hence, if the detector gives an estimate for the first sample in the symbol at m, the first correlation window begins at m-s, where s denotes the safety shift back. The correlation window slides over time till a search size of $2s + 1$ are evaluated. This implies that the maximum absolute value between $2s + 1$ output correlation result corresponds to the maximum likelihood starting sample. This algorithm has proven its robustness against multi-path fading channels, besides the advantage of being unaffected by the received power level. Only the number of samples L contributing in the cyclic prefix (CP) correlation is affecting the performance of the estimator.

The proposed embedded engine is capable of supporting a maximum correlation length of 256 samples. This maximum is chosen with respect to the required accuracy. This maximum is justified by the comparison between floating point results and the proposed engine results in Section 6. The maximum length of the contributing samples is 256 and can be scaled easily with respect to the required performance. The most complex operation here is the complex multiplication in Equation (8). Auto-correlation and Euclidean distance (ED) calculation (energy) in Equation (8) can be realized by the standard complex multiply-accumulate (C-MAC) unit shown in Figure 7a Subtraction of the output of ED from the output of CP auto-correlation is performed on-line by controlling the accumulation sign. Adders in accumulators are two's complement Add/Sub modules controlled by the processor control unit. The last step is the maximum absolute search between the $2s + 1$ evaluated result with the ML unit shown in Figure 7b

4.3 Fractional CFO estimation

After determination of the FFT window boundaries, the CP is removed according to the indices given by FBD. Fast acquisition of the FCFO requires a pre-FFT algorithm that works without the need of training symbols. With the existence of a frequency offset (Δf) in the received signal $r_{l,n}$, it will take the form in equation (10)

$$r'_{l,n} = r_{l,n} e^{j2\pi \Delta f n t_s} \tag{10}$$

The task now is to derotate (multiply by exponential) the received symbols with the term $e^{-j2\pi \Delta f n t_s}$ to establish accurate subcarrier orthogonality quickly. Multi-stage synchronization strategy is used to achieve both fast and accurate acquisition. In particular, two acquisition stages (pre-FFT for FCFO and post-FFT for ICFO) are used in our system, and then come the tracking of any possible variations. The authors in [8] have proposed a non-data aided estimation algorithm of the FCFO based on the correlation result of the removed CP in Equation (11).

$$\Delta f = \frac{1}{2\pi N t_s} * \arg\left(\sum_{n=\theta}^{L-1} r_{T+n} r^*_{T+n+N}\right) \tag{11}$$

where T is the estimated starting sample index from FBD, θ is the starting point of the correlation window and L is the cyclic prefix length. The reason for not starting the correlation window from the beginning of the cyclic prefix is the multi-path fading channel delay spread τ effect on the estimation performance. The value of θ is chosen such that $\theta > \tau$, so that the channel

effect is the same in the two parts of the correlation and the output is affected only by the added white noise.

Equation (11) can simply be executed on the same complex multiply-accumulate (CMAC) unit used for FBD in Figure 7a Memory requirements here depend on the selected correlation window length with a maximum of 256 samples as stated before. The only difference is the calculation of the correlation angle before multiplying it by a constant. Angle estimation is carried out using the iterative CORDIC algorithm [12]. More details about the implementation of this algorithm were described in Section 3.

4.4 Joint ICFO & CS estimation

In the literature, estimation of the ICFO usually depends on the reference preamble symbol like in 802.16e (WiMAX) or a special synchronization symbols like in LTE. These reference symbols carry also the CELL-ID information. Joint CELL-ID detection and ICFO estimation algorithms are proposed in [19,20]. The task of finding the CELL-ID is named Cell Search (CS). In standards like 802.16e and LTE release 8, reference signals that are used to carry such information are binary random sequences. This feature can make the implementation of this block easier. Assume that the received preamble is $Q(k)$, where k is the subcarrier index. $H(k)$ is the channel impulse response at subcarrier number k. Autocorrelation of the received reference symbol is evaluated as follows:

$$\begin{aligned}\Re\{Q(k)Q^*(k-1)\} &= \Re\{H(k)P_j(k+I)H^*(k-1)P_j^*(k+I-1)\}\\ &\approx |H(k)|^2 D_j(k+I)\end{aligned} \tag{12}$$

where

- $Q(k-1)$: is the first non-zero subcarrier before k.
- $P_j(k)$: is the stored reference sequence of index (j).
- $D_j(k) = P_j(k).P_j^*(k-1)$: is the autocorrelation result of the stored sequences P_j.
- I : is the integral frequency offset normalized to the subcarrier spacing.

For correlation purposes, shifted versions of P_j must also be stored. For example, if we have a maximum ICFO of I_m, $[-I_m, I_m]$, we must store $2I_m + 1$ version from each correlation sequence P_j.

The autocorrelation in Equation (5.4) is used to mitigate the effect of the multi-path fading channel $H(k)$ by multiplying each active subcarrier by the conjugate of its predecessor assuming the channel added phase is nearly equal on both of them. A correlation of the reference patterns shifted by the expected values of the ICFO is evaluated as follows:

$$M_i^{I,j} = \sum_{k=0}^{N_p-1} D_j(k+I)\Re\{Q(k)Q^*(k-1)\} \tag{13}$$

Where $\Re\{Q(k)\,Q^*(k-1)\}$ is called the differential signal and N_p is the number of reference subcarriers contributing to the cross-correlation between the received reference sequence and the stored sequences. The estimated ICFO and CELL-ID is given by:

$$\left(\hat{I},\hat{j}\right) = \arg\max_{I,j} M_I^{I,J} \tag{14}$$

The complexity of this method is acceptable and can be implemented easily on the proposed embedded ASIP, noting that $D_j(k+I)$ in Equation (13) does not have to be calculated on the fly, but can be calculated in advance and stored in the receiver memory. The calculation of $\Re\{Q(k).\,Q^*(k-1)\}$ for $k = 0, 1, \ldots, N_p - 1$ in Equation (13) requires an N_p complex multiplications. The binary nature of $D_j(k+i)$ in Equation (13) makes the remaining computations needed to obtain $M_I^{I,j}$ Binary correlations can be performed using the controlled accumulation (C-ACC) unit shown in Figure 7c Accumulation sign is controlled via stored (shifted and normalized) reference sequence. The constraint on the defined maximum possible shift I_m comes from the symbol duration and the available cycle budget. For example, in IEEE 802.16e, for a maximum ICFO of nine subcarriers (in the range [-9,9]) we need to evaluate 722 [20] different correlation outputs to choose the maximum absolute value as the correct estimate. Comparison between the evaluated correlation results is done on the fly after every new correlation output. Further optimizations of this processing type is done in the design of processor computational core. The engine is optimized not only for this algorithm, but for many other algorithms as well.

4.5 Joint RCFO & SCFO estimation

All previous tasks belong to the acquisition phase. After the acquisition phase, subcarrier orthogonality is loosely established. However, OFDM based systems are very sensitive to the variations of frequency offset. Hence, tracking of these variations with time is important to maintain the resulting signal to error ratio (SER). Another important issue is the SCFO between the transmitter and receiver. A joint data-aided estimation algorithms for RCFO and SCFO are proposed in [15,13], where reference pilot subcarriers are used. In [15], the joint effect of a RCFO δf_r with the existence of a SCFO ζ on the received subcarrier phases after the FFT are shown in Figure 8 This effect is translated in Equation (15)

$$X'_{l,k} = e^{-j2\pi\delta f_r\left(l(N+N_g)-\frac{N}{2}\right)t_s} e^{-j2\pi k\left(l\frac{N+N_g}{N}-0.5\right)\varsigma} X_{l,k} \tag{15}$$

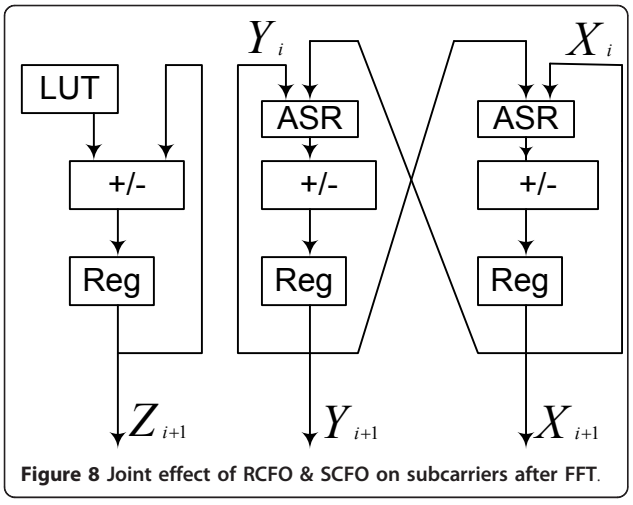

Figure 8 Joint effect of RCFO & SCFO on subcarriers after FFT.

To obtain Equation (15), some terms were neglected. In reality these terms are not neglected and will cause ICI that is measured and stated in the engine results in Section 6.

In general, the symbol number l will have a phase error line with bias $-2\pi\delta f_r(l(N + N_g))$ and slope $-2\pi\varsigma\left(\frac{l(N+N_g)}{N} - 0.5\right)$. In [13], pilot subcarriers are used to form a phase error line that has a bias b and slope a. Let the differential angle at pilot subcarrier index k in symbol number l is $\varphi_{k,l}$, and x_k is the pilot index. Estimation of the phase error line bias b and slope a is carried out as follows:

$$\begin{aligned} a\sum x_k^2 + b\sum x_k &= \sum x_k\phi_{k,l} \\ a\sum x_k + bN_p i &= \sum \phi_{k,l} \end{aligned} \tag{16}$$

The differential angle $\varphi_{k,l}$ is evaluated by multiplying the pilot k at symbol l with the conjugate of the similar pilot at the same index k at symbol l - 1 (the similar pilot could be in an earlier symbol) to mitigate the channel effect. The SCFO ζ is evaluated from the estimated line slope a and the RCFO δf_r could be found from the bias b.

$$\begin{aligned} \varsigma &= \frac{aN}{2\pi(N+N_g)} \\ \delta f_r &= \frac{b}{2\pi t_s(N+N_g)} \end{aligned} \tag{17}$$

The complexity of this algorithm on the proposed embedded engine is dominated by the vector complex conjugate multiplication of the received pilot pattern every symbol and the measurement of pilots angles in Equation (16) using the CORDIC Algorithm [12]. These measured angles are accumulated in two manners: normal accumulation, real multiply accumulate in Figure 7d The memory used to same successive pilot patterns plus the resulting vectors after the conjugate multiplication is relatively large and considered well in the design of the memory system.

5 Algorithm programming on the engine

5.1 Packet detection

The engine was programmed to run the packet detection algorithm, where a new sample read operation is issued every t_s (one sample duration). In order to implement the packet detection algorithm with the time requirements of 802.16e and LTE release 8, a down sampling of the received signal by a factor of 5 is required. Simulation results showed that there is no significant performance loss due to the down sampling needed for the execution of this task on the engine core. The whole packet detection program executes in 31 cycles and repeated with the next new sample. Once a packet is detected a detection control signal will rise to begin the symbol timing procedure.

5.2 Symbol timing

The symbol timing algorithm begins with a read operation of the two parts of the cyclic prefix from External Port 1. In the read operation, we identify the destination memory bank and the length of the read vector (Ex: *READ P*1 : *DM*1,276). The core unit is configured to run the auto-correlation operation through a CMAC instruction with a conjugate flag set on the second input. The output is stored in a register from Register File 2 till the energy (Euclidean distance *EDACC*) contained in the first and the second part of the cyclic prefix are evaluated. A 24-bit subtraction operation (*LADD*) between the stored result and the resulting energy will give the first correlation output. Till now this is considered the only and the largest result, so its index is stored in a register from Register File 1.

Beginning the correlation in Equation 8 from scratch every time is not practical and consumes more energy. The next correlation output can be extracted directly from the estimated correlation output by setting the memory address step by 1 and repeat what is done for the evaluation of the first correlation output. The correlation length this time is not the whole cyclic prefix length but only a single data point. This iterative approach reduces the required execution energy as well as the program length. Once we get a new correlation result, comparison between the stored largest correlation result and the new result is executed. If the new result is larger, an update is done for both the stored value and the index of the largest. A time locking control signal is flagged from the engine when the whole $2 * s + 1$ results are evaluated. The stored index, which corresponds to the largest correlation result, is the correct start

location. The output Starting Address is updated via the output updater by a *SET* instruction with the stored value in Register File 1.

To clarify how the assembly code looks like, a part of the used programming code for symbol timing is shown in Figure 9

5.3 Fractional carrier frequency offset estimation

At this stage, the cyclic prefix is still stored in DM1 and DM2. So, no read operation is issued and the correlation begins directly with the known estimated index from FBD. The correlation output then passes by the CORDIC algorithm using the *ANGLE* instruction to estimate the output phase. The output of the *ANGLE* instruction is stored in a register from Register File 1. According to Equation (11), FCFO is estimated from the output phase by a constant multiplication by $\frac{1}{2\pi N t_s}$. The update on the output FCFO register is carried out via the output updater with the same *SET* instruction.

5.4 Cell-search & integral carrier frequency offset

With the existence of an ICFO, the number of correlations needed to identify the transmitter (CELL-ID) in modern cellular networks can be very large due to the large number of reference sequences associated with each standard (114 for 802.16e, 504 for LTE release 8). A special instruction, called *BPCACC*, is used for the evaluation of a correlation with a binary sequence. The *BPCACC* instruction is capable of evaluating a binary correlation of length N_c in $\frac{N_c}{4} + 1$ clock cycles. The core unit should have 4 new complex numbers every cycle in the execution of the *BPCACC* instruction.

The symbol number of the received reference symbol is known at the receiver. Separation of this symbol is

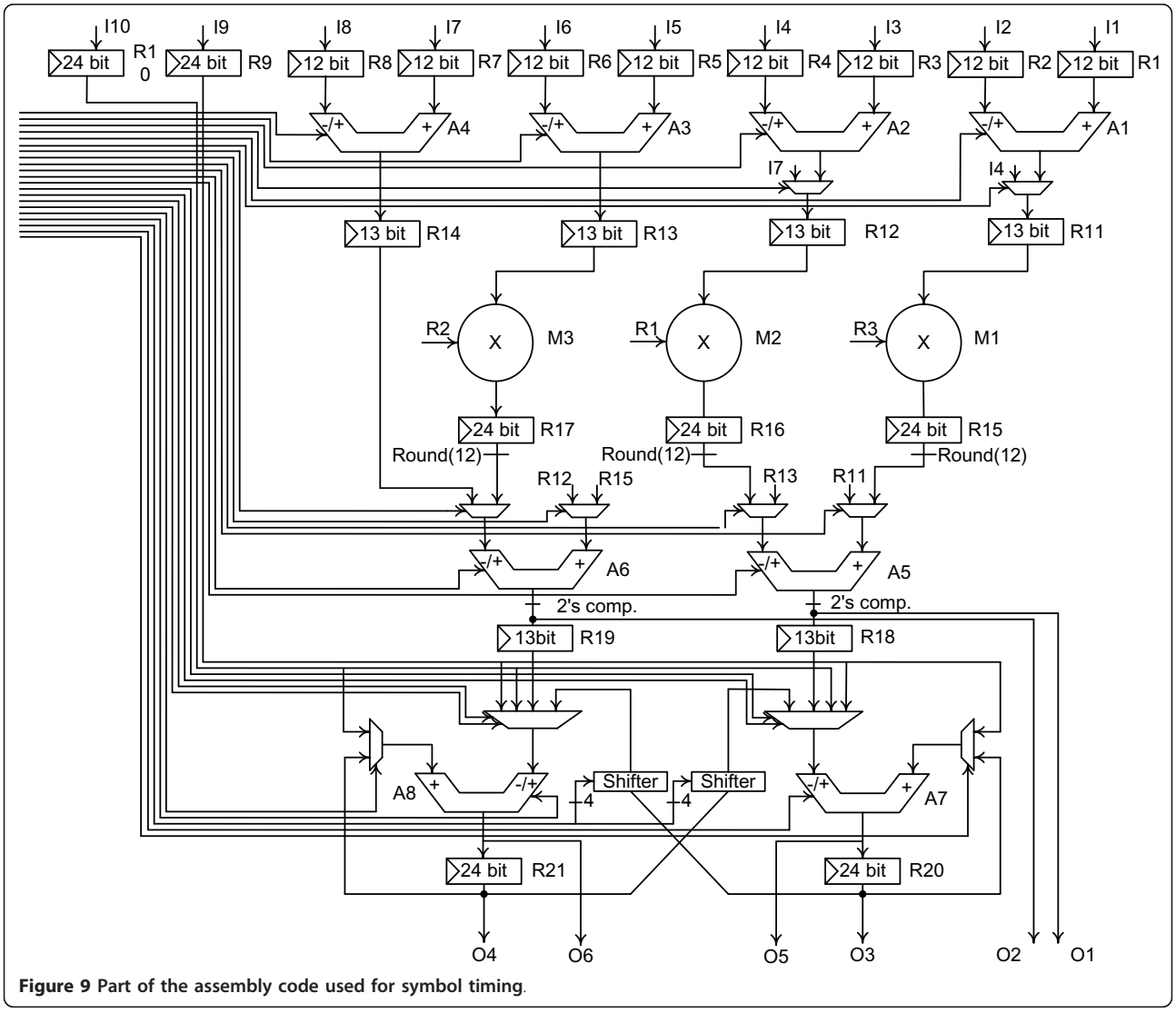

Figure 9 Part of the assembly code used for symbol timing.

done after the FFT, as well as the removal of the guard bands. A read operation from External Port 2 is issued to store the received reference symbol in DM1. To read four successive complex samples from the received reference symbol, a copy of the received reference symbol is stored in DM2. Memory step registers are set to four, so that each port from the four ports of the two memory banks will provide the engine by a different complex data sample every cycle. This allows the memory system to output four consecutive complex words each cycle. We use DM1 to get the differential signal in, which is of length N_c, and store it in DM2. Then, a copy of the contents of DM2 is moved again to DM1. The core unit is configured to perform the binary correlation by adding four complex numbers together with the ADD/SUB signals controlled by the correlation sequence. Every combination of 4-bits from the correlation sequence correspond to a combination of 8 ADD/SUB signals to control the operation of the adders $A1$ to $A8$.

Every new correlation result is compared with the maximum previous result and the index of the maximum correlation output is stored in a register from Register File 1. This index corresponds to the correct ICFO and the attached CELL-ID. The final step in the aquistion phase is updating the values of ICFO and CELL-ID output registers.

5.5 Joint RCFO & SCFO

For symbols that carry pilot subcarriers, an input control signal is activated and a read operation is issued from External Port 2. For example, in IEEE 802.16e and 3GPP LTE release 8, indices of pilots in a certain symbol are shifted from the indices of pilots in the previous symbol. Figure 10 shows the pilot pattern in IEEE 802.16e in case of DL-PUSC. Pilots are arranged in the data memory with the same order they are received. Pilot patterns are arranged as follows: The first received pilot pattern and the third pilot pattern are stored in $DM1$, while the second pattern and the fourth pattern are stored in $DM2$. This arrangement helps to make the cross-correlation in Equation (16) easier. The CU is configured to perform the cross-correlation between DM1 and DM2 initialized at the correlation starting point (number of the contributing pilots is scalable). The output of the cross-correlation between the first received pilots (P1) and previously received pilots that have the same indices (P3 in 802.16e and LTE release 8) is stored in DM2. Every output complex data word is passed through the CORDIC algorithm with the *ANGLE* instruction to estimate its phase angle. Operations on the estimated angles are easier with the real data paths chosen for the implementation of the proposed engine. Estimated angles are multiplied with the corresponding known pilot indices and accumulated to get the term $\Sigma x_k \varphi_{k,l}$ in Equation (16). Then, the output phases are summed together to get the term $\Sigma \varphi_{k,l}$. Now, all terms of Equation (16) are known and stored in the internal registers or given as immediate values (like the number of pilots N_p). The estimated RCFO is added to the current total CFO and updated on the output ports. With

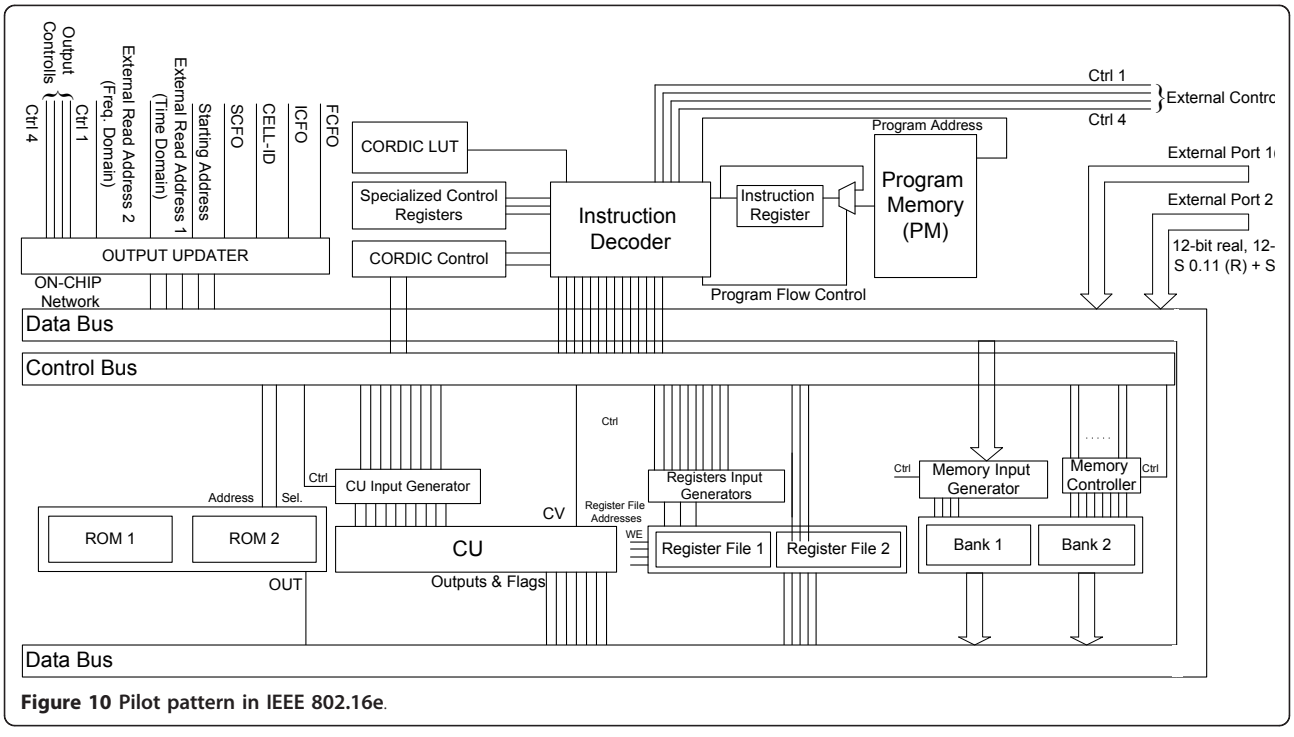

Figure 10 Pilot pattern in IEEE 802.16e.

another *SET* instruction, SCFO is fed to the ROB/STUFF correction Algorithm [13] connected with the SCFO port.

6 Performance evaluation

The performance of the proposed engine is measured against floating point Matlab model to assess its accuracy and the effect of round-off errors. The implementation efficiency is projected for the resulting core area and power consumption to implement the supported operations.

Bit-accurate fixed point simulations are used for functional verification, the generation of test vectors and building verification suites. The word length chosen for the execution unit is determined by the maximum precision needed in any algorithm. FCFO estimation needs 24-bit word, which turned out to be the largest number needed. To verify the accuracy of the chosen number of bits, comparison between floating point (FP) results and engine results for FCFO estimation (normalized by subcarrier spacing) is shown in Figure 11 with a correlation length of 96 and 60 in WiMAX and LTE, respectively.

The processor uses one embedded configurable unit (CU) besides the controller core. A total of 47.4 Kbit of memory is distributed among the main data banks, the reference ROMS, CORDIC LUT and a 10 Kbit program memory.

The Altera Stratix III FPGA kit is used to functionally verify the proposed design while Synopsys Design Compiler is used to estimate the chip area and the design static power consumption (ASIC design). The engine is coded using Verilog HDL, which is compatible with Synopsys Design Compiler. Mentor-Graphics Modelsim was used for functional simulations.

First the design was synthesized with the Altera Quartus II and programmed on a Stratix III (Stratix III EP3SC150 FPGA kit) FPGA to verify the design functionality. Then, The processor was synthesized in a 0.18μm CMOS process at a voltage of 1.8 V using Synopsys Design Compiler. The engine, without the memory, is estimated to occupy 1.1mm^2 and is estimated to consume an average static power of 7.9 mW when running at a speed of 120 MHz.

The proposed engine's control overhead is less than 10% of the total processing cycles. Pipelined processing of data is interrupted mainly by CORDIC subroutine in an average of 4820 cycles/symbol. A total of 95 MIPS are supported @ 120 MHz.

Engine features that helped to get a low chip area and power consumption are:

1. The use of optimized instruction set. This removed many control overheads and allowed faster executions.

2. Memory architecture that reduces memory interactions, even with complex vector instructions.

3. The mechanism of data movement to/from the CU and the memories.

4. Grouping of all units and increasing the degree of hardware reuse.

5. No cache memory is used.

Comparison between the engine results and other dedicated and configurable architectures results in terms of power consumption, not accounting for memory in our engine, is shown in Table 3 The powers of the stated architectures are scaled (technology & frequency

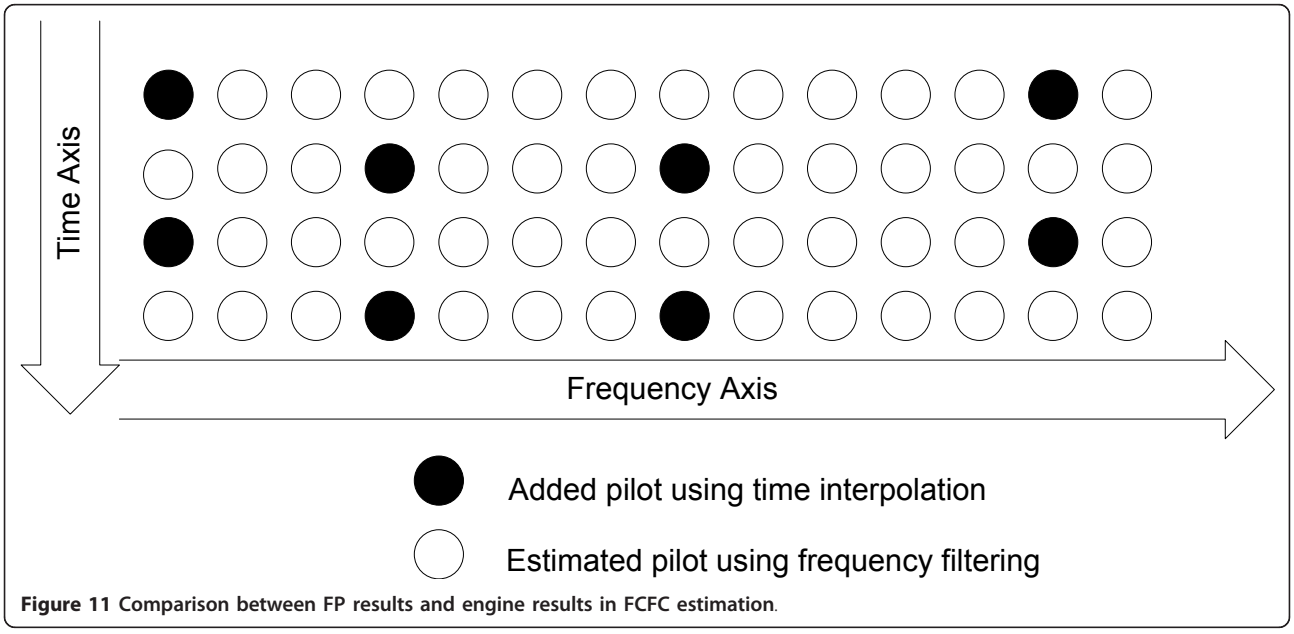

Figure 11 Comparison between FP results and engine results in FCFC estimation.

Table 3 Power comparison between the proposed engine and other architectures

Method	Implementation type	Task	Scaled power (*mW*)
[21]	Configurable	Symbol timing	148.58
[22]	Dedicated	Symbol timing	22.5
[23]	Dedicated	Symbol timing, FCFO	5.13 (Conventional)
			0.3541 (sign ML)
Proposed ASIP	Configurable	All in Table 4	8

scaling) to match the proposed engine. Based on the estimated power consumption of the proposed engine, the engine is more power. In [23], the proposed architecture supports only symbol timing and FCFO estimation. It is ten times power efficient as it uses only the signs of the data samples, but it causes large performance degradation.

Table 4 illustrates the cycle budget for various synchronization sub-tasks in IEEE802.16e and 3GPP LTE Rel. 8 for the most demanding parameters specified in Table 5

A comparison between FPGA dedicated hardware implementation results and engine results in WiMAX is shown in Table 6 for the parameters specified in Table 5 The area is reduced by a factor of three while the latency is slightly increased in some tasks but still meeting the standard requirements.

Syntheses results of the CU on a STRATIX III FPGA are tabulated in Table 7

We programmed the processor using one complete program that is executed at the beginning of every new frame. Otherwise, only the packet detection part of the code is executed till the beginning of an incoming frame. Tracking algorithms have proven to be the most power consuming as it is executed every symbol, unlike the acquisition algorithms that executes only at the beginning of reception.

To fully utilize the capabilities of the proposed engine, it is recommended to maximize the utilization of operations that have a fast execution phase in the instruction set. This requires understanding the proposed architecture with the attached instruction set when developing the algorithms. The assembly programmer should take care of various parameters that make the execution faster. For example, if pilot subcarriers are arranged at the same indices over time, there is no need to use special arrangements like the one shown in Figure 10 Normal arrangement of pilots beside each other will reduce the complexity when executing the rest of RCFO & SCFO estimation algorithm.

The engine is verified by running all the synchronization subsystem tasks in 802.16e and LTE release 8. It meets the timing requirements @ 120 MHz, while the maximum operating frequency is 149 MHz. To support larger systems in a scalable way, more than one CU can be inserted and controlled with the same control vector. In this case, accumulation outputs $O3$, $O4$ in Figure 4

Table 5 Parameters in IEEE 802.16e, LTE Rel. 8 (PSS, SSS are primary and secondary synchronizationsymbols)

parameter	IEEE 802.16e	LTE Rel. 8
Useful symbol time (μs)	91.428	66.667
Subcarrier spacing (KHz)	10.9	15
Max. CP length (samples)	256 (22.85 μs)	160 (10.41 μs)
Max. number of pilots	240	200
Max. preamble length	2048	72 (PSS), 72 (SSS)

Table 6 Comparison between dedicated implementations and engine results in WiMAX

Process	# Gates (k)	# Mul.	# Cycles	Max. Freq. (MHz)
FBD	49	1	21	147
FCFO	18	1	142	161
CS # ICFO	251	4	5112	107
RCFO	54	2	2644	145
SCFO	54	2	2640	145
Proposed	118	3	Table 4	159

Table 7 Synthesis results

FPGA type	Altera Stratix III EP3SC150
Total ALUT	752/113,600 (< 1%)
DSP blocks (18bit)	3
Dedicated logic registers	262/113,600 (< 1%)
Block RAM	54.4844/5499 Kbit (< 1%)
Max. clock frequency	159MHz

Table 4 Resulting cycle budget for different tasks

Processing Task	# Cycles (WiMAX)	Latency @ 120 MHz (μs)	# Cycles (LTE)	Latency @ 120 MHz (μs)
FBD	894/frame	7.45	606/frame	5.05
FCFO	214/frame	1.78	142/frame	1.16
CS # ICFO	10830/frame	90.25	3520/frame	29.33
RCFO	5288/symbol	44.117	4408/symbol	36.73
SCFO	5284/symbol	44.08	4404/symbol	36.70

are fed directly to another unit through the two input ports *I*9, *I*10.

7 Conclusion

In this article, a scalable embedded reconfigurable baseband ASIP for OFDM synchronization sub-system has been proposed. The processor can support a multitude of OFDM-based standards. Although the engine is optimized for OFDM synchronization purposes through detailed analysis of synchronization tasks in the different OFDM-based standards, it also offers a high degree of flexibility to support other simple and vector operations. Area, power and hardware complexity are reduced through reconfiguration of a single unit to support multiple special operations optimized for synchronization sub-system. The processor was successfully tested on IEEE 802.16e and 3GPP LTE Rel. 8 standards. Synthesis results show that it is efficient in terms of throughput, area and power consumption.

Acknowledgements

The authors would like to thank H.A.H. Fahmy, K. Osama, and H. Hamed for their invaluable comments while preparing this article.

Competing interests

The authors declare that they have no competing interests.

Received: 15 July 2011 Accepted: 26 March 2012
Published: 26 March 2012

References

1. Glossner J, Iancu D, Jin L, Hokenek E, Moudgill M: **A software-defined communications baseband design.** *IEEE Commun Mag* 2003, **41**:120-128.
2. Vogt N, Wehn T: **A Reconfigurable ASIP for convolutional and turbo decoding in an SDR environment.** *IEEE Trans Very Large Scale Integrat (VLSI) Syst* 2008, **16**:1309-1320.
3. Abdelall M, Shalash AF, Fahmy HAH: **A reconfigurable baseband processor for wireless OFDM synchronization sub-system.** *IEEE Int Symp Circ Syst* 2011.
4. Azar C, Ojail M, Chevobbe S, David R: **CERA: a channel estimation reconfigurable architecture.** *IEEE Int Conf Telecom-mun ICT* 2010, **17**:957-964.
5. *ETSI Broadband radio access networks (BRAN), Hiperlan type2; physical (PHY) layer, ETSI BRAN, Technical Report* 2000, **101**:475.
6. *IEEE802.11, Wireless LAN Medium Access Control(MAC) and Physical Layer (PHY) specification: High-Speed Physical Layer in the 5GHz Band, IEEE Std 802.11a-1999, IEEE Computer Society* 2000.
7. *IEEE, (IEEE) Standard for Local and metropolitan area networks, Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems, IEEE 802.16e-2005 and IEEE 802.16-2004/Corl-2005* 2006.
8. Speth M, Fechtel SA, Fock G, Meyr H: **Optimum receiver design for wireless broad-band systems using OFDM: Part I.** *IEEE Trans Commun* 1999, **47**:1668-1677.
9. Mei B, Lambrechts A, Verkest D: **Architecture exploration for a reconfigurable architecture template.** *IEEE Des Test Comput* 2005, **22**:90-101.
10. Ebeling C, Fisher C, Xing G, Shen M, Liu H: **Implementing an OFDM receiver on the RaPiD reconfigurable architecture.** *IEEE Trans Signal Process* 2004, **53**:1436-1448.
11. Poon ASY: **An energy-efficient reconfigurable baseband processor for wireless communications.** *IEEE Trans VLSI* 2007, **15**:319-327.
12. Dawid H, Meyr H: **CORDIC algorithms and architectures.** *Digital Signal Process Multimedia Syst* 1999, **2**:623-655.
13. Wu J-M, Chou C-H: *Baseband Sampling Clock Frequency Synchronization for WiMAX Systems, Institute of Communications Engineering National Tsing Hua University Hsinchu* 2005.
14. van de Beek JJ, Sandell M, Borjesson PO: **ML Estimation of Time and Frequency Offset in OFDM Systems.** *IEEE Trans Signal Process* 1997, **45**:1800-1805.
15. Speth M, Fechtel SA, Fock G, Meyr H: **Optimum receiver design for OFDM-based broadband transmission.** *II A case study IEEE Trans Commun* 2001, **49**:571-578.
16. Bhatt T, Sundaramurthy V, Zhang JC, McCain D: *Initial Synchronization for 802.16e Downlink,Signals, Asilomar Conference on Systems and Computers ACSSC* 2006, **40**:701-707.
17. Tang H, Lau KY, Brodersen RW: **Synchronization Schemes for Packet OFDM System.** *IEEE Int Conf Commun ICC* 2003, **5**:3346-3350.
18. Heiskala H, Terry JT: **OFDM wireless LANs: A Theoretical and Practical Guide.** *Sams Publishing, Indianapolis* 2002.
19. Hung K-C, Lin DW: **Joint detection of integral carrier frequency offset and preamble index in OFDMA WiMAX downlink synchronization.** *IEEE Wireless Communications and Networking Conference (WCNC)* 2007, 1959-1964.
20. Lin Y-C, Su S-L, Wang H-C: **A low complexity cell search method for IEEE 802.16e OFDMA systems.** *International Conference on Advanced Communication Technology (ICACT)* 2009, **2**:980-984.
21. Harju L, Nurmi J: **A synchronization coprocessor architecture for WCDMA/OFDM mobile terminal implementations.** *International Symposium on System-on-Chip* 2005, 141-145.
22. Troya A, Maharatna K, Krstic M, Grass E: **Low-power VLSI implementation of the inner receiver for OFDM-Based WLAN systems.** *IEEE Trans Circ Syst* 2008, **55**:672-686.
23. Li X, Zheng Y, Lai Z: **A low complexity sign ML detector for symbol and frequency synchronization of OFDM systems.** *IEEE Trans Consumer Electron* 2006, **52**:317-320.

doi:10.1186/1687-3963-2012-2
Cite this article as: Said *et al.*: Embedded reconfigurable synchronization & acquisition ASIP for a multi-standard OFDM receiver. *EURASIP Journal on Embedded Systems* 2012 **2012**:2.