# Low-Power VLSI Implementation of the Inner Receiver for OFDM-Based WLAN Systems

Alfonso Troya, *Member, IEEE*, Koushik Maharatna, *Member, IEEE*, Miloš Krstič, Eckhard Grass, Ulrich Jagdhold, and Rolf Kraemer, *Member, IEEE*

***Abstract*—In this paper, we propose low-power designs for the synchronizer and channel estimator units of the Inner Receiver in wireless local area network systems. The objective of the work is the optimization, with respect to power, area, and latency, of both the signal processing algorithms themselves and their implementation. Novel circuit design strategies have been employed to realize optimal hardware and power efficient architectures for the fast Fourier transform, arctangent computation unit, numerically controlled oscillator, and the decimation filters. The use of multiple clock domains and clock gating reduces the power consumption further. These blocks have been integrated into an experimental digital baseband processor for the IEEE 802.11a standard implemented in the 0.25-$\mu$m 5-metal layer BiCMOS technology from Institute for High Performance Microelectronics.**

***Index Terms*—Coordinate rotation digital computer (CORDIC), fast Fourier transform (FFT), orthogonal frequency-division multiplexing (OFDM), wireless local area network (WLAN).**

## I. Introduction

THE use of the orthogonal frequency division multiplexing (OFDM) technique is currently an active field of research in the area of communication and has been used to develop wireless local area network (WLAN) systems, for example the IEEE 802.11a/g standards [1], [2]. New standardisation processes already foresee the application of OFDM in future WLAN [3] and ultra-wide-band (UWB) systems [4]. The concept underlying such a system is to modulate a number of mutually orthogonal sub-carriers with the input data. This enables the realization of high-speed transmission systems. However, the entire system performance depends on maintaining the orthogonality of the sub-carriers and failing to maintain this property results in detrimental effects for example inter-carrier interference (ICI) and inter-symbol interference (ISI) during signal reception. In a real system, the orthogonality property of the sub-carriers can be disturbed during the RF up- and down-conversion and by the characteristics of the transmission channel.

Manuscript received July 24, 2006; revised December 15, 2006; June 11, 2007, and June 24, 2007. This paper was recommended by Associate Editor K. Chakrabarty.

A. Troya was with Institute for High Performance Microelectronics (IHP), Frankfurt (Oder) 15236, Germany. He is now with the European Patent Office, 2280-HV Rijswijk, The Netherlands (e-mail: atroyachinchilla@epo.org).

K. Maharatna was with Institute for High Performance Microelectronics (IHP), Frankfurt (Oder) 15236, Germany. He is now with the Electronics System Design Group, University of Southampton, Southampton SO17 B1J, U.K. (e-mail: km3@ecs.soton.ac.uk).

M. Krstic, E. Grass, U. Jagdhold, and R. Kraemer are with Institute for High Performance Microelectronics (IHP), Frankfurt (Oder) 15236, Germany (e-mail: Krstic@ihp-microelectronics.com; Grass@ihp-microelectronics.com; Jagdhold@ihp-microelectronics.com; kraemer@ihp-microelectronics.com).

Digital Object Identifier 10.1109/TCSI.2007.913732

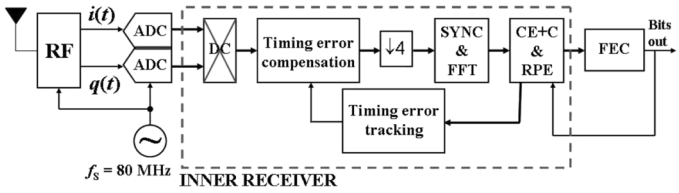



Fig. 1. General block diagram of the proposed IRx.

To make the system work efficiently, re-establishment of the orthogonality condition at the receiver is necessary. This is done by the so called *Inner Receiver* (IRx) [5] block shown in Fig. 1. In essence, two main operations are carried out inside the IRx, namely *signal acquisition* and *channel correction*. The acquisition operation is realized by means of a synchronization block, which should be able to perform reliable frame detection (FD), estimations for the carrier frequency offset (CFO) and symbol timing offset (STO). The channel correction operation is needed to estimate and compensate the channel transfer function (CTF), provided that orthogonality has been restored to a great extent by the synchronizer. The final goal is to supply the decoding and demodulator block with In-phase and quadrature-phase components of the signal that are as similar as possible to the originals.

Although the IRx is an integrated part of the OFDM-based WLAN system, its design complexity is frequently underestimated. Particularly, the synchronizer and channel estimator are extremely hardware intensive and power consuming units. In this article we investigate hardware and power-optimal realizations of the synchronizer and channel estimator for an efficient implementation of the IRx for IEEE 802.11a system. A *joint algorithm and architecture optimisation* has been undertaken to design these units using power consumption, silicon area, system latency and overall noise performance as the "quality/efficiency" parameters for the target system. The power consumption and silicon area have been considered as two of the main parameters since the system is targeted for mobile and portable applications where saving of battery life as well as the total size of the system are crucial. Latency has been considered based on the principles of operation of the IEEE 802.11a MAC protocol [1].

Some parts of the present work have been published in discrete and short form [6]–[12]. In this paper we provide a more detailed design description and an integrated view of the complete IRx design process. We start with the description of the main units of the synchronizer and channel estimator and then present the overall integration and control strategies adopted for the reduction of power dissipation. The rest of the paper is organized as follows. Section II looks into the algorithmic formulations of the synchronizer and channel estimator in brief (since detailed algorithmic analysis is beyond the scope of this

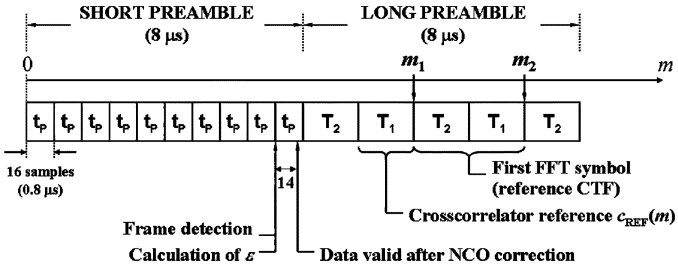



Fig. 2. Preamble symbols as defined by the 802.11a standard together with the timing schedule followed inside the synchronizer.

paper and has already been reported in [6] and [7]). Section III deals with the architectural description of the synchronizer, the channel estimator and the decimation filters. The integration strategy with power management and the control process of the whole system is described in Section IV. The result of the VLSI implementation of the complete baseband processor is detailed in Section V and conclusions are drawn in Section VI. It is to be noted that although the full description of the baseband processor design is beyond the scope of this paper, the fabrication and measurement results are presented to emphasize the real life performance of the synchronizer and channel estimator which contribute significantly to the overall functionality and area of the complete processor.

## II. Algorithmic Overview

### A. Synchronizer

One of the most computationally intensive and area consuming components of the IRx is the synchronizer. It needs to perform a variety of operations, namely FD, CFO estimation, the determination of symbol timing and the extraction of reference channel estimation (CE). The periodic structure of the preamble symbols points strongly towards the use of autocorrelators as the main building blocks in the synchronizer [7]. Moreover, the order in which the different operations are carried out strongly determines the final architecture. Fig. 2 shows the preamble symbols as defined in the IEEE 802.11a standard, whereas Fig. 3 depicts the synchronizer itself as proposed in [7]. The autocorrelators are defined by the duple ($\mathrm{N_d}$, $\mathrm{N_{avg}}$), where $\mathrm{N_d}$ stands for the delay and $\mathrm{N_{avg}}$ for the integration time (depth of the moving average block). The outputs of the autocorrelators, i.e., $J_{\mathrm{C}}(k)$ and $J_{\mathrm{F}}(k)$, provide the means to perform FD and CFO estimation, as described below.

*1) FD Mechanism:* The first operation to be carried out by the synchronizer is FD [6], [7]. For this purpose the shapes of the signals $|J_{\mathrm{C}}(k)|^2$ or $|J_{\mathrm{F}}(k)|^2$ are of significance due to the plateaus they display, as depicted in Fig. 4. However, since in a practical implementation the settling time of the automatic gain control (AGC) may affect the incoming signals resulting in false plateau detection, a FD mechanism based on the signal $|J_{\mathrm{F}}(k)|^2$ is preferred over a method based on $|J_{\mathrm{C}}(k)|^2$. If the plateau in $|J_{\mathrm{F}}(k)|^2$ can be reliably detected, that will be an indication that a frame is being received. Taking advantage of the shape of $|J_{\mathrm{F}}(k)|^2$, its differentiation will yield a peak corresponding to the beginning of the first plateau as shown in Fig. 5. Thus, a simple peak detector in conjunction with a differentiator circuit will be sufficient to detect the incoming frame.

*2) CFO Estimation and Correction:* The synchronizer provides an estimation of the normalized CFO, which is defined as $\varepsilon = f_\varepsilon/\Delta f$, where $f_\varepsilon$ stands for the actual CFO and $\Delta f$ is the sub-carrier spacing in the OFDM signal ($\Delta f = 312.5$ kHz in the 802.11a). The phases of $J_{\mathrm{F}}(k)$ and $J_{\mathrm{C}}(k)$ give a means to perform a *fine* $(\alpha)$ and a *coarse* $(\beta)$ CFO estimation, respectively. In [7] it is found that the range of normalized frequencies that can be estimated by means of the argument of the autocorrelator output signal depends on the parameter $\mathrm{N_d}$. Hence, from $J_{\mathrm{F}}(k)$, which is the output of an autocorrelator with $\mathrm{N_d} = 64$, the normalized CFO estimation will be bounded as $|\alpha| < 0.5$. Similarly, the phase of $J_{\mathrm{C}}(k)$ allows an estimation of the normalized CFO bounded as $|\beta| < 2.0$, since in this case $\mathrm{N_d} = 16$ as shown in Fig. 3. The final estimation of $\varepsilon$, which is considered to be bounded as $|\varepsilon| < 1.5$, will be a nonlinear combination of $\alpha$ and $\beta$. This is because although $\beta$ shows a linear dependency over the range of possible values of CFO, $\alpha$ does not. In this case, $\beta$ will only serve as a range pointer and will provide the integer value of $\varepsilon$ (either $-1$, $+1$ or 0) whereas $\alpha$ will provide the fractional part of the estimation.

The estimation of the CFO takes place exactly at that time instant when the incoming frame is detected. At that time instant the signals $J_{\mathrm{F}}(k)$ and $J_{\mathrm{C}}(k)$ are fed into an arctangent calculation unit which evaluates $\alpha$ and $\beta$ from $J_{\mathrm{F}}(k)$ and $J_{\mathrm{C}}(k)$, respectively. The correction of the CFO will follow naturally by using a numerically controlled oscillator (NCO) once $\varepsilon$ has been estimated.

*3) Symbol Timing Estimation:* While for CFO estimation the periodicity property of the short preamble symbols has been used, the long preamble symbols are used for the symbol timing estimation. This can be done by crosscorrelating the input frame with a reference signal directly obtained from the long preamble. The crosscorrelator can only be applied once the samples of the incoming frame have been fully corrected by the NCO and contain no frequency offset. The timing schedule inside the synchronizer is also shown in Fig. 2.

The fraction of the long preamble symbol selected as the crosscorrelator reference $c_{\mathrm{REF}}(m)$ corresponds to the sequence defined as $\mathrm{T_1}$ in Fig. 2. The reference has a length of 32 complex samples, which is the shortest possible length for this reference in order to obtain appropriate results after correlation [7]. The reference sequence used in the crosscorrelator is as follows:

$$\begin{aligned} c_{\mathrm{REF}}(31:0)^* = \{&1,1,0,0,1+j,j,j,j,j,0,1,1,0,0,1 \\ &1+j,1+j,0,1+j,1+j,j,1+j,1 \\ &1+j,j,j,1+j,1,1,1+j,j,1\}. \end{aligned} \quad (1)$$

When the preamble symbols go through the crosscorrelator, the output shows two major peaks at instants $m_1$ and $m_2$ (Fig. 2). Both peaks will occur when the portions $\mathrm{T_1}$ of the long preamble symbols are inside the crosscorrelator. For our purpose it is enough to detect the first peak by setting a certain threshold at the output of the crosscorrelator.

More sophisticated methods based on an active peak search may be used at the expense of increased latency. The 64 samples coming immediately after the first peak, i.e., the sequence

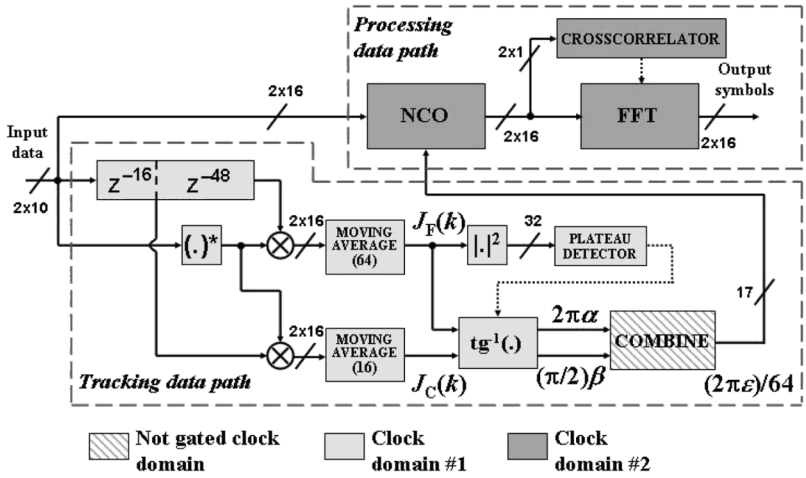



Fig. 3. Block diagram of the synchronizer showing the different clock domains defined therein. The diagram also shows the bit length of the most relevant signals.

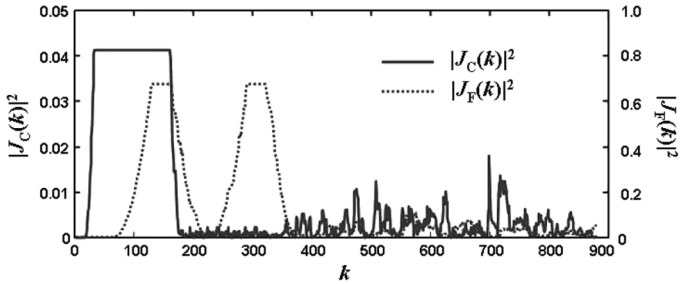



Fig. 4. Autocorrelation resulting signals when applying the preamble symbols: $J_{\mathrm{C}}(k)$ obtained with $\mathrm{N_d} = \mathrm{N_{avg}} = 16$, and $J_{\mathrm{F}}(k)$ obtained with $\mathrm{N_d} = \mathrm{N_{avg}} = 64$.

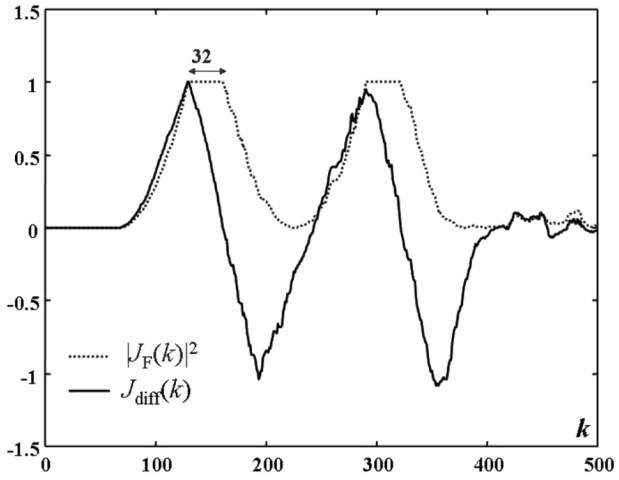



Fig. 5. Representation of $|J_{\mathrm{F}}(k)|^2$ and its differentiated signal showing that the peak of the differentiated signal coincides with the beginning of the plateau.

$\{\mathrm{T_2}, \mathrm{T_1}\}$ are fed into the fast Fourier transform (FFT) in order to extract the *reference CE*. In the 802.11a standard the long preamble symbol is defined as the sequence $\{\mathrm{T_1}, \mathrm{T_2}\}$, i.e., in our case a cyclic delay of 32 samples is introduced into a sequence of 64 samples. Therefore, the resulting sequence after FFT calculation has to be multiplied by $(-1)^k$, $k$ being the frequency variable, in order to eliminate the remaining linear phase.

### B. Channel Estimator

One important feature of OFDM signals is their native capability to overcome frequency selective channels. The spectrum of the OFDM signal is divided into a number of sub-carriers or sub-bands, each one being narrow enough to observe a frequency nonselective channel. Hence, an OFDM channel estimator need only estimate the value of the CTF at each sub-carrier. The correction of the CTF factor in a particular sub-carrier is done by means of a complex division, as shown in Fig. 6. The scheme in Fig. 6 represents a simplified version of the channel estimator proposed in [7], where the line widths do not represent bit widths but the number of complex samples. The channel estimator starts with an initial CE obtained from the preamble symbols (refer to the CTF estimation, Fig. 2), and afterwards makes updates of this reference by means of a decision-feedback mechanism. The channel zero-forcing (ZF) estimator requires a complex division, however this can be replaced by a complex multiplier, since the coefficients $(\hat{A}_{k,l-\mathrm{L}})^{-1}$ take a limited number of values that depend on the modulation scheme (either BPSK, QPSK, 16-QAM or 64-QAM), and can be stored beforehand.

The channel estimator further requires a delay buffer to synchronize the input data with the feedback data, which is available several OFDM symbols later. This latency is mainly due to the Viterbi decoder implemented in the forward error correction (FEC) block. In general it is also possible to replace the complex division by a complex multiplier to realize channel correction if the input data are multiplied by the complex conjugate version of the CE. As a result, the data input samples are phase corrected but not amplitude corrected. The amplitude correction is taken over by the demapper block. However, channel correction based on a complex division is absolutely necessary in our solution due to the residual phase correction (RPC) block depicted in Fig. 6. The algorithmic aspects of the RPC block are detailed in [7]. This novel RPC solution highly simplifies the required operations for residual phase estimation and correction, making it worth while to include a complex divider for channel correction. The solution for this complex divider is detailed in Section III-C.

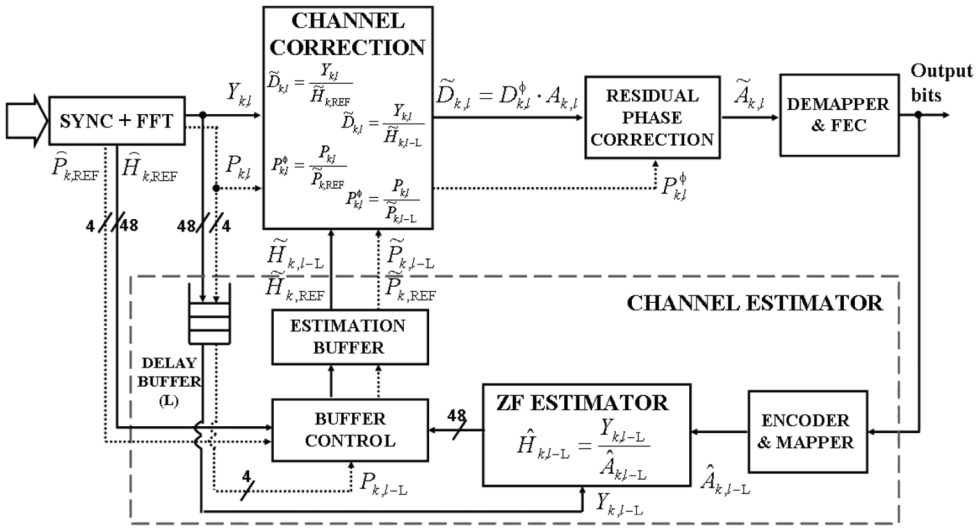



Fig. 6. Block diagram of the channel estimator and equalizer.

## III. Architectural Description

From the discussion presented in Section II-A we can identify two mutually exclusive time domain operations to be performed by the synchronizer, namely tracking of the training sequences and processing of the received symbols. Accordingly, we have divided the synchronizer architecture into two modules: the *tracking* data path and the *processing* data path, as shown in Fig. 3. The mutual exclusiveness of these paths has been exploited to reduce the power consumption of the complete synchronizer by assigning them two different clock domains and applying a clock gating technique.

### A. Synchronizer: Tracking Data Path

The main function of the tracking data path is to detect the beginning of the incoming frame by searching for the periodic structure of the preamble symbols and to estimate the CFO. The design of the main blocks of the tracking data path is described in the following subsections.

*1) Autocorrelator:* In our design, we use two autocorrelators having lengths of 16 and 64. While the autocorrelator of length 64 is used for FD and estimation of $\alpha$, the length 16 autocorrelator is used for estimation of $\beta$ only. However, instead of using two separate delay lines for these two autocorrelators, a single delay line of length 64 has been used and the data from its 16th and 64th positions are tapped off. To perform the basic autocorrelations two separate complex multipliers have been used and their outputs are accumulated by the moving average blocks, which are equivalent to finite-impulse response (FIR) filters with all the coefficients being 1. This allows a simple implementation of the moving average block by calculating its next output sample as the addition of the current output sample plus the new sample entering the first-in first-out (FIFO) and then subtracting the sample leaving the FIFO buffer.

*2) Plateau Detector:* The proposed plateau detector for the FD contains two blocks, namely a *differentiator* and a *peak detector* as shown in Fig. 7(a). The autocorrelation block together with the differentiator and the peak detector constantly sound the channel. When the peak detector identifies an absolute maximum at the output of the differentiator, the synchronizer will

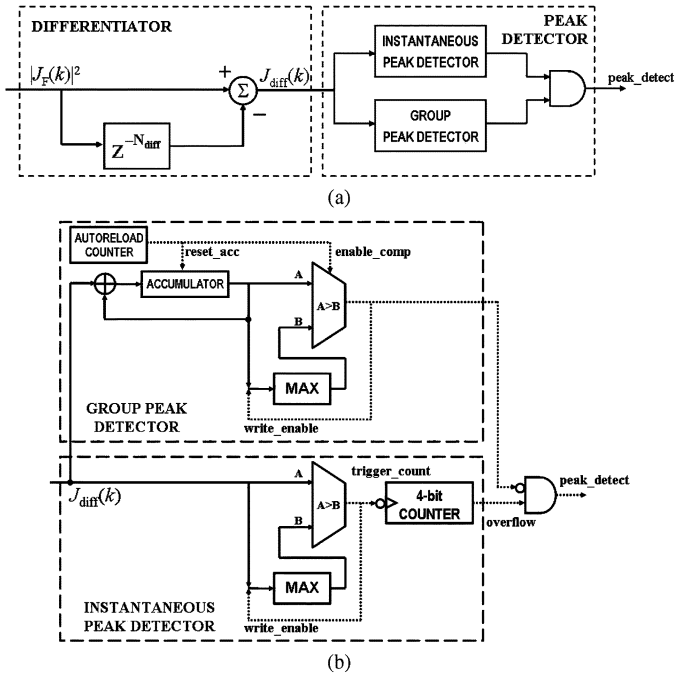



Fig. 7. Block diagram of: (a) the proposed FD algorithm, based on a differentiator and a peak detector and (b) detailed structure of the peak detector.

consider that a new frame has arrived and the CFO estimator will be activated. However, due to the noise and, more importantly, to the AGC effect, the peak detection is not a trivial task and a smart algorithm is necessary to distinguish the absolute from the relative maxima. For this purpose the peak detector is also divided into two blocks, namely *group peak detector* and *instantaneous peak detector*, as shown in Fig. 7(b). The instantaneous peak detector is basically a combination of a comparator and a counter. The present sample $J_{\text{diff}}(k)$ coming out from the differentiator is compared with the last recorded maximum $J_{\max}$ ($J_{\max} = 0$ at $k = 0$). As long as the sample $J_{\text{diff}}(k)$ is bigger than $J_{\max}$, the register storing $J_{\max}$ will be updated with the new sample $J_{\text{diff}}(k)$ as the new encountered maximum and the counter will be reset. If $J_{\text{diff}}(k)$ is smaller or equal than $J_{\max}$, the counter will be triggered and it will increase its count by one.

If this situation persists until the counter overflows, the instantaneous peak detector will activate a signal stating that a relative peak has been found inside the counting scope of the counter. On the other hand, the group peak detector is used to detect the falling edges in $J_{\mathrm{diff}}(k)$, and its main component is also a comparator circuit. Here, the input signal is accumulated in groups of six samples and the average value of the present group is compared with the previous one. If the value for the present group is smaller than the one stored from the previous group, then it can be interpreted as the starting point of the falling slope. If the group peak detector finds a falling slope at the same time as the instantaneous peak detector finds a relative peak, then the detected peak is actually an absolute peak and accordingly the detected frame is acquired by asserting the signal *peak_detect* shown in Fig. 7.

The influence of noise in the FD algorithm is mitigated by applying a threshold to the signal $|J_{\mathrm{F}}(k)|^2$ prior to plateau detection. At the time instant when the *peak_detect* becomes active, the sample from $J_{\mathrm{F}}(k)$ at that particular time instant is stored in a register and the arctangent calculator is activated. On the next clock cycle the signal $J_{\mathrm{C}}(k+1)$ is also input to the arctangent calculator.

*3) Arctangent Computation Unit:* The arctangent computation unit computes the phase angles of the complex signals $J_{\mathrm{F}}(k)$ and $J_{\mathrm{C}}(k)$ to generate the values of $\alpha$ and $\beta$, respectively. In practice, it outputs the values of $2\pi\alpha$ and $(\pi/2)\beta$ since the arctangent computation is bounded in the range $[-\pi, +\pi)$ and $|\alpha| < 0.5$, $|\beta| < 2.0$. Evaluation of arctangents is a nontrivial and computationally intensive task. For this purpose the coordinate rotation digital computer (CORDIC) algorithm has been used in its vectoring mode of operation in circular coordinates [13]. The most attractive point about CORDIC is that it requires only shift-and-add operations and thus the resulting hardware is very economical. However, the main limitations of the classical CORDIC approach are twofold: slow operation speed and the generation of a bulk scale factor, whose compensation requires extra circuitry and processing time, yet it should be noted that the scale factor compensation is not necessary in the evaluation of arctangent.

In [9], we have developed a virtually scaling-free CORDIC algorithm that eliminates the requirement of the scale factor compensation step, speeds up the CORDIC operation by adaptively executing only the required iterations and exhibits guaranteed convergence over the entire coordinate space. In this formulation it is considered a one-sided vector rotation only as opposed to the classical CORDIC, which approximates the target angle/final vector through to-and-fro vector rotation. The operation of the scaling-free CORDIC can be described by the following equation set [9]:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \prod_{i=p}^{b-1} \begin{pmatrix} 1-2^{-(2i+1)} & 2^{-i} \\ -2^{-i} & 1-2^{-(2i+1)} \end{pmatrix} \begin{pmatrix} x_i \\ y_i \end{pmatrix}$$
$$z_{i+1} = z_i + 2^{-i} \tag{2}$$

where $x'$ and $y'$ are the components of the final output vector, $z_i$ is the accumulated angle up to the $i$th iteration, $p = \lfloor (b - 2.585)/3 \rfloor$, $b$ is the wordlength, and $\lfloor x \rfloor$ rounds $x$ to the nearest integer towards minus infinity.

The basic idea for designing a virtually scaling-free CORDIC processor lies in extending the convergence range of the scaling-free CORDIC algorithm given by (2), which originally has a limited range of convergence, by applying a technique known as *domain folding*. Using *domain folding* we have shown that the result of the CORDIC operation with any target angle/vector (for rotation/vectoring operation) lying in the coordinate space can be computed by first mapping the angle/vector to a range $[0, \pi/8]$ and then executing the CORDIC operation on it. Fig. 8 shows a flowchart of the virtually scaling-free CORDIC algorithm operating in the vectoring mode. The complete theoretical formulation and performance results can be found in [9] and are omitted here for conciseness.

It is pretty straightforward to map the algorithm to hardware. However, from an implementation point of view, further optimization can be done by exploiting one sided vector rotation in the virtually scaling-free CORDIC framework.

Rotating the vector in one direction only essentially means that the accumulated angle can be described as a pure summation of powers of two. In this process, the iteration steps not actually needed are skipped. The final accumulated angle can be described by a bit pattern that contains logic "1" corresponding to the required iteration steps and logic "0" corresponding to the not required iterations. In essence, this technique eliminates all the unnecessary addition/subtraction operations along the angle accumulation (z) datapath and reduces the hardware cost drastically. This process is described by the flow chart in Fig. 8.

An implementation of this algorithm is described in detail in [10]. However, in this particular work we need only the arctangent computation part of the complete vectoring CORDIC. The arctangent computation unit consists of three modules *viz.* the *domain detection circuit*, the *basic CORDIC section*, and *the output unit* as shown in Fig. 9(a). An analysis of the target specification shows that a 16-bit arctangent computation unit is sufficient for our purpose. Two's complement arithmetic is used throughout the implementation.

The *domain detection circuit* is responsible for detecting the appropriate quadrant and the corresponding domain in which the vector lays. It consists of two comparators, two adders and a scaling unit of $\sqrt{2}$. The scaling circuit is realized using a shift-and-add technique and thus it is more economical than a full multiplier. It generates two 2-bit signals, namely *quad* and *domain*. While the *quad* signal indicates the initial quadrant in which the vector lays, the *domain* signal indicates the domain in the first quadrant from where the vector has been mapped. The entire operation requires one clock cycle.

The *basic data path unit* (called elementary rotational unit) used for constructing the CORDIC pipeline is shown in Fig. 9(b), where $i$ is the iteration index. It consists of four shifters and four adder/subtracters. For the pipelined implementation the shifters get reduced to only the wire connections and thus, the effective hardware complexity of the elementary rotational unit becomes equal to four adder/subtracters only. However, for a 16-bit implementation and for $i \geq 7$ the hardware complexity of the basic rotational unit becomes equal to two adder/subtracters since a right shift by 15-bits only results in retention of the sign bit. Each of these basic data path units is also equipped with a signal $d_i$ which is asserted when the

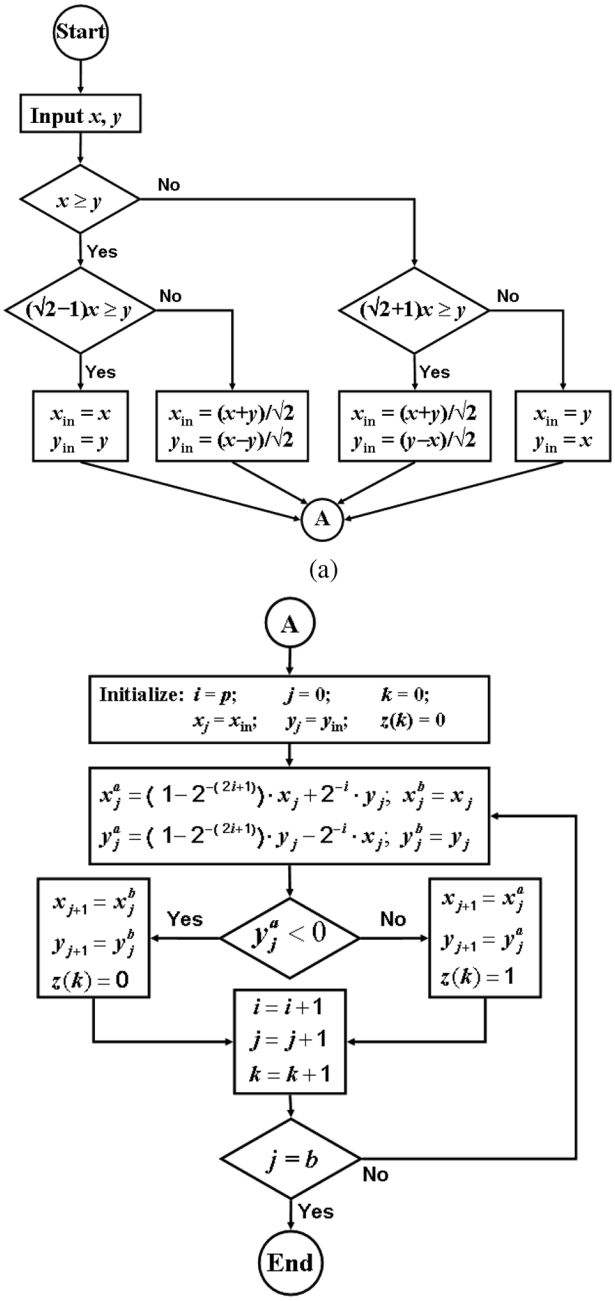



Fig. 8. Flow diagrams of the vectoring CORDIC: (a) sign/domain detector (domain folding) being $x$ and $y$ the real and imaginary parts, respectively; and (b) vectoring pipeline unit ($i \equiv$ index of rotation angle; $j \equiv$ index of pipeline stage; $k \equiv$ bit position at output angle).

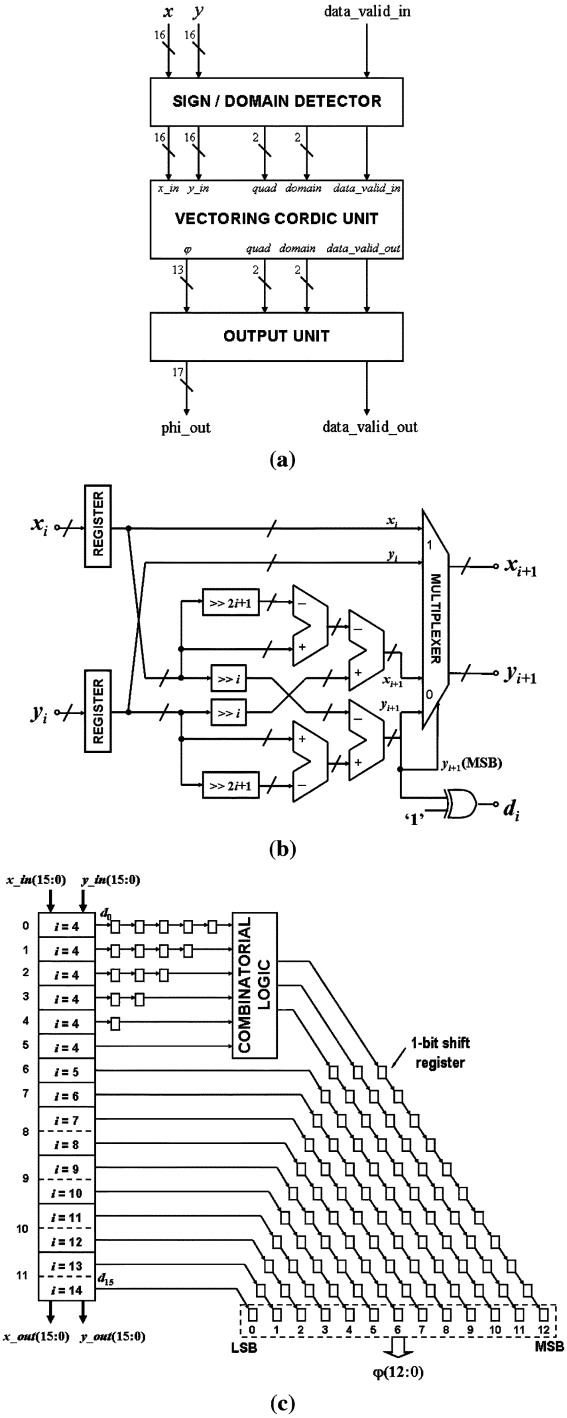



Fig. 9. Proposed arctangent calculator. (a) General block diagram showing the three processing steps. (b) Block diagram of a basic rotation unit. (c) Pipeline structure containing 16 basic rotation units for computation of arctangent in the range $[0, \pi/8]$.

rotation operation in that basic unit is performed. Otherwise the value of $d_i$ stays at "0." Fig. 9(c) shows the basic CORDIC pipeline. For a 16-bit implementation, according to the theory developed in [9], the smallest value for $i$ is 4. In order to cover the convergence range of $[0, \pi/8]$, we have used the $i = 4$ stage six times and the $i = 5, 6, \ldots, 14$ stages once each. The stage $i = 15$ is omitted since the right shift of a 16-bit number by 15-bit positions essentially results in the retention of the sign bit only. In order to balance the pipeline, we have merged the sections $i = (7, 8)$, (9, 10), (11, 12) and (13, 14) as shown by the dotted boundary in Fig. 9(c). Thus, the entire length of the pipeline is 12 stages. An array of registers is associated with different pipeline stages to keep the intermediate binary values of the accumulated angle. Depending on the decision of a particular stage, i.e., whether a rotation operation is accepted or not, a logic "1" or "0" (generated by the $d_i$ signal) is appended at the least significant bit (LSB) position of the register array and the value is passed to the next stage. However, a simple combinatorial circuit is required to interpret the decisions made by the six $i = 4$ stages. The decisions made in these sections

give the three most significant bits (MSBs) of the final representation of the accumulated angle. At the end, the basic CORDIC pipeline generates a 13-bit unsigned value for the accumulated angle which can be further processed by the output unit.

The data and the information about the quadrant (signal *quad*) and domain (signal *domain*) shown in Fig. 9(a) of the initial vector detected in the domain detection circuitry are transferred synchronously between two successive sections of the pipeline in a local register transfer manner. This means that each of the data in different sections of the pipeline has a *token* attributed to it that carries the information about the initial quadrant and domain of that particular data (not shown in Fig. 9(c)), which is used by the output unit to generate the final result.

The main hardware of the *output unit* consists of one adder and some registers. Depending on the *domain* and *quad* signals, it generates the final phase angle by addition/subtraction of the accumulated angle to/from $\pi/4$ or $\pi/2$ in accordance with the theory developed in [9]. At the same time it asserts an output enable signal for two cycles to indicate to the *Combine* block that valid data are now present at its input. All the operations in this unit are performed in one clock cycle.

The entire architecture exhibits a latency of 14 clock cycles. The total hardware complexity of the arctangent unit is 816 full adders and 553 registers. A big advantage of this design is that it completely eliminates all the arithmetic operations associated with the angle accumulation data path of a conventional CORDIC implementation. This results in about 35% saving in terms of registers although the required number of adders is slightly more than that of the classical CORDIC implementation when used for computing the arctangent function only.

*4) Combine Unit:* The combine unit is responsible for the nonlinear combination of $\alpha$ and $\beta$ (or strictly speaking $2\pi\alpha$ and $(\pi/2)\beta$) to generate the estimated normalized CFO $(2\pi\varepsilon)$. It consists of four 17-bit comparators, some logic circuitry with a small number of logic gates, two adder/subtractors and a multiplexer. The comparators are required for comparing the values of $(\pi/2)\beta$ with $\pm\pi/8$ and $\pm 3\pi/8$. The conditions $2\pi\alpha \geq 0$ or $2\pi\alpha < 0$ are determined by checking the most significant bit of $2\pi\alpha$. The adder/subtractors compute the terms $2\pi(1+\alpha)$ and $2\pi(-1+\alpha)$ in parallel and these are fed into the multiplexer along with $2\pi\alpha$. The output from the logic circuitry acts as the select line of this multiplexer, choosing the appropriate value for $2\pi\varepsilon$. However, the input data to the NCO needs to be $(2\pi\varepsilon/64)$, which is represented using 17 bits and is obtained by shifting the value of $2\pi\varepsilon$ by six bit positions to the right. This value is stored in the NCO register when the signal *rot_val_ok* is asserted by the combine unit. Note that no sign inversion of $(2\pi\varepsilon/64)$ is required since the results provided by the arctangent calculator already contain the correct sign information for frequency correction.

### B. Synchronizer: Processing Data Path

Activity of the processing data path starts when the signal *rot_val_ok* is asserted by the combine unit. This part of the synchronizer performs the carrier frequency error correction, estimates the symbol timing and obtains the reference CE. It consists mainly of the NCO, the 64-point FFT and the crosscorrelator. The designs of the NCO and the FFT blocks are described in the following subsections.

*1) NCO:* The main task of the NCO is to multiply the $n$th input data by $e^{jn\theta}$, where $\theta = (2\pi\varepsilon/64)$. A natural choice for carrying out such an operation is a complex multiplier with a look-up table (ROM memory) where the pre-computed values of one period of the sine and cosine functions are stored. This solution is also known as direct digital synthesis (DDS), and its main drawback is the size of the ROM memories. Hence, each memory position contains the phase value $\varphi_i = 2\pi i/2^{\mathrm{D}}$, $i = 0, \ldots, 2^{\mathrm{D}}-1$, where D is the number of addressing bits, i.e., the phase increment between two memory positions equals $2\pi/2^{\mathrm{D}}$. In our case the phase increment is given by $\theta = (2\pi\varepsilon/64)$, resulting in the design condition $2^{\mathrm{D}} = 64/\varepsilon$. The arctangent calculator has been designed to provide a minimum value of $\varepsilon = 0.01$, which results in $\mathrm{D} = 13$. Since each sample of the sine/cosine functions is represented with 16 bits, the total memory requirement of the DDS solution is 32 KB. Obviously the memory requirement could be reduced to the half if the sine/cosine functions were obtained by addressing one single memory, but this would double the required clock frequency and increase the control burden. On the contrary, the CORDIC algorithm in its rotational mode of operation in circular coordinate offers a flexible and hardware efficient realization of the NCO [13]. To design the NCO we have used once again the virtually scaling-free adaptive CORDIC method developed in [9]. The theoretical details of this unit, including the data for numerical accuracy and hardware comparison with similar kind of processors, can be found in [11], and is not provided here due to space restrictions. However, for the sake of completeness we present a brief description of it subsequently.

The general structure of the NCO is nearly the same as that for the arctangent computation unit. It consists of three modules namely, the domain detection circuitry, the basic CORDIC pipeline and the output unit. However, although the basic CORDIC pipeline of the NCO and the arctangent computation unit are the same, there are some subtle differences owing to the fact that while the arctangent unit computes the phase angle, the NCO actually rotates the input vector by a target angle. For implementation of the NCO we have selected a binary two's complement fixed-point representation and the decimal 1 is represented as 0010000000000000. Now, once again, because of the domain folding [9], all the angles lying in the coordinate space are effectively mapped into the range $[0, \pi/8]$. Thus, the effective maximum target angle that has to be computed is $\pi/8 = 0.392699$ radian. Using the definition of decimal 1 stated above, this angle can be represented in binary format as 0000110010010001 with an error of $O(2^{-16})$. Thus, from the implementation point of view, by representing the absolute value of any angle lying in this modified convergence range one can omit the first 4 MSBs and use the 13 LSBs. In our architecture we use this fact to reduce the total computation in the angle approximation data path.

The *domain detection circuitry* has two 16-bit wide data words for the two primary inputs (the real and imaginary part of the input data respectively) and a 17-bit data word for the target angle. In our implementation we have assumed that the target angle lies in the range $[-\pi, +\pi)$ and thus it can be rep-

resented by 17 bits. The principal task of the domain detection circuitry is to detect the sign and domain of the target angle and subsequently, it applies the *domain folding* technique to derive the 13-bit unsigned representation of the modified target angle $\phi$. Like the arctangent computation unit, it also generates two 2-bit signals, namely *domain* and *quad* which have identical functionality of the *domain* and *quad* signals of the arctangent computation unit. The entire operation in this unit is performed in one clock cycle.

The domain detection unit needs two 16-bit adders and two comparators. However, in both comparators one of the input variables is constant. The hardware complexity of each of these comparators is estimated conservatively as equivalent to one 4-bit adder.

As mentioned earlier, the *basic CORDIC pipeline* of the NCO uses an elementary rotational unit (shown in Fig. 9(b), without the circuitry for $d_i$ signal) similar to the one employed in the arctangent unit. Since we use one-sided vector rotation only [9], [11] the target angle is approximated as a pure summation of $2^{-i}$. As a result, the appropriate elementary rotational sections to be selected for a particular target angle have a one-to-one correspondence with the position of a logic "1" in the 13-bit unsigned representation of $\phi$. As an example, let us consider $\phi = 20^\circ$ (0.349 radian): The binary unsigned representation of this angle is 1011001010111. To achieve this target angle, the rotational sections that have to be activated are $i = 4$, 4, 4, 4, 4, 5, 8, 10, 12, 13, and 14. The three MSBs of $\phi$ are used to decide the number of active stages with $i = 4$. In the present example these bits are $101_{\rm b} = 5$. The deactivated elementary rotational sections (corresponding to the bit positions having logic "0") are bypassed. This implies a significant reduction of arithmetic computations that results in a reduction of power consumption. It is to be noted that as the range of $\phi$ is $[0, \pi/8]$, under no condition a logic "1" can arise at twelfth-, eleventh-, and tenth-bit position of the unsigned representation of $\phi$ simultaneously. To keep the pipeline operation intact, we feed the individual bits of the 13-bit unsigned representation of $\phi$ to the appropriate elementary rotational sections as an enable signal for that particular section through an array of single bit shift registers. The number of the shift registers corresponding to each section is chosen in such a manner that the appropriate section gets enabled at the correct clock cycle. To generate the enable signals for the six $i = 4$ sections, we need a simple logic combination of the twelfth, eleventh, and tenth bit. For the other elementary rotational sections the respective bits can be fed directly to the 0th shift register of the corresponding shift register array. This arrangement essentially mimics the procedure for adaptive selection of the elementary rotational stages described in [9] and eliminates the attendant hardware for the angle approximation data path completely. The structure of the basic pipelined CORDIC processor using this arrangement is shown in Fig. 10. Here the solid lines indicate the boundary of each of the elementary rotational sections whereas the dotted lines indicate the concatenated elementary rotational stages.

The last unit of the NCO block is the *output unit*. This circuit consists of two fixed scaling units of $1/\sqrt{2}$ and two adder/subtracter units. Each of the scaling units is realized using five 16-bit adders. Thus, the overall hardware complexity of this

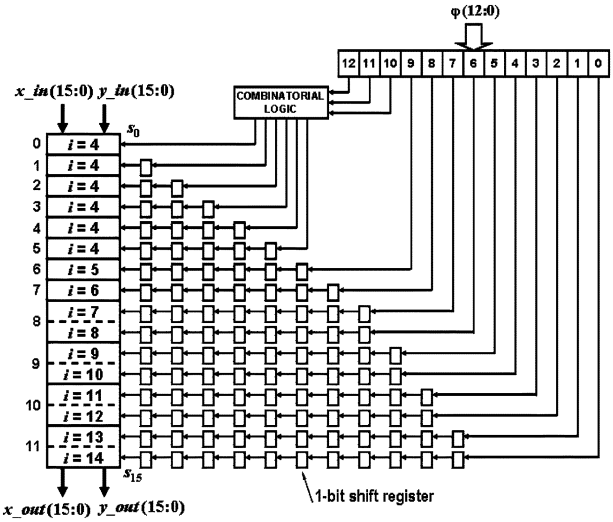



Fig. 10. Basic pipelined structure of the proposed NCO.

unit is equivalent to twelve 16-bit adders. Depending on the *domain* and *quad* signals this unit assigns the sign, applies either a scaling of $1/\sqrt{2}$ or simply passes the output vector emerging from the pipelined CORDIC to the primary output. All the operations in the output unit are completed in one clock cycle.

The total hardware cost associated with the NCO is equivalent to 768 full adders and 533 registers. Although the hardware complexity of the basic CORDIC pipeline is the same for both the arctangent computation unit and the NCO, the main difference comes from the domain detection and output units. Compared to the hardware cost of the conventional CORDIC processor with scaling circuitry, the design requires 22% less adders and 53% less registers, and on average 50% less computation to converge to the target angle [9], [11]. The significant saving in hardware as well as the number of computations reduces the power consumption of the design.

*2) Symbol Timing Estimator:* The processing task following the frequency offset correction is the symbol timing estimation. As has been discussed in Section II, this has been performed using the long preamble symbol and crosscorrelator. The crosscorrelator compares the input symbols with the reference symbol given in (1). However, from an implementation point of view, the complex crosscorrelator is a very hardware and power consuming component since it requires complex multiplication operations. To alleviate this problem we have used a simplified scheme for the crosscorrelator based on simple 1-bit XNOR multipliers. The basic multiplier cell used here is shown in Fig. 11(a). In this scheme instead of multiplying $b$-bit complex numbers, the XNOR multiplier only performs multiplication of the sign bit of the complex input values and assigns "1" when this value is positive and "0" otherwise. A further simplification of this structure is possible if one of the inputs is fixed and known beforehand. In this simplification the XNOR gates can be replaced by inverters resulting into smaller silicon area.

The overall structure of the crosscorrelator is shown in Fig. 11(b). The structure of each of the multiplier cells is decided upon the reference signal $c_{\rm REF}(31 : 0)^*$ (given by (1)), where it has already been explained that the reference is complex conjugated, hard-limited and order-reversed.

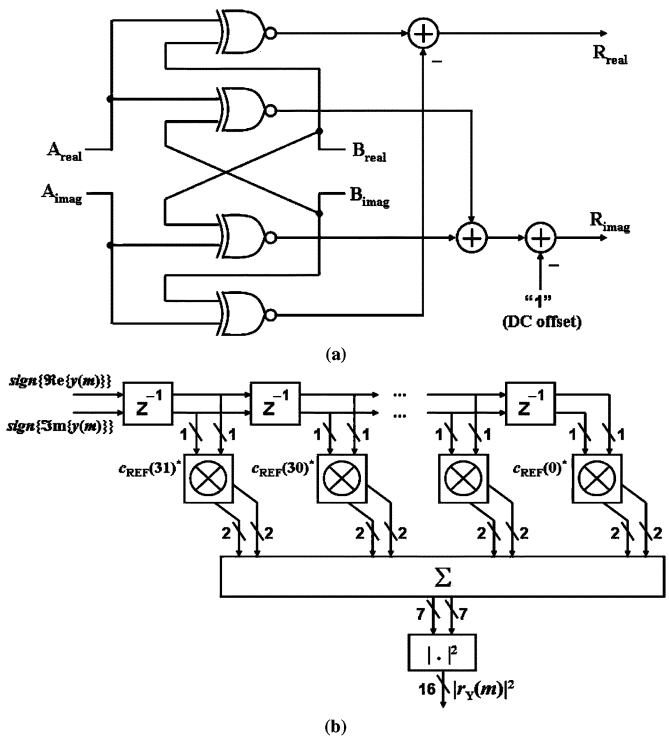



Fig. 11. Block diagram of: (a) sign-bit complex multiplier and (b) crosscorrelator with indication of bit lengths.

As discussed in Section II, the result of this crosscorrelation shows two major peaks at the instants $m_1$ and $m_2$ in Fig. 2. For our purpose it is sufficient to detect the first peak by setting a threshold value at the output of the crosscorrelator. A single real multiplier is used to compute the square of the absolute magnitude of the complex signal emerging from the crosscorrelator output. This value is compared with the predefined threshold value for detecting the occurrence of the first peak. The 64 incoming samples immediately after the first peak are fed into the FFT processor for the reference CE, which is used by the channel correction block.

*3) FFT Processor:* The third part of the processing data path is the FFT processor. The 64 data samples coming after the symbol timing detection are used as its input. However, before FFT operation, the data symbols go through a *cyclic prefix extraction* block since for every data symbol, 80 samples are expected but only the last 64 are fed to the FFT block. According to the 802.11a standard [1], a 64-point FFT [or inverse FFT (IFFT)] computation has to be performed in 4 $\mu$s. As discussed in [12] a radix-2 butterfly algorithm is not a good choice for achieving the required speed. Radix-4 FFT formulation has the potential to satisfy the speed but a radix-8 reformulation of the 64-point FFT gives some advantages in terms of power and silicon area. A particularly important feature for radix-8 formulation is the reduction of the number of nontrivial complex multiplications. This can be achieved by considering the fact that the twiddle factors for the first two columns of a decimation in time (DIT) 8-point FFT flow graph are either $\pm 1$ or $\pm j$ whereas the twiddle factors required for the third column can be easily realized using only a shift-and-add principle. Considering this we have selected a radix-8 implementation of the 64-point FFT.

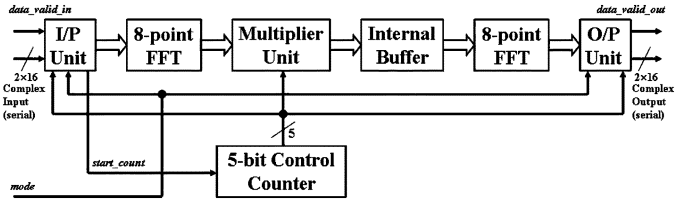



Fig. 12. Block diagram of the 64-point FFT/IFFT processor.

The detailed design of the FFT/IFFT processor can be found in [12]. Here we provide a brief overview of the design for completeness.

As discussed in [12], the 64-point FFT can be computed by performing two 8-point FFTs. Here every eighth sample of the input data set forms a data subset that undergoes 8-point FFT computation. The result of the first 8-point FFT has to be multiplied by the corresponding inter-dimensional coefficients $W_{64}^{ls}$ before the computation of the second 8-point FFT. In this formulation 49 nontrivial complex multiplications are needed for a 64-point FFT computation [12]. These can be performed using only eight pairs of unique nontrivial twiddle factors and thus less storage space is required for the twiddle factors compared to the other algorithms.

The block diagram of the FFT processor is shown in Fig. 12. It consists of an input buffer of size $56 \times 16$ bits, two 8-point FFT units, one multiplier unit, one internal storage buffer of size $64 \times 16$ bits and an output buffer of size $56 \times 16$ bits. The incoming data samples are fed into the input buffer in a serial manner and the outputs are tapped off from the positions $8m$ where $m \in \{0, \ldots, 7\}$, which serve as the inputs to the first 8-point FFT. At every clock cycle the data at position $k$ of the input buffer is shifted to the position $k-1$, $k$ ranging from 1 to 56. Thus, after 56 cycles (when the buffer is full) the appropriate data becomes self-aligned with the inputs of the first 8-point FFT. This arrangement substantially reduces the number of globally multiplexed wires. Owing to the reduced complexity of the 8-point FFT we have realized it in a fully parallel manner and thereby perform the 8-point FFT in one clock cycle. The data coming out from the first 8-point FFT are then fed into the multiplication unit where they are multiplied by eight pairs of twiddle factors. To save area and power, we have utilized the shift-and-add principle for doing the multiplication operation and thus each of the twiddle factors reduces to a hard-wired connection of adders [12]. Eight such hard-wired twiddle factors are placed in parallel and hence eight complex multiplications are carried out in one single clock cycle. The resulting data are stored in an intermediate buffer from where they are fed into the second 8-point FFT unit. To reduce the number of globally multiplexed wires an approach similar to the input buffer (downward shifting of data at every clock cycle) has been adopted. The data coming out from the second 8-point FFT are fed at the $8t$ positions, $t \in \{0, \ldots, 7\}$, of the output buffer and the data residing in the buffer are shifted downwards by one position at every clock cycle to generate the serial output in the correct order.

Although it is possible to use a serial architecture for the multiplier unit, we have avoided it for two reasons: Firstly, using a serial architecture incurs significantly more global wiring since

each output of the 8-point FFT (16-bit complex data) has to be multiplexed at the input of the multiplier unit. There are significant routing overheads involved in this. Secondly, if a complex multiplier is used with a small RAM to store the values of the twiddle factors, then, in order to maintain the timing, it needs to be run at a frequency eight times higher than the base frequency (20 MHz). This leads to high power dissipation as well as complex multiple-frequency design. On top of this, since data come from the 8-point FFT in parallel, one needs to keep a buffer to hold the data set, which once again consumes significant area. Alternatively, a serial 8-point FFT architecture may be used with a complex multiplier. Although it eliminates the requirement of buffering, it would increase the data multiplexing, and hence global routing, at the input of the 8-point FFT enormously. In addition, in order to meet the timing specification, the entire processor would need to operate at a much higher clock frequency than 20 MHz, which once again would result in high power dissipation.

To control the entire operation we have used a 5-bit master control counter; the operations of the different parts of the architecture are synchronized with the count number. On the other hand, two separate 6-bit counters have been used for controlling the input and output unit. The overall FFT block can be enabled by asserting a signal *data_valid_in*. The input control counter gets incremented at every clock cycle to track the data input phase. Once the 56th datum is entered, the input control counter enables the master control counter and from then on the operations are controlled by the master counter. Similarly, on availability of the first output data set to the output buffer, the master counter enables the output counter, which then controls the data movement in the output buffer. Upon activation the output counter also asserts the signal *data_valid_out* to indicate that the valid data is now available at the output.

This architecture performs the parallel-to-parallel FFT in 23 clock cycles and thus a clock rate of 20 MHz is sufficient to satisfy the timing requirements.

A detailed performance comparison of this processor with some other existing 64-point FFT processors is made in [12] considering the number of cycles, silicon area and power consumption. The comparison shows that this processor offers significant advantages in all the three areas. These performance parameters are omitted from this paper; interested readers are referred to [12] for a detailed analysis.

The last operation in the processing data path is *channel reordering*. After FFT computation, the output is arranged in natural order, which needs to be changed for further processing. This is done in the output buffer of the FFT processor itself by a simple hard-wired mapping of the data emerging from the second 8-point FFT to the output register.

### C. Channel Correction

As described earlier, the chosen method for CE is based on a decision-directed approach, in which the demodulated bits (output from the Viterbi decoder) are used by the algorithm as a reference to perform the CE. The CE block requires a number of buffers to compensate for the latency of the Viterbi decoder. This latency comes from the fact that the CE algorithm works on a symbol basis whereas the number of bits transmitted on each symbol varies according to the transmission rate. The processing latency of the channel estimator block in the forward direction is fixed (650 ns) but the loop latency varies due to the different processing times required by the Viterbi decoder for different data rates. The loop delay (the time elapsed between the data being available at the output of the channel estimator and the coded information for the same symbol appearing at the output of the feedback loop) varies between 11.3 $\mu$s (54 Mbps) to 15.05 $\mu$s (6 Mbps.) This means that the CE needs to buffer up to four OFDM symbols in the worst case. Nevertheless, the loop delay does not affect the MAC layer, but the processing latency, i.e., 650 ns.

A key component in the CE is the complex divider. The division of two complex values $a = a_{\mathrm{R}} + j \cdot a_{\mathrm{I}}$ and $b = b_{\mathrm{R}} + j \cdot b_{\mathrm{I}}$ results in $c = c_{\mathrm{R}} + j \cdot c_{\mathrm{I}}$ with

$$\begin{aligned} c &= \frac{a_{\mathrm{R}} + j a_{\mathrm{I}}}{b_{\mathrm{R}} + j b_{\mathrm{I}}} \times \frac{b_{\mathrm{R}} - j b_{\mathrm{I}}}{b_{\mathrm{R}} - j b_{\mathrm{I}}} \\ &= \underbrace{\frac{(a_{\mathrm{R}} b_{\mathrm{R}} + a_{\mathrm{I}} b_{\mathrm{I}})}{|b|^2}}_{c_{\mathrm{R}}} + j \underbrace{\frac{(a_{\mathrm{I}} b_{\mathrm{R}} - a_{\mathrm{R}} b_{\mathrm{I}})}{|b|^2}}_{c_{\mathrm{I}}}. \end{aligned} \tag{3}$$

Nevertheless, the following simplification of (3) is possible:

$$\begin{aligned} c &= \frac{1}{|b|^2} \left( (a_{\mathrm{R}} b_{\mathrm{R}} + a_{\mathrm{I}} b_{\mathrm{I}}) + j(a_{\mathrm{I}} b_{\mathrm{R}} - a_{\mathrm{R}} b_{\mathrm{I}}) \right) \\ &= \frac{1}{|b|^2} \left( (a_{\mathrm{R}} b_{\mathrm{R}} + a_{\mathrm{I}} b_{\mathrm{I}} + a_{\mathrm{R}} b_{\mathrm{I}} - a_{\mathrm{R}} b_{\mathrm{I}}) \right. \\ &\qquad \left. + j(a_{\mathrm{I}} b_{\mathrm{R}} - a_{\mathrm{R}} b_{\mathrm{I}} + a_{\mathrm{R}} b_{\mathrm{R}} - a_{\mathrm{R}} b_{\mathrm{R}}) \right) \\ &= \frac{1}{|b|^2} \left( (a_{\mathrm{R}} (b_{\mathrm{R}} - b_{\mathrm{I}}) + b_{\mathrm{I}} (a_{\mathrm{I}} + a_{\mathrm{R}})) \right. \\ &\qquad \left. + j \left( a_{\mathrm{R}} (b_{\mathrm{R}} - b_{\mathrm{I}}) + b_{\mathrm{R}} (a_{\mathrm{I}} - a_{\mathrm{R}}) \right) \right). \end{aligned} \tag{4}$$

In (4) the values $c_{\mathrm{R}}$ and $c_{\mathrm{I}}$ have a common element $a_{\mathrm{R}} \cdot (b_{\mathrm{R}} - b_{\mathrm{I}})$, which has to be calculated only once. Comparing (3) with (4) shows that the latter requires three more addition operations than the former, but at the same time it saves one multiplication operation, which comparatively is much more computationally intensive than the three adders. However, two real division operations are to be performed as explained next.

The realization of the real dividers is based on a pipelined implementation of the linear CORDIC algorithm, where the relationship between the input and the output vectors may be written as follows [13]:

$$\begin{pmatrix} x_{\mathrm{out}} \\ y_{\mathrm{out}} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -\theta & 1 \end{pmatrix} \begin{pmatrix} x_{\mathrm{in}} \\ y_{\mathrm{in}} \end{pmatrix}. \tag{5}$$

Again, the CORDIC algorithm is used in vectoring mode by accumulating only the necessary rotations until the condition $y_{\mathrm{out}} = 0$ is reached (or at least $y_{\mathrm{out}}$ becomes a machine zero.) From (5) it is readily seen that in this case $\theta = y_{\mathrm{in}}/x_{\mathrm{in}}$.

In our implementation it is considered that both $x_{\mathrm{in}}$ and $y_{\mathrm{in}}$ are always positive variables. The input stage to the divider should ensure this condition and provide the sign information in the form of a token that has to be propagated through the pipeline. Similar to the circular CORDIC implementation (see

Section III), the angle $\theta$ is expressed as an addition of $b$ elementary positive angles $\alpha_i = 2^{b-1-i}$, $i = 0, \ldots, b-1$, yielding the following modifications to (5):

$$\begin{pmatrix} x_{\text{out}} \\ y_{\text{out}} \end{pmatrix} = \prod_{i=0}^{b-1} \begin{pmatrix} 1 & 0 \\ -\alpha_i & 1 \end{pmatrix} \begin{pmatrix} x_{\text{in}} \\ y_{\text{in}} \end{pmatrix}. \tag{6}$$

Each stage $i$ will have three input signals $(x_i, y_i, z_i)$ and three output signals $(x_{i+1}, y_{i+1}, z_{i+1})$ obtained as follows:

$$x_{i+1} = x_i \tag{7.a}$$
$$y_{i+1} = y_i - \alpha_i x_i \tag{7.b}$$
$$z_{i+1} = z_i + \alpha_i \tag{7.c}$$

with the initialization $x_0 = x_{\text{in}}$, $y_0 = y_{\text{in}}$ and $z_0 = 0$.

At each stage $i$ the input $x_i$ is right-shifted by $b-1-i$ positions. Since $y_{i+1}$ should always be positive, the operation in (7.b) is only performed if $y_i > (2^{b-1-i})x_i$ and if the $b-1-i$ MSBs in $x_i$ are all zero. In this case (7.c) is equivalent to setting the bit at position $b-i$ in $z_i$ to a logic "1." If both conditions are not satisfied, then $y_{i+1} = y_i$ and $z_{i+1} = z_i$. We have realized a 16-bit implementation: the input $x_{\text{in}}$ is represented using 16 bits whereas the input $y_{\text{in}}$ requires 32 bits. The whole operation has a latency of 16 clock cycles, but after this the data delay instead is only one clock cycle.

### D. Decimation Filter

Interpolation and decimation filters are necessary to adapt the sampling rates at the analog-digital interface as shown in Fig. 1. In this section we concentrate on the decimation filters, but the same filters may be used for interpolation with slight modifications. The solution is driven by two properties that must be exhibited by the filter.

1) It must show a constant group delay in order to minimize the error vector magnitude after the FFT operation.
2) Although the filter must have a very steep transition band, its order should be small so as to avoid ISI.

The decimation is realized by two concatenated identical decimators with a factor $\mathrm{N_{DEC}} = 2$. Each decimator contains a low-pass filter $\mathrm{H(z)}$ that is expressed as an average of two *all-pass* filters $\mathrm{H}_{a0}(\mathrm{z})$ and $\mathrm{H}_{a1}(\mathrm{z})$, i.e.

$$\mathrm{H(z)} = \frac{\mathrm{H}_{a0}(\mathrm{z}) + \mathrm{H}_{a1}(\mathrm{z})}{2} \tag{8}$$

with

$$\mathrm{H}_{a0}(e^{j\omega}) = e^{j\Phi_0(\omega)}$$
$$\mathrm{H}_{a1}(e^{j\omega}) = e^{j\Phi_1(\omega)}. \tag{9}$$

Both phases, $\Phi_0(\omega)$ and $\Phi_1(\omega)$, are monotonically decreasing functions. It is straightforward to see that the passband of $\mathrm{H}(\omega)$ will occur when $\Phi_0(\omega) \approx \Phi_1(\omega)$ whereas, the stopband will occur when $\Phi_0(\omega) - \Phi_1(\omega) \approx \pi$.

Another way to express a low-pass filter is by using a polyphase representation [14], i.e.

$$\mathrm{H(z)} = \frac{1}{2}\left(\mathrm{H}_0(\mathrm{z}^2) + \mathrm{z}^{-1}\mathrm{H}_1(\mathrm{z}^2)\right). \tag{10}$$

In our particular scenario, both filters $\mathrm{H}_0(\mathrm{z}^2)$ and $\mathrm{z}^{-1}\mathrm{H}_1(\mathrm{z}^2)$, must be all-pass filters. Therefore, in order to design a low-pass filter with approximately linear phase in the passband, the following optimization problem has to be solved.

1) $\Phi_0(\omega) = -(2R-1)\omega$; for $\omega \in [0, \omega_{\mathrm{P}}]$.
2) $\Phi_0(\omega) = -(2R-1)\omega + \pi$; for $\omega \in [\omega_{\mathrm{S}}, \pi]$.

The above are under the assumption that $\mathrm{z}^{-1}\mathrm{H}_1(\mathrm{z}^2) = \mathrm{z}^{-2R+1}$ (all-pass filter of order $2R-1$.) The parameters $\omega_{\mathrm{P}}$ and $\omega_{\mathrm{S}}$ are the passband and stopband frequencies, respectively. Furthermore, the filter $\mathrm{H}_0(\mathrm{z}^2)$ needs to be of order $2R$ according to [15] in order to obtain a low-pass characteristic.

The previous statement is a nontrivial optimization problem that has been tackled by several authors in the literature [15]–[17]. In our case, the solution proposed in [15] has been adopted, since it seems to provide the simplest algorithm. The number of coefficients, and consequently, the number of degrees of freedom for an $M$th-order linear phase low-pass filter is $R = (M+1)/4$. In our implementation we chose a value of $R = 2$. Fig. 13(a) and (b) depicts the magnitude and phase of the obtained filter $\mathrm{H(z)}$with $\omega_{\mathrm{P}} = 2\pi(9/40)$ and $\omega_{\mathrm{S}} = 2\pi(11/40)$, which shows an attenuation in the stopband of 20 dB. The final implementation of the whole decimator is shown in Fig. 13(c), where we make use of the so-called Noble Identity #1, as provided in [14]. As a result, the filters do not need to run at 80 and 40 MHz but at 40 and 20 MHz, with the subsequent power saving. The overall attenuation in the stopband is 40 dB. By invoking Noble Identity #2 a similar simplification can be achieved for the interpolation filters [14].

After synthesis, the decimator resulted in a cell area of 0.32 $\mathrm{mm}^2$ with an estimated power consumption of 4.3 mW.

## IV. System Integration: Control Strategy and Power Management

As mentioned earlier, the control of the complete IRx is not trivial. Instead of using a single central controller it is easier to employ distributed control with *token flow*. Considering this, the main architecture of the IRx has been configured in such a way that it forms a linear pipeline structure. Each module has been provided with an input data valid signal. Assertion of this signal activates the respective modules. Similarly, upon completion of the computation, each module asserts an output data valid signal which indicates that the output data is now valid. The output data valid signal of one module is tied to the input data valid signal of the next module in the pipeline thus forming a distributed self controlled architecture. The distributed control not only eases the controlling of the entire architecture but also helps to reduce the power consumption using a clock gating approach.

Unfortunately the design flow we have used is based on standard cell and therefore it was not possible for us to use transistor level power optimization techniques for the power critical blocks. To reduce the power consumption we have adopted

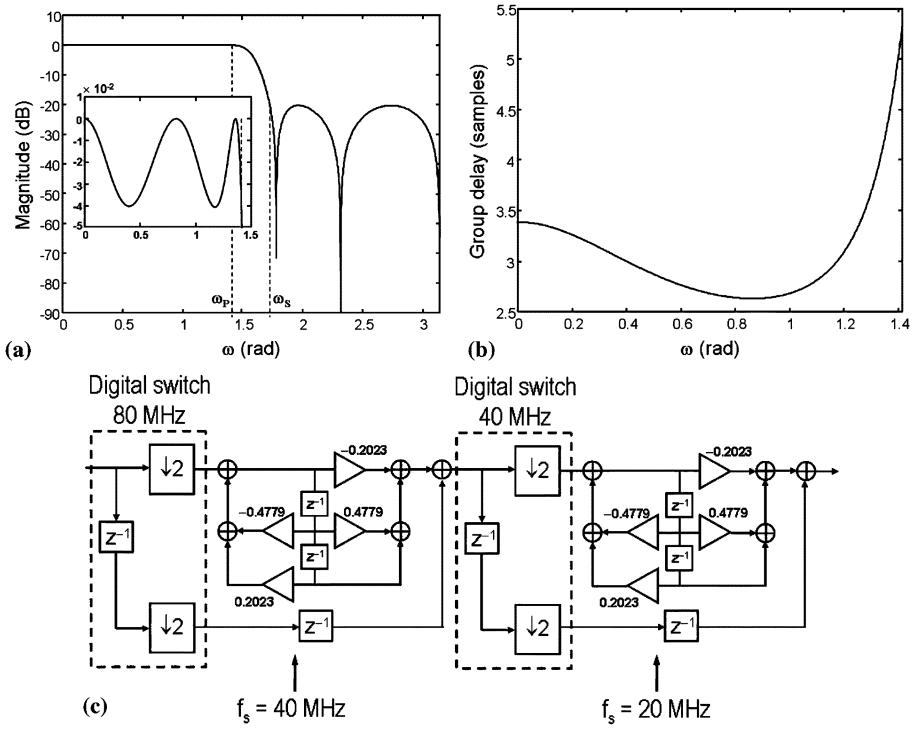



Fig. 13. Design of the decimation low-pass filter with $R = 2, \omega_{\rm P} = 2\pi(9/40)$ and $\omega_{\rm S} = 2\pi(11/40)$. (a) Magnitude of $\rm H(z)$. (b) Group delay of $\rm H(z)$. (c) Final filter design as the concatenation of two low-pass filters $\rm H(z)$.

two fold approaches. Firstly, the algorithms of all the computationally intensive blocks are reformulated in such a way that they require less number of computations and their architectural level mapping has been done with particular emphasis on reduction of the number of global wires as discussed in the foregoing sections. Secondly, the timing behaviour of the blocks has been carefully analyzed and the design has been divided into different clock domains. As an example, we have shown the clock domain partitioning for the synchroniser in Fig. 3. Extensive clock gating has been applied throughout the design to reduce the unwanted clock transition and thereby reducing the power consumption.

In the particular case of the synchronizer, although a single CORDIC implementation could have performed both the arctangent calculation, and the NCO operation, we mapped both operations onto different processors. This separation not only eases the control flow but also helps to reduce the power consumption when clock gating is applied. Please note that once the FD and the value of $\varepsilon$ are determined, the tracking data path (including the arctangent calculator) can be switched off completely through clock gating. Using this approach, a power reduction of 54% has been achieved at the expense of 33% increase in silicon area compared to the single CORDIC alternative.

For the final integration of the synchronizer and the CE in an experimental baseband processor we decided to omit the FFT unit from the synchronizer. This is because the data path of the baseband itself has an IFFT/FFT unit which could be reused to perform the reference channel extraction. Since the FFT is one of the most power and hardware consuming blocks, the reuse of this block is preferred at the expense of a slightly more complicated control structure.

TABLE I
BASEBAND PROCESSOR AREA AND ESTIMATED POWER PROFILE

| Hardware part | Cell Area ($mm^2$) | Power (mW) |
|---|---|---|
| Interleaver | 0.501 | 13 |
| Pilot insertion | 0.820 | 31 |
| GI insertion | 1.011 | 4 |
| Input buffer | 0.201 | 1 |
| TX BIST | 0.130 | 1 |
| **Total TX** | **3.064** | **63** |
| Synchronizer | 6.183 | 25 |
| Channel Estimator | 5.297 | 15 |
| Viterbi decoder | 5.910 | 53 |
| CE buffer | 1.446 | 13 |
| Deinterleaver | 1.786 | 21 |
| Demapper | 0.422 | 1 |
| RX BIST | 0.247 | 1 |
| **Total RX** | **22.201** | **145** |
| **FFT / IFFT** | **7.556** | **41** |
| **Clock tree and pads** | **---** | **144** |
| **Total** | **32.851** | **393** |

## V. VLSI IMPLEMENTATION AND COMPARISON

The result of our work has been the realization of an experimental baseband processor chip compatible with the IEEE 802.11a standard, in which the proposed IRx has been integrated along with the other system components. The synthesis results for the baseband chip are shown in Table I. These results have been obtained using Institute for High Performance Microelectronics (IHP)'s 0.25-$\mu$m 5-metal layer BiCMOS process. The power estimations at the layout level have been provided by

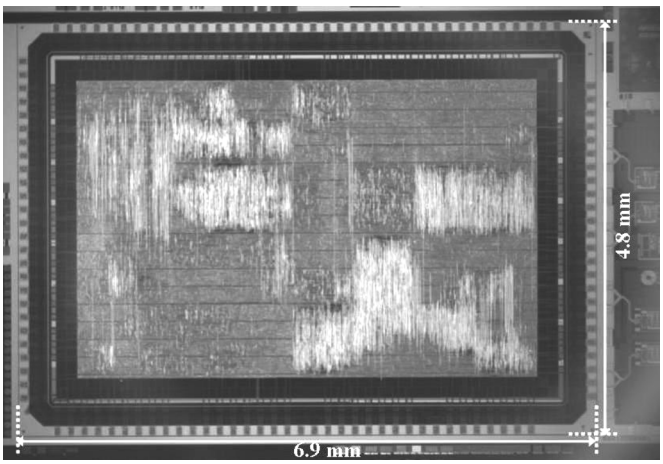



Fig. 14. Die photo of IEEE 802.11a experimental baseband processor chip developed at the IHP.

the Synopsys' PrimePower tool, which supplies fairly realistic power estimates based on the switching activity inside the chip under certain operating conditions. However, in Table I we keep the power consumption of the FFT/IFFT module separated from the Transmitter (TX) and Receiver (RX) sections of the baseband processor since it is common to both of them. In this case, a transmission with 54 Mbps was used for the power estimations. The Prime Power based estimation of power consumption of the entire chip is 393 mW, of which the power consumption in transmit and receive direction are 104 and 146 mW respectively. The measurements of the fabricated chip showed that in the realistic scenario with transmission and reception of 100-B frames at 54 Mbps, the average dynamic power consumption of the processor is 332 mW which matches with the Prime Power prediction. The chip area is 33 $mm^2$ including pads (core area only 19.5 $mm^2$), and the die photograph of the chip is shown in Fig. 14. Additionally, the overall digital processing latency is 8.36 $\mu$s in the receive mode for the 54 Mbps case, measured as the time from the last I/Q sample entering the baseband processor to the last data byte coming out from the Viterbi decoder. In the transmit mode this latency is only 0.35 $\mu$s.

We compare our architecture with three other architectures proposed earlier [18]–[20]. It is to be noted that because of the use of different technologies and implementation strategies adopted for the referenced designs it is extremely difficult to make a fair comparison with these designs. We provide the comparisons only to highlight the main contributions of this work.

The architecture proposed in [18] uses 0.25-$\mu$m 5-layer CMOS technology with 2.5 V core supply. The power consumptions of the core at 54 Mbps data rate are 219 mW and 203 mW for transmit and receive directions respectively, which are higher than the ones presented in this work.

The design proposed in [19] uses 0.18-$\mu$m 6-layer CMOS technology and 1.5-V core power supply. Like our design it uses 20/40/80 MHz clock frequencies. But the additional feature is that it integrates the MAC and PHY layers. The power consumption of the chip is 958 mW. It is not mentioned under which condition the power is measured. Also the breakdown of power consumption for MAC and PHY are not clearly stated making it difficult to compare with the digital core presented in this work.

The architecture proposed in [20] uses 0.25-$\mu$m technology and 2.5-V supply voltage. The power consumption is 109 mW which is lower than that of our design. Unfortunately once again it is not mentioned at which condition the power measurement is done. Also it does not include several of the key functional blocks which are included in our design like the scrambler, the convolutional encoder, and the interleaver in the transmit direction; and the deinterleaver, the descrambler, the residual phase corrector and the Viterbi decoder in the receive direction. Nevertheless, it includes a CFO tracking algorithm which is not necessary in our design. Another important difference of our design with the design in [20] is that although both the channel estimators are decision-directed the design in [20] makes a hard decision on the bits after the channel correction and this decision is fed back to the channel estimator which leads to much less data storage. Our design uses the output of Viterbi decoder as the input for the feedback loop. This improves the performance significantly but also increases the number of OFDM symbols to be stored which consequently increases the size of the required RAM.

## VI. Conclusion

In this paper, we have investigated the main issues related to the VLSI realization of the IRx as the most challenging component of an experimental baseband processing chip compatible with the IEEE 802.11a standard. Optimizing computationally intensive units like FFT, NCO etc., at the algorithm level for the reduction of arithmetic operations and then properly mapping these to architecture results in the overall power dissipation reducing significantly. Partitioning the design into different clock domains and adopting a distributed control strategy in conjunction with clock gating reduces the power consumption further. The power and area figures obtained confirm the energy efficiency of the solution when compared with other solutions reported in the literature.

## Acknowledgment

The authors would like to thank Prof. U. Ramacher and Dr. B. Gunzelmann for their unconditional support in publishing this paper. They also would like to thank Prof. A. Brown for his insightful comments and all the anonymous reviewers for their constructive criticism.

## References

[1] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: High Speed Physical Layer in the 5-GHz Band*, IEEE P802.11a/D7.0, 1999, Part II.

[2] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications. Amendment 4: Further Higher Data Rate in the 2.4-GHz Band*, IEEE P802.11G, 2003, Standard for IT—Telecommunications and information exchange between systems LAN/MAN—Part II.

[3] *WIGWAM Project (Wireless Gigabit With Advanced Multimedia Support)*, [Official website of the] [Online]. Available: http://www.wigwam-project.com

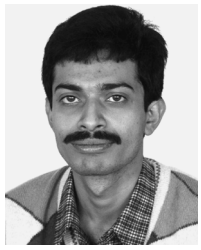
[4] *WiMedia Alliance*, [Official website of the] [Online]. Available: http://www.multibandofdm.org

[5] H. Meyr, M. Moeneclaey, and S. Fechtel, *Digital Communication Receivers: Synchronization, Channel Estimation, and Signal Processing*. New York: Wiley, 1998.

[6] A. Troya, "Synchronization and Channel Estimation in OFDM: Algorithms for Efficient Implementation of WLAN Systems," Ph.D. dissertation, Technical University of Brandenburg, Cottbus, Germany, 2005.
[7] A. Troya, K. Maharatna, M. Krstic, E. Grass, U. Jagdhold, and R. Kraemer, "Efficient inner-receiver design for OFDM-based WLAN system," *IEEE Trans. Wireless Commun.*, vol. 6, no. 4, pp. 1374–1385, 2006.
[8] M. Krstic, A. Troya, K. Maharatna, and E. Grass, "Optimized low-power synchronizer design for the IEEE 802.11a standard," in *Proc. IEEE ICASSP'03*, Apr. 2003, vol. II, pp. 333–336.
[9] K. Maharatna, A. Troya, S. Banerjee, and E. Grass, "Virtually scaling-free adaptive CORDIC rotator," *Proc. IEE Comput. Digit. Tech.*, vol. 151, no. 6, pp. 448–456, Nov. 2004.
[10] K. Maharatna, A. Troya, S. Banerjee, and E. Grass, "A CORDIC like processor for computation of arctangent and absolute magnitude of a vector," in *Proc. IEEE ISCAS'04*, Vancouver, Canada, May 2004, vol. II, pp. 713–716.
[11] K. Maharatna, S. Banerjee, E. Grass, M. Krstic, and A. Troya, "Modified virtually scaling free adaptive CORDIC rotator algorithm and architecture," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 11, pp. 1463–1474, Nov. 2005.
[12] K. Maharatna, E. Grass, and U. Jagdhold, "A 64-point Fourier transform chip for high-speed wireless LAN application using OFDM," *IEEE J. Solid-State Circuits*, vol. 39, no. 3, pp. 484–493, Mar. 2004.
[13] J. S. Walther, "A unified algorithm for elementary functions," in *Proc. Joint Spring Comput. Conf.*, Jul. 1971, vol. 38, pp. 379–385.
[14] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*. Englewood Cliffs, NJ: Prentice Hall, 1993.
[15] H. Johansson and L. Wanhammar, Design of bireciprocal linear-phase lattice wave digital Filters Linköping, Sweden, Rep. LiTH-ISY-R-1877, 1996.
[16] A. Krukowski and I. Kale, "Polyphase filter design with reduced phase nonlinearity," in *Proc. 5th WSES/IEEE World Multiconf. Circuits, Syst., Commun. Comput. CSCC'01*, Rethymnon, Crete, Greece, Jul. 2001.
[17] S. S. Lawson, "On design techniques for approximately linear phase recursive digital filters," in *Proc. IEEE ISCAS'97*, Jun. 1997, pp. 2212–2215.
[18] J. Thomson *et al.*, "An integrated 802.11a baseband and MAC processor," in *Dig. Tech. Papers IEEE ISSCC 2002*, Feb. 2002, vol. 1, pp. 126–127.
[19] T. Fujisawa *et al.*, "A single-chip 802.11a MAC/PHY with a 32-b RISC processor," *IEEE J. Solid-State Circuits*, vol. 38, no. 11, pp. 2001–2009, Nov. 2003.
[20] W. H. Tseng, C. C. Chang, and C. K. Wang, "Digital VLSI OFDM transceiver architecture for wireless SoC design," in *Proc. IEEE ISCAS'05*, 2005, pp. 5794–5797.

**Alfonso Troya** (M'95) was born in Barcelona, Spain, in 1975. He received the M.Sc. degree in telecommunications engineering from the Technical University of Catalonia, Barcelona, Spain, and the Dr.-Ing. degree from Brandenburg University of Technology, Cottbus, Germany, in 1999 and 2004, respectively.

He joined the Institute for High Performance Microelectronics (IHP), Frankfurt (Oder), Germany, in 1999, as a Research Associate in the Wireless Communication Systems Department, where he worked on the development and implementation of digital signal processing algorithms for broadband wireless communication systems. In October 2004 he joined Infineon Technologies AG, Munich, Germany, as an Algorithm Concept Engineer, where he was involved in the development of orthogonal frequency-division multiplexing based communication systems and their implementation on Software-defined radio architectures. In April 2007 he joined the European Patent Office, Rijswijk, The Netherlands, as a Patent Examiner in the Audio, Video and Media Directorate.



**Koushik Maharatna** (M'02) received the M.Sc. degree in electronic science from Calcutta University, Calcutta, India, in 1995 and the Ph.D. degree from Jadavpur University, Calcutta, India, in 2002.

From 1996 to 2000, he was involved in projects sponsored by the Government of India undertaken at the Indian Institute of Technology (IIT), Kharagpur, India. From 2000 to 2003, he was a Research Scientist with Institute for High Performance Microelectronics (IHP), Frankfurt (Oder), Germany. During this phase, his main involvement was related to the design of a single-chip modem for the IEEE 802.11a standard. In September 2006, he joined the Electronics System Design Group, School of Electronics and Computer Science of the University of Southampton, Southampton, U.K., where he is currently a Senior Lecturer. His research interests include development of VLSI architectures for the application in digital signal processing and communication, computer arithmetic, low-power digital circuit design, analog signal processing, and cellular neural networks.

**Miloš Krstić** was born in Niš, Serbia, in 1973. He received the Dipl.-Ing. and the M.Sc. degrees in electronics from the University of Niš, Niš, Serbia, in 1997 and 2001, respectively, and the Dr.-Ing. degree in electronics from Brandenburg University of Technology, Cottbus, Germany, in 2006.

Since 2001, he has been with the Institute for High Performance Microelectronics (IHP), Frankfurt (Oder), Germany, as a Research Associate within the Wireless Communication Systems Department, where he is currently working on low-power digital design techniques for wireless applications and globally asynchronous locally synchronous (GALS) methodologies for digital systems integration.

**Eckhard Grass** received the Dr.-Ing. degree in electronics from the Humboldt University, Berlin, Germany, in 1992.

He worked as a Visiting Research Fellow at Loughborough University, Loughborough, U.K., from 1993 to 1995 and as a Senior Lecturer in Microelectronics at the University of Westminster, Westminster, London, U.K., from 1995 to 1999. He has been with the Institute for High Performance Microelectronics (IHP), Frankfurt (Oder), Germany, since 1999, where he currently leads a project on the development and implementation of a broadband wireless communication system in the 60-GHz band. His research interests include data-driven (asynchronous) signal processing structures and low-power VLSI implementation of communication systems.

**Ulrich Jagdhold** received the Diploma in physics (M.Sc. degree) from the Technical University of Dresden, Dresden, Germany, in 1987.

From 1987 to 1996, he was with the Technology Integration Group, Institute for High Performance Microelectronics (IHP), Frankfurt (Oder), Germany, working on CMOS, BiCMOS, and SiGe technologies and device physics. In 1997, he joined the Wireless Communication Systems Department of IHP, where he has been working on wireless local area network system development projects, focusing on baseband integration issues and application-specific integrated circuits design, including development of digital CMOS libraries.

**Rolf Kraemer** (M'79) received his Diploma and Ph.D. degrees in electrical engineering and computer science from the Rheinsch-Westfälische Technische Hochschule (RWTH) Aachen, Aachen, Germany, in 1979 and 1985, respectively.

He has worked for 15 years in R&D of communication and multimedia systems at Philips Research Laboratories in Hamburg and Aachen. Since 1998 he has been s a Professor of Systems at the Brandenburg University of Technology, Cottbus, Germany. He also leads the Wireless Communication Systems Department of the Institute for High Performance Microelectronics (IHP), where his research focus is on wireless Internet systems spanning from application down to system-on-chip. He is co-founder of the startup company *lesswire AG*, where he holds the position of the Chief Technology Officer. He has published over 150 conference and journal papers, and holds 16 international patents.

Prof. Kraemer is a member of the IEEE Computer Society, the VDE-NTG, and the German Informatics Society.