Symbol Synchronization for SDR Using a Polyphase Filterbank Based on an FPGA

Pavel FIALA, Richard LINHART

Dept. of Applied Electronics and Telecom., University of West Bohemia, Univerzitní 26, 306 14 Plzeň, Czech Republic

pavelf@kae.zcu.cz, rlinhart@kae.zcu.cz

Abstract. *This paper is devoted to the proposal of a highly efficient symbol synchronization subsystem for Software Defined Radio. The proposed feedback phase-locked loop timing synchronizer is suitable for parallel implementation on an FPGA. The polyphase FIR filter simultaneously performs matched-filtering and arbitrary interpolation between acquired samples. Determination of the proper sampling instant is achieved by selecting a suitable polyphase filterbank using a derived index. This index is determined based on the output either the Zero-Crossing or Gardner Timing Error Detector. The paper will extensively focus on simulation of the proposed synchronization system. On the basis of this simulation, a complete, fully pipelined VHDL description model is created. This model is composed of a fully parallel polyphase filterbank based on distributed arithmetic, timing error detector and interpolation control block. Finally, RTL synthesis on an Altera Cyclone IV FPGA is presented and resource utilization in comparison with a conventional model is analyzed.*

Keywords

Digital communication, digital filters, FPGA, signal processing, synchronization, VHDL

1. Introduction

The increasing popularity of Software Defined Radio (SDR) in recent years is forcing complex digital signal processing blocks to be implemented in parallel design flow on an FPGA or ASIC. FPGA technology has undergone rapid evolution in the last decade and low-end gate arrays like the Altera Cyclone IV are now available for Digital Signal Processing (DSP) at a reasonable price. Symbol synchronization and carrier synchronization are crucial digital receiver blocks. Symbol synchronization is the method of estimating a clock signal, aligned in phase and frequency with the clock signal generated at the transmitter side [1]. The clock must be extracted from the received (noisy) signal because it is not efficient at allocating the spectrum for next the channel, which carries clock information only. Synchronization can be performed using polyphase filter banks, which allow greater flexibility and efficiency in the receiver by computing only those multiplications necessary for matched filtering, while simultaneously interpolating in order to achieve a sample point sufficiently close to the optimum [2].

The application of the polyphase filter to perform the desired interpolation required for symbol synchronization in general was presented in [1], using the Maximum Likelihood (ML) approach. The ML timing error detector requires the Matched Filter and the Derivative Matched Filter to operate correctly. This approach was further expanded upon by the authors of [2], [3] and [4]. In our work we utilize a Zero-Crossing or Gardner Timing Error Detector, which do not need a derivative matched filter. We also pay particular attention to the polyphase FIR filterbank because this structure has a significant impact on performance of the overall system. The new architecture presented in this paper can be thought of as the significant evolution of symbol synchronization for high-performance digital receivers and demonstrates the deployment of distributed arithmetic. This method is well suited for fixed-point implementation within FPGAs and ASICs.

This paper is organized as follows. In Sec. 2 we discuss the symbol timing techniques and definitions assumed in this paper, respectively. In Sec. 3 we describe each part of the proposed synchronization system and VHDL description is also provided. In Sec. 4 we present experimental results obtained from simulations. The second part of this section deals with the RTL synthesis, the usage of the system resources are discussed. Section 5 is devoted to conclusions.

2. Symbol Timing Recovery

The symbol synchronization loop can generally accommodate either a feed-forward or feedback structure. The first variant is frequently used in packetized burst communication and relies on a sequence of known symbols. Additional classification can be carried out according to their treatment of the following two data categories: Data-Aided and Non-Data-Aided (NDA) methods [5]. The Data-Aided method assumes that transmitted symbols (a training sequence) are known. Decision-Directed (DD) technique, a special subcategory of Data-Aided, is a technique where decoded symbols are used for decision making. This paper further focuses on feedback (digital) Phase-

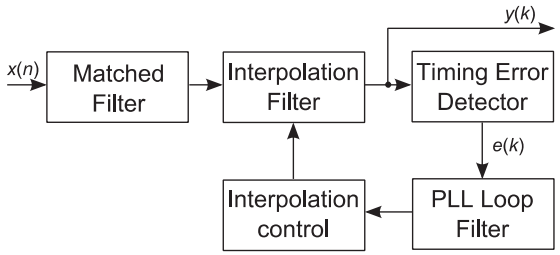

Fig. 1. Block diagram of a discrete feedback PLL timing loop.

locked loop (PLL) NDA/DD synchronization methods for the software defined m-QAM receiver. Figure 1 shows the conventional feedback PLL DSP approach to symbol synchronization.

The samples labeled $x(n)$ are initially processed in the Matched Filter (MF) and the interpolation filter calculates samples $y(k)$ at the desired positions from the offset samples provided by the free-running ADC. These position corrected samples are used by the timing error detector to form the timing error signal $e(k)$. The interpolation control section estimates the coefficients (namely, the fractional interval μ_k [6]) of the interpolation filter via the filtered error signal.

The timing error detector plays an important role in the synchronization system because the overall architecture of this timing loop depends on it. We have chosen to use a Zero-Crossing Timing Error Detector (ZCTED). The ZCTED is based on determining zero crossing in the eye diagram. It operates using 2 samples / symbol and produces zero error when every other sample is time-aligned with zero crossing in the matched filter output [6]. We assume that the matched filter produces outputs at a rate of 2 samples / symbol and are indexed by the symbol index k:

$$\ldots, x\left((k-1)T_s-\tau\right), x\left((k-1/2)T_s-\tau\right), x\left(kT_s-\tau\right), \\ x\left((k+1/2)T_s-\tau\right), x\left((k+1)T_s\right), \ldots \quad (1)$$

The timing error signal $e(k)$ can then be expressed as

$$e_{ZCTED}(k) = x\left((k-1/2)T_s+\hat{\tau}\right)\left[a(k-1)-a(k)\right] \quad (2)$$

where T_s is the sampling period (the sample clock is independent of the data clock used by the transmitter); τ is the timing offset; $\hat{\tau}$ is the timing delay estimate for the detector and symbol decisions a for the DD detector are

$$a(k-1) = \operatorname{sgn}\left[x\left((k-1)T_s+\hat{\tau}\right)\right], \\ a(k) = \operatorname{sgn}\left[x\left(kT_s+\hat{\tau}\right)\right]. \quad (3)$$

Special modification of the ZCTED is called Gardner Timing Error Detector (GTED). This exclusively NDA detector is independent of any carrier phase rotation and can be placed before the carrier synchronization subsystem [6]. It also operates using 2 samples / symbol where the error signal $e(k)$ is

$$e_{GTED}(k) = x\left((k-1/2)T_s+\hat{\tau}\right) \\ \left[x\left((k-1)T_s+\hat{\tau}\right)-x\left(kT_s+\hat{\tau}\right)\right]. \quad (4)$$

Polyphase filter decomposition is useful when implementing decimation or interpolation. This approach can be applied to symbol synchronization using an upsample filter performed solely by a polyphase filterbank (or else by a polyphase interpolator). The discrete impulse response of the m-th filter in the bank of M filters is given by [2]

$$h_m(mT_s) = h\left(nT_s + \frac{m}{M}T_s\right). \quad (5)$$

All of these sub-filters in the polyphase filterbank have the same magnitude transfer function, but they are separated by a sample delay, which generates a phase offset. The synchronization timing loop must select the appropriate sub-filter at index m such that the fractional portion m/M of the sampling period T_s is close to the desired timing offset. We use a finite number of filters in the filterbank, which means that there are only a finite numbers of interpolants. The optimum sampling point is never reached, but with the appropriate filterbank size, fine resolution can be achieved. This is clearly shown in Fig. 2 where $M = 8$. A conventional interpolate filter usually utilizes a linear, quadratic, or cubic polynomial interpolator in a very similar way to how the optimum sampling instant is retrieved.

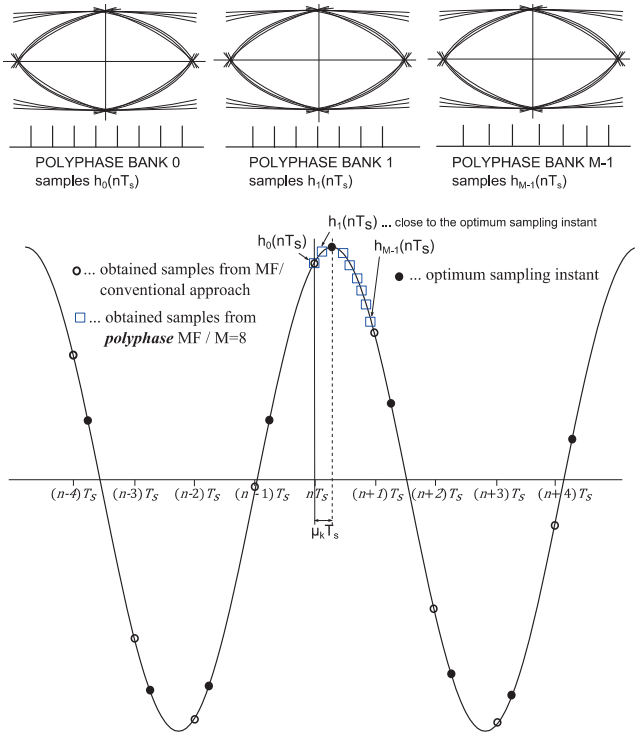

Fig. 2. Basic principle of the polyphase interpolator for symbol synchronization and the issue with optimum sampling instant determination

3. System Model Design

Based on the above considerations, we have designed a synchronization system model incorporating a polyphase FIR interpolator. This timing loop shown in Fig. 3 consists of the mentioned polyphase FIR filterbank, the ZCTED (GTED) detector, the PLL loop filter and the interpolation control block.

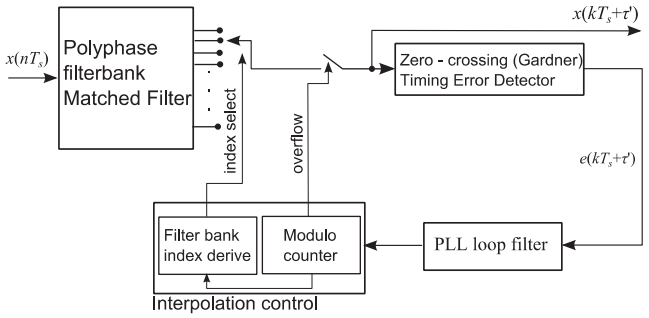

Fig. 3. A block diagram of the proposed synchronization system using polyphase filterbank.

3.1 Polyphase Filterbank Matched Filter

Distributed arithmetic technique is used for the proposed polyphase filterbank matched FIR filter. Distributed arithmetic (DA) generally represents a family of algorithms that use Look-Up Tables to calculate a sum of products or multiply accumulate (MAC) products [7]. Linear convolution can be formulated as a sum of products:

$$y=\sum_{n=0}^{N-1}c[n]\times x[n] \\ =c[0]x[0]+c[1]x[1]+..+c[N-1]x[N-1]. \tag{6}$$

If the filter coefficients $c[n]$ are fixed, then the partial MAC product term ($c[n]\times x[n]$) is multiplied with the constant. This is a crucial assumption of DA technique. In an unsigned DA system, the variable $x[n]$ can be expressed as:

$$x[n]=\sum_{b=0}^{B-1}x_b[n]\times 2^b \text{ where } x_b[n]\in\{0,1\} \tag{7}$$

where x_b signifies the b^{th} bit of $x[n]$. The final product y can be then expressed as

$$y=\sum_{n=0}^{N-1}c[n]\times\sum_{b=0}^{B-1}x_b[n]\times 2^b\,. \tag{8}$$

Redistributing the order of summation in term (8) and providing the extension to work with the two's complement signed numbers lead to the following final expression:

$$y=-2^B\times\underbrace{\left(c[n]\times x_b[n]\right)}_{\text{MSBbit}}+\sum_{b=0}^{B-1}2^b\times\sum_{n=0}^{N-1}\left(c[n]\times x_b[n]\right). \tag{9}$$

This extension is possible because the MSB bit in two's complement distinguishes between positive and negative numbers. Implementation of mapping ($c[n]\times x[n]$) in (9) warrants special attention. The FPGA can perform this mapping using the LUT table. Alternatively, integrated RAM memory blocks can be used [8]. This means that the 2^N LUT table is preprogrammed to accept the N-bit input vector $x_b=(x_b[0],x_b[1],x_b[2],\ldots x_b[N-1])$, where the output is $f(c[n]\times x[n])$. Individual mappings are then weighted by the corresponding power-of-two factor and accumulated. The resulting structure does not utilize integrated multipliers (only a logic shift operation is required), which can be preferably used in specialized DSP blocks of the system. At

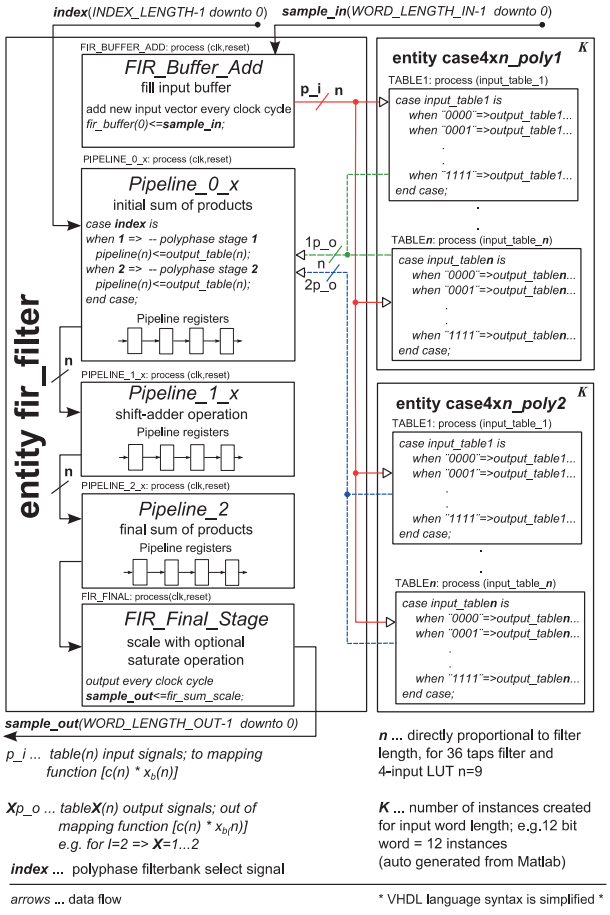

Fig. 4. The VHDL structure of a fully parallel DA FIR polyphase interpolator for symbol synchronization (the interpolation factor is only $I=2$ for illustration).

first glance, this concept is only suitable for short filters with length L because of the limited addressable memory space in one LUT table (e.g. $L\leq 4$ for a 2^4 x 1 - bit LUT table). Because FIR filters belong to the class of linear filters, the combination of N lower order filters can create the desired FIR filter of the higher order.

The structure of VHDL implementation of the proposed, fully parallel, signed FIR matched filter (with a polyphase filterbank) based on DA is shown in Fig. 4. This type of filter calculates output in a single clock cycle and is specifically adapted for symbol synchronization purposes. The pipeline registers are utilized to the maximum extent possible.

The general purpose interpolation filter may read new input samples every I clock cycle and outputs are produced every clock cycle. Only one clock signal is then found in the filter entity. This clock signal is I-times faster than the clock signal, which drives original sampled input signal. The necessary modifications for symbol synchronization purposes are evident from Fig. 2 and 3 and can be summarized as follows:

- The most recent polyphase sub-filter is selected by an *index* signal generated by the *interpolation control* block. This signal corresponds to the address for the multiplexer.

- There is no need for the internal modulo *I* counter as new samples are read every clock cycle. Only one sub-filter with the desired phase offset over time is required. This sub-filter is selected by an *index* signal.

The VHDL entity *fir_filter* is the main unit of this model and represents a template, which is only slightly modified for different coefficient filter sets. The input vector *sample_in* is saved to the buffer by length, which is equal to the number of sub-filter taps. Every clock cycle, a new vector is obtained and older vectors are shifted. The process *FIR_Buffer_Add* has a basic shift register function. Mapping $f(c[n]\times x[n])$ is then concurrently performed using instances of the *case4x**n**_poly_**I*** entity, where *n* is directly proportional to the filter taps and *I* is the interpolation factor. We can also assume that the 2^4 x 1 - bit LUT tables are used. We have labeled these entities case***4***… for that reason. For every sub-filter in the bank, an independent *case4x**n**_poly_**I*** entity (e.g. $I = 8 \rightarrow 8$ sub-filters $\rightarrow$ 8 entities) must be created. For fully parallel operation, we must create *k* instances of this entity, which means that for every bit of input word one instance is required.

The Process *PIPELINE_0_x* uses a multiplexer with an address controlled by the *index* signal generated by the interpolation control block, in order to select the appropriate sub-filter and to perform the initial calculation of the sum of the products. The lowercase letter *x* in the title indicates the VHDL process *id* in the same pipeline section. The section *PIPELINE_1_x* produces shift-adder operation; embedded multipliers are not utilized. General purpose multipliers are saved for other DSP subsystems. The process *PIPELINE_2* finalizes the computation of the products and the *FIR_Final_Stage* implements scale (with optional saturate) operation. The entities *case4x**n**_poly_**I*** are automatically generated from Matlab. The pipeline section *PIPELINE_0_x* can produce very comprehensive code and is also automatically generated. The appropriate VHDL code with signal definitions is then added to the *fir_filter* main entity (details in Sec. 4.2). The developed structure can be further examined from the gate-level point of view in Fig. 6.

Instead of LUT tables, the previous structure can utilize integrated RAM blocks configured as single-port ROM memory for the coefficients store. This extension has been tested with Altera M9K integrated RAM blocks using the *Altsyncram Megafunction* [9]. The disadvantage of this option is that we have to count one extra mandatory clock cycle. On the other hand, it saves valuable logical blocks for other DSP subsystems.

3.2 Timing Error Detector and PLL Loop Filter

The block implementation of the ZCTED was created under term (2). For the GTED, we used a slightly modified term (4), where the sign(.) function is missing. The formed error signal $e(k)$ is then filtered by the second order IIR filter (proportional-plus-integrator). The constant of proportionality K_p (detector gain) must be calculated for this synchronization system to operate properly. This constant should be obtained from what is known as the *S*-curve of the ZCTED (GTED). This is generally defined as the expectation of the adjustment signal $e(k)$, conditioned on a fixed value of the timing error [10]. The *S-curve* of the ZCTED is an estimate of the slope $r_a(-\tau_e)$ using values of $r_a(t)$ one-half a symbol time before and after $-\tau_e$, where r_a is the autocorrelation function of the pulse shape and τ_e is the timing error expressed as:

$$\tau_e = \tau - \hat{\tau}. \tag{10}$$

The square-root raised cosine (SRRC) pulse shape with 50% excess bandwidth is used in further simulations. Raised cosine filtering is divided between the transmitter and the receiver employing the same SRRC filter on both sides, which results in minimum inter-symbol interference (ISI). Experimentally, the *S-curve* is measured by opening the recursive timing loop and measuring the average of adjustment signal $e(k)$. The *S-curve* denoted as $S(\tau_e)$ can be thus formulated as:

$$S(\tau_e) = E\{e(k)|\tau_e\}. \tag{11}$$

The constant K_p is the slope of $S(\tau_e)$ at $\tau_e = 0$. Simulation of ZCTED *S-curves* for QPSK with various signal-to-noise ratio (SNR) values was performed using Matlab and the results are shown in Fig. 5.

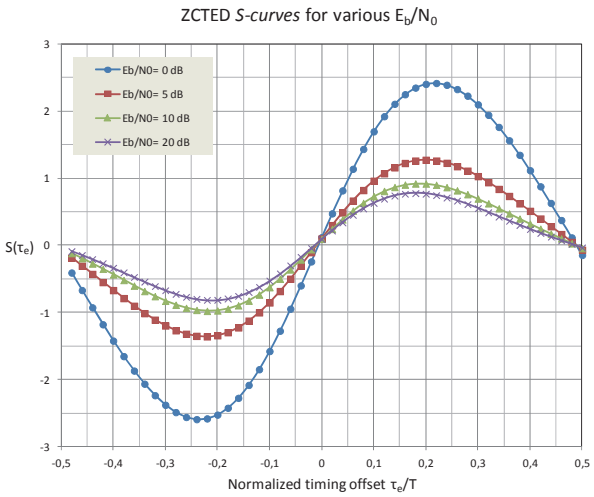

Fig. 5. S-curves for decision directed ZCTED; using the SRRC pulse with 50% excess bandwidth and unit average energy.

The block diagram of the *S-curve* measurement is illustrated for clarification in Fig. 7. We also show the table with recorded K_p values. GTED *S-curve* is determined in a similar fashion [10].

3.3 Interpolation Control Block

Interpolation is controlled by a *modulo-1* counter. This counter can be implemented in principle as an incrementing or decrementing counter. The incrementing counter is incremented by $1/N$ (quantized for fixed-point operation) on average. Because the ZCTED and GTED detectors

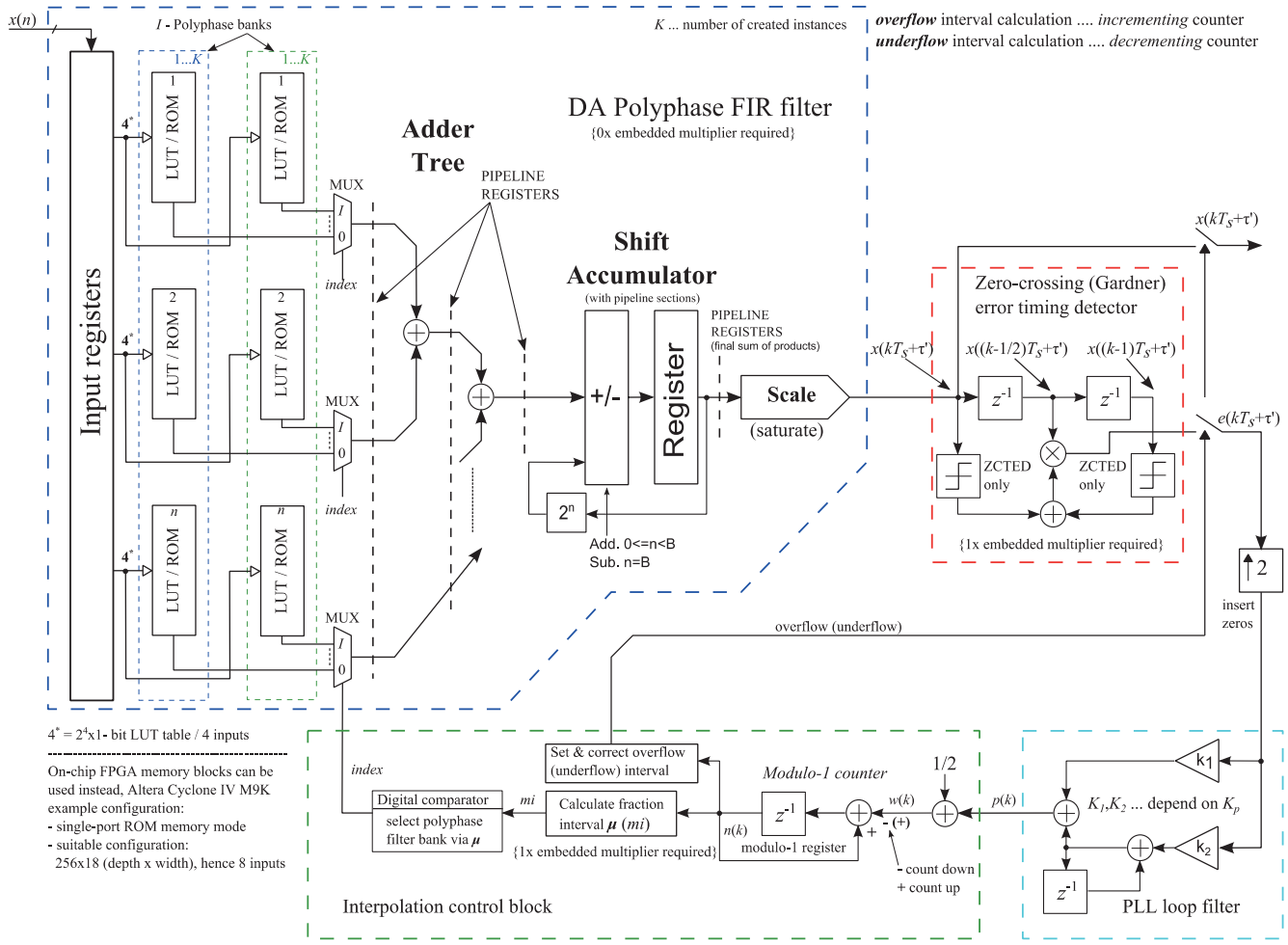

Fig. 6. Detailed structure of the proposed fixed-point, symbol synchronization system. This system consists of a DA polyphase FIR filter, a Zero-Crossing (Gardner) Timing Error Detector, a PLL loop filter and an Interpolation control block.

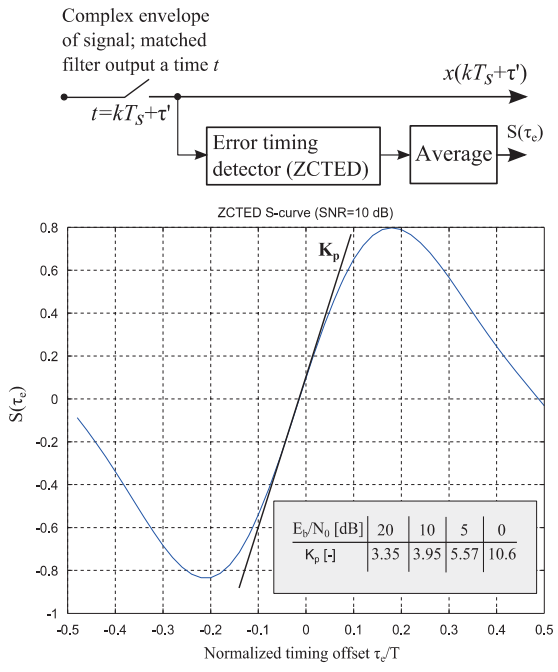

Fig. 7. *S-curve* measurement and obtained K_p constants.

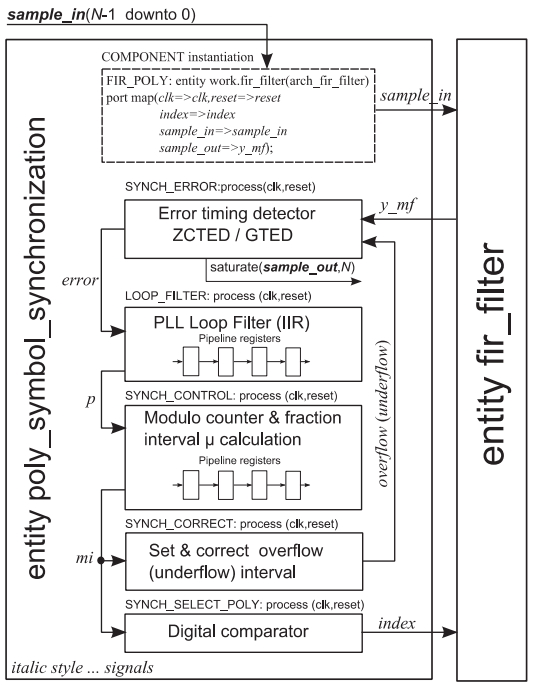

Fig. 8. VHDL structure of the overall synchronization system.

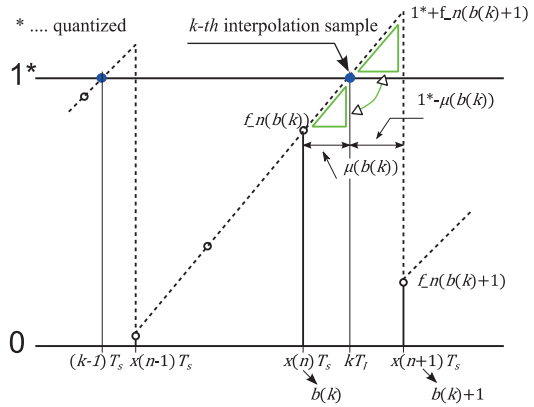

Fig. 9. Calculation of the fractional interval μ using the incrementing *modulo-1* counter.

are intended to operate using 2 samples / symbol, N is equal to 2. The counter overflows each N samples and the PLL loop filter output $p(k)$ (see Fig. 6) in order to adjust the amount by which the counter increments. The fractional interval μ is calculated for every overflow period; the method for determining this interval is illustrated in Fig. 9.

We denote desired sample at time kT_I as the k-th interpolation sample. If the k-th interpolation sample is between samples $x(nT_s)$ and $x(n + 1)T_s$, the sample with index n is called the k-th base point index and is here denoted as $b(k)$. The k-th fractional interval $\mu(b(k))$ is related to the base point index $b(k)$, which is a certain fraction of the sampling period T_s. The size of this interval must be expressed using signals from Fig. 6. When incrementing counter overflows

$$n\big(b(k)+1\big) = n\big(b(k)\big) + w\big(b(k)\big) - 1. \tag{12}$$

As shown in Fig. 9, the values $n(b(k) + 1)$ and $n(b(k))$ create similar triangles. This leads to the relationship

$$\frac{\mu\big(b(k)\big)}{1-n\big(b(k)\big)} = \frac{1-\mu\big(b(k)\big)}{n\big(b(k)+1\big)}. \tag{13}$$

Finally $\mu(b(k))$ is expressed as

$$\mu\big(b(k)\big) = \frac{1-n\big(b(k)\big)}{n\big(b(k)+1\big)-n\big(b(k)\big)+1} = \frac{1-n\big(b(k)\big)}{w\big(b(k)\big)}. \tag{14}$$

General division in (14) has to be eliminated for fast fixed-point implementation and optimal VHDL synthesis on the FPGA. We can recognize that $1/N \approx T_I/T_s$ [11]. Although the exact T_I/T_s is unknown and irrational, the nominal value ξ_0 expressed to finite precision, can often be an excellent approximation of the true value [12]. The fractional interval μ can then be approximated as follows

$$\mu\big(b(k)\big) \approx \xi_0\big[1-\eta(b(k)\big] \quad \text{where } \xi_0 = N = 2 \tag{15}$$

If the deviation ξ_0 is too large, a first order correction should then be used; the standard deviation in μ to $\Delta\xi^2/\xi_0^2\sqrt{12}$ is reduced, again without requiring division [11]. The decrementing counter works in a similar fashion, where the counter underflows every N samples.

The correction of the overflow (underflow) interval is necessary because the real ADC never provides exact samples using 2 samples / symbol (for example). The real sampling period T_r is always of a smaller or higher interval than the desired sampling period T_s. The interval is any fraction of the sampling interval T_s. The correction must be applied for this timing loop to operate properly. We can summarize the correction process in cases where 2 samples/symbol are used:

- If the sample rate T_r is faster, i.e. $T_r > T_s/2$, then a sample with zero value is inserted to avoid two consecutive overflow intervals.
- If the sample rate T_r is slower, i.e. $T_r < T_s/2$, then a sample is skipped to avoid the overflow interval after two samples instead of one.

The digital comparator finally selects a polyphase filter bank based on the fractional interval μ.

4. Simulations and FPGA Synthesis

4.1 Matlab Simulation

Closed-loop simulation is performed for the modified timing loop with a polyphase filterbank and forms an important block of the proposed QPSK transmission simulation model. This model consists of a QPSK signal generator, a channel with white noise (AWGN channel), a digital down convertor with a free running oscillator, an automatic gain control loop (AGC), the proposed symbol timing loop and a carrier phase synchronization loop block. Figure 10 shows this simulation model including some sub-blocks not mentioned above. We will further explore, with a few exceptions, the symbol synchronization section with the following parameters only.

The matched filter has 8 phases of sub-filters with a roll-off factor of 0.5, where the filter order in symbols is N_{sym}= 6. In theory, the resulting 96 coefficients are partitioned into I = 8 polyphase sub-filters (phases), each having 12 coefficients in cases where 2 samples / symbol are used. Because the length of the SRRC matched filter impulse response is N_{sym}×*SamplesPerSymbol* + 1, each sub-filter has 13 coefficients. The timing error signal is filtered by a proportional-plus-integrator loop filter, which is updated once per symbol period. The proportional constants K_p are set to 3.95 for the ZCTED and to 1 for the GTED.

The simulations have confirmed that previous K_p values for E_b/N_0 = 10 dB can be used over a wide range of SNR values (refer to Fig. 10) with minimal impact on overall PLL performance. The PLL loop filter is constructed according to the following performance requirements:

- $B_{PLL}T_s$= 0.005, where B_{PLL} is the PLL bandwidth.
- The filter loop transfer function with the above values is utilized for calculating K_1 (0.00665), K_2 (2.2148e–05) constants, where the damping factor ζ = 1.

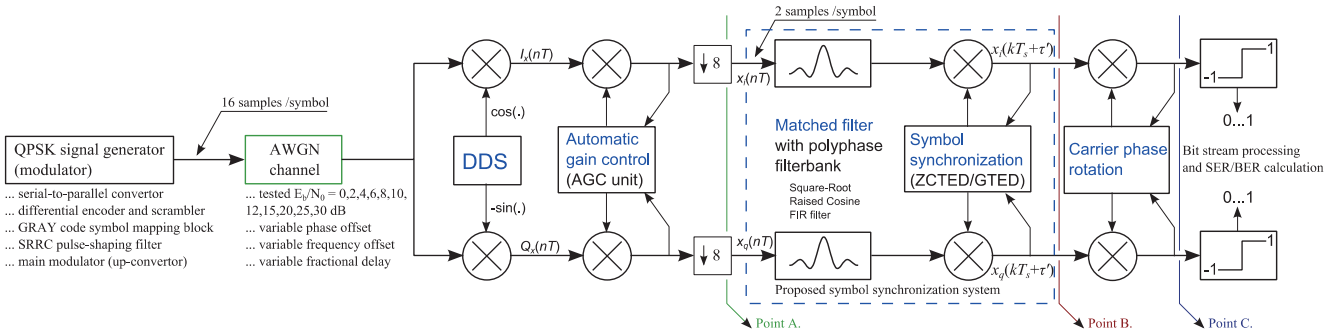

Fig. 10. Matlab simulation scheme for the QPSK receiver incorporating the proposed symbol synchronization system with a polyphase filterbank. The simulation consists of a QPSK modulator, an AWGN channel and a QPSK receiver.

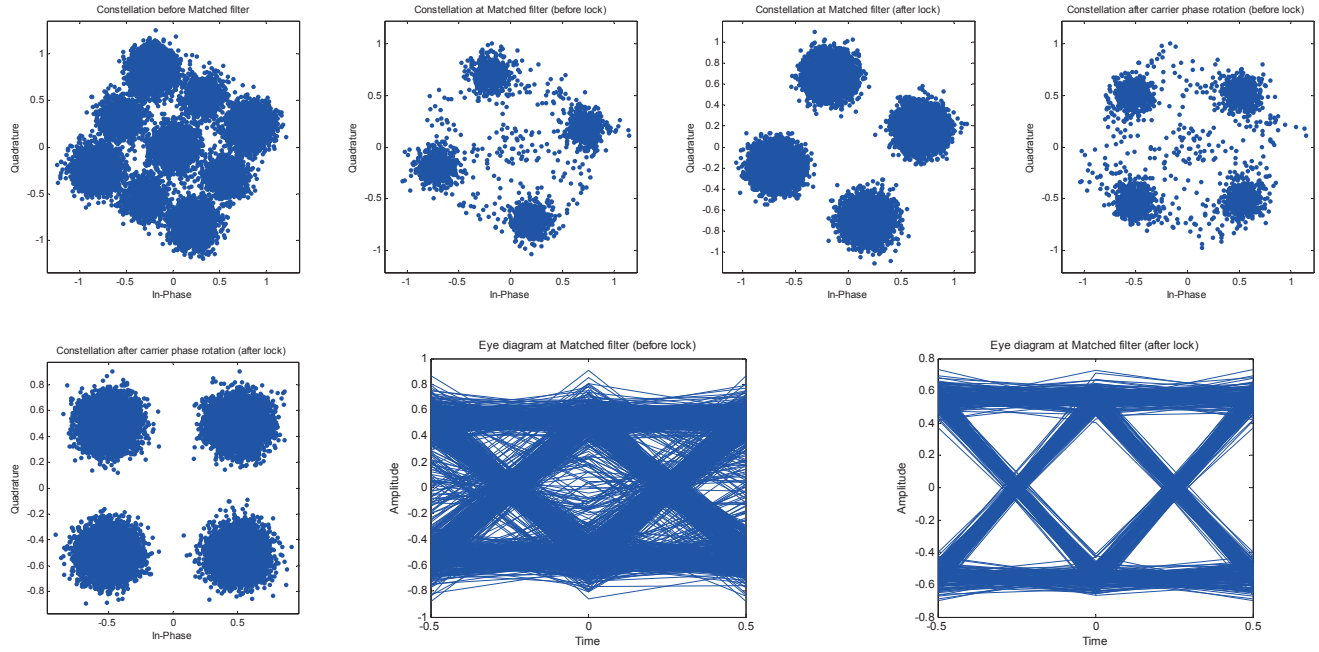

Fig. 11. Constellation and eye diagrams at the receiver side with $E_b/N_0 = 8$ dB and phase offset $\varphi = 30°$. GTED is used. Description from top down: Constellation before matched filter (point A.), constellation at matched filter – before lock for 50 – 3000 symbols (point B.), constellation at matched filter – after lock for 4000 – 20000 symbols, constellation after carrier phase synchronization – before lock for 50 – 3000 symbols (point C.), constellation after carrier phase synchronization – after lock for 4000 – 20000 symbols, eye diagram at matched filter – before lock, eye diagram at matched filter – after lock.

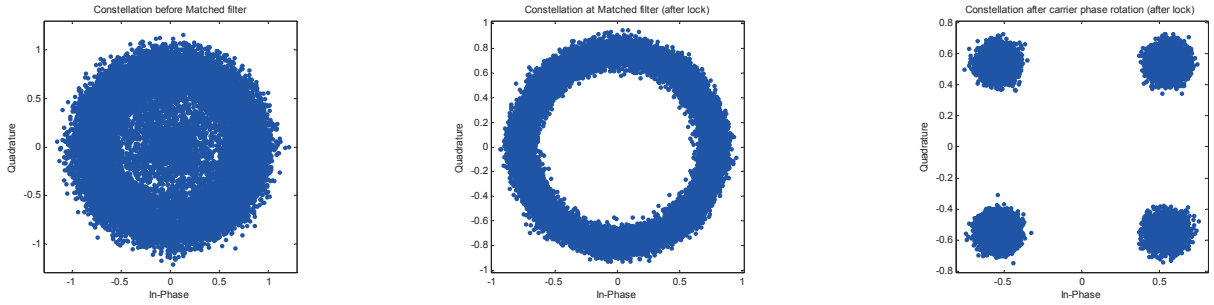

Fig. 12. Constellation at the receiver side with $E_b/N_0 = 12$ dB and phase offset $\varphi = 30°$. GTED is used. The sampling clock frequency is 1/4000 of a symbol rate faster than 2 samples / symbol. Description from left: Constellation before Matched filter, constellation at Matched filter – after lock, constellation after carrier phase synchronization – after lock (with equal number of samples as in Fig. 11).

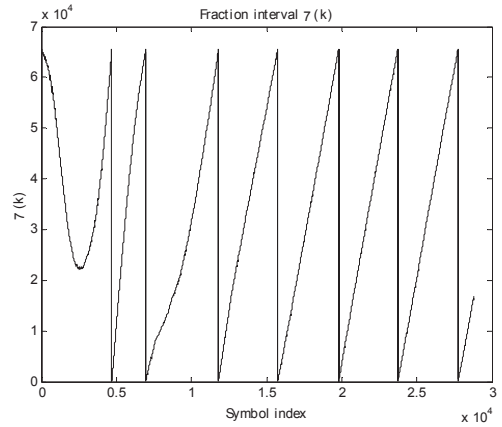

Fig. 13. Fractional interval $\mu(k)$ plot, where $E_b/N_0 = 8$ dB and phase offset $\varphi = 30°$ are shown, GTED is used.

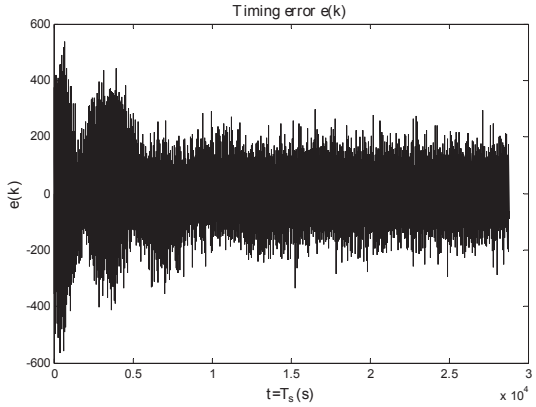

Fig. 14. Timing error $e(k)$ plot, where $E_b/N_0 = 8$ dB and phase offset $\varphi = 30°$are shown, GTED is used.

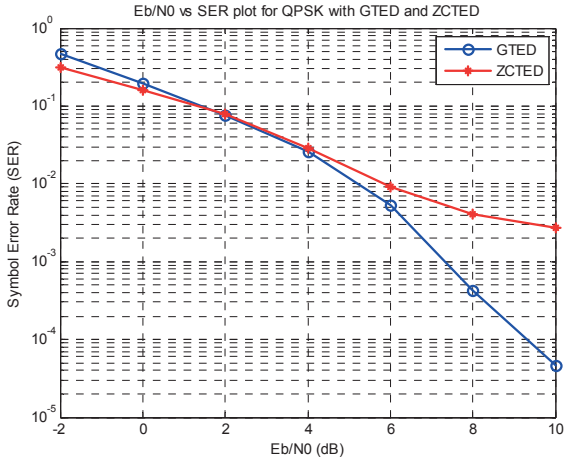

Fig. 15. Evaluation of SER for GTED and ZCTED, phase offset $\varphi = 40°$, after lock for 4000 – 20000 symbols, averaged over 50 realizations.

Because the correction of the phase rotation follows symbol synchronization, a rotationally-invariant timing error detector like the GTED is generally better for this simulation arrangement. Evaluation of the Symbol Error Rate (SER) was performed using the GTED and ZCTED. Simulations have shown that for a higher ratio of SNR (respectively for $E_b/N_0 > 4$ dB) the GTED outperforms the ZCTED. For $E_b/N_0 < 4$ dB performance of both detectors is similar (see Fig. 15). The figure for the fraction interval $\mu(k)$ in Fig. 13 is displayed for the condition, where the sampling clock frequency is 1/4000 of a symbol rate faster than 2 samples / symbol. The fractional interval then ramps from 0 to 1 and rolls over every 4000 symbols.

In addition to standard floating-point simulation, extended, fixed-point simulation was also carried out. This approach allows faster transition to the fixed-point VHDL model description. The fractional interval μ is then calculated by (15) instead of (14).

4.2 Modelsim RTL Simulation

The development cycle starts with the designed polyphase FIR filter with floating-point coefficients using Matlab. The developed *m-file* script performs actions described in Fig. 16.

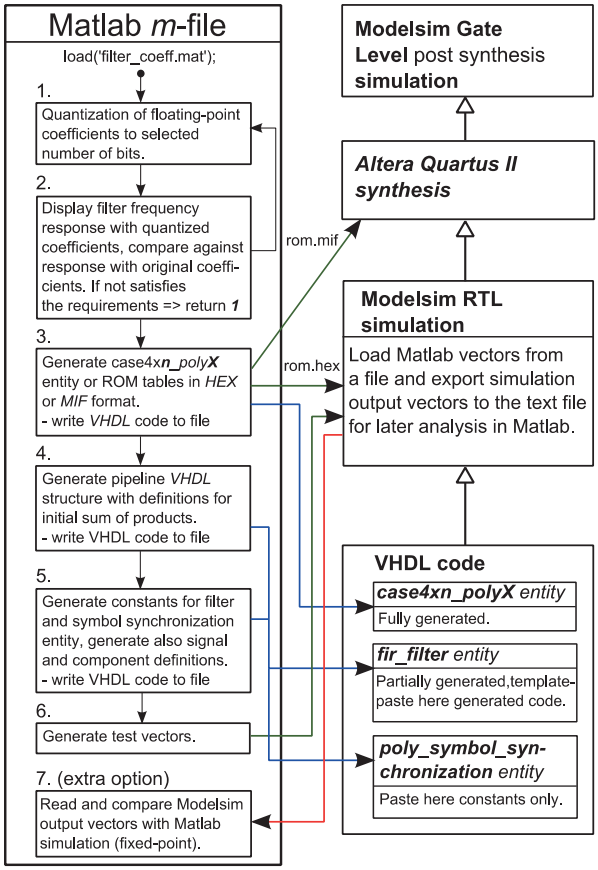

Fig. 16. Design development flow chart.

RTL simulation was performed using the Altera Modelsim program with a VHDL test-bench, which loads test data vectors from a text file. These fixed-point vectors were exported from Matlab. The test-bench allows the simulation output vectors to be exported back to the file for further analysis and compares fixed-point simulation using Matlab. The created *m-file* script also generates constants for the IIR loop filter in the symbol synchronization entity.

4.3 FPGA Synthesis

Synthesis was carried out using a low-cost Altera Cyclone IV FPGA (Cyclone IV EP4CE115F29C7) and Terasic DE2-115 development board with a high-speed AD/DA conversion card. Synthesis results are sorted for the proposed synchronization scheme in Tab. 1. The results above are given only for one branch of I/Q receiver. For I/Q operation, it is necessary to create two instances of the main entity (*poly_symbol_synchronization*).

Synthesis results: input data – **12 bit** constants – **16 bit** coefficients – **16 bit** *Polyphase matched SRRC filter* - 97 taps / 8 sub-filters / 13 taps each	**Cyclone IV EP4CE115F29C7 (114.480 LEs)** *Altera Quartus 12.0sp2* LEs / MEM bits	Combination functions	Logic registers	Embedded Multiplier (9-bit elements)
Polyphase synch. (LUT version) f_{MAX}=108.9 MHz	4967 (4%) / 0	4726 (4%)	1533 (1%)	2/532 (<1%)
Polyphase synch. (ROM version) f_{MAX}=110.2 MHz	2770 (2%) / 737.280 (19%)	2560 (4%)	1161 (2%)	2/532 (<1%)
Polyphase synch. (LUT, without DA) f_{MAX}=114.4 MHz	3805 (3%) /0	2997 (3%)	2485 (2%)	70/532 (13%)

Altera Quartus II main synthesis options: Fitter level – *standard*; Optimization technique – *balanced,* TimeQuest Timing Analyzer – *slow analysis*

Tab. 1. Synthesis results for the proposed synchronization system; the last row provides result without DA.

The test environment is essentially the same as for the simulation. An additional FPGA board with a DAC is used as the QPSK modulator. Transmission channel parameters are influenced by a noise generator and a programmable attenuator. *Altera Signal Tap* results conform to simulation. The last row of the table provides comparison of version with and without utilization of DA; a multiplier based fully parallel matched filter is used. This implementation significantly enlarges the demands on the number of multipliers (DSP blocks) needed on FPGA. This may be the serious issue during implementation on FPGA, which are found for example in lower cost SDR platforms like USRP N200 [13]. The implementation of some parallel challenging SDR blocks like symbol synchronization on an FPGA can significantly reduce the performance requirements on a personal computer. The analysis of the resource utilization by authors in [14] and [15] for their symbol timing system confirmed too that more than one quarter of dedicated multipliers is necessary for RTL synthesis on FPGA. The presented synchronization system was compared with a conventional synchronization system, where a Farrow structure [11] is used to implement the time-varying fractional delay, FIR filter. This conventional symbol timing system is given with synthesis results in the Appendix section (Fig. 1A). The authors of [16] and [17] have analyzed and implemented different forms of this conventional timing system. They did not incorporate DA technique. Our new timing loop design reduces design complexity by more than 20%. The proposed system using a polyphase filterbank saves high amounts of embedded multipliers. In addition, the conventional design loop delivers a larger transport delay through the cascade of two filters than the filterbank design's loop through the delay of a single filter.

5. Conclusion

In this paper, we present a combined, matched and interpolation filters. This filter uses a polyphase filterbank for symbol synchronization purposes for an SDR receiver, based on an FPGA. We introduce an original, fully parallel, polyphase filterbank structure especially adapted for symbol timing purposes based on DA. Applying distributive arithmetic to the symbol synchronization domain represents a new and innovative idea, since our knowledge this design has not been published before. This method can also dramatically improve timing loop performance. We present a comprehensive analysis of the Zero-crossing (Gardner) Timing Error Detector, whereas previous works have only focused on the ML approach using an additional derivative filterbank. Detailed simulations with fixed-point extensions were carried out and the results illustrate the response of the loop filter and timing control block. The reliability of the proposed symbol synchronization system has been verified using a complete, coherent QPSK SDR receiver. The structure was tested on a low-cost Altera Cyclone IV FPGA having utilization at 4% and a maximum operating clock speed of over 100 MHz. Only two embedded multipliers were utilized. The main design benefits are: optimal logic resource utilization, easy portability using standard libraries, and impressive performance. The size of the design can be further reduced using the polyphase filterbank as a serial structure but lowers performance of timing loop.

Appendix

In this Appendix, we illustrate our previously developed, conventional, symbol synchronization model for comparison purposes. The Farrow structure using a piecewise, parabolic or cubic interpolator is used to implement the time-varying fractional delay FIR filter. The Farrow structure is well suited for practical hardware implementations, but conventional approaches fully utilize general purpose multipliers. We have extended the classical Farrow structure using DA, where multipliers are used only in the last stage. The Farrow structure features a piecewise, parabolic interpolator and utilizes two FIR filters with different coefficients (three are needed for the cubic interpolator). A separate matched filter is required as this structure only performs arbitrary interpolation.

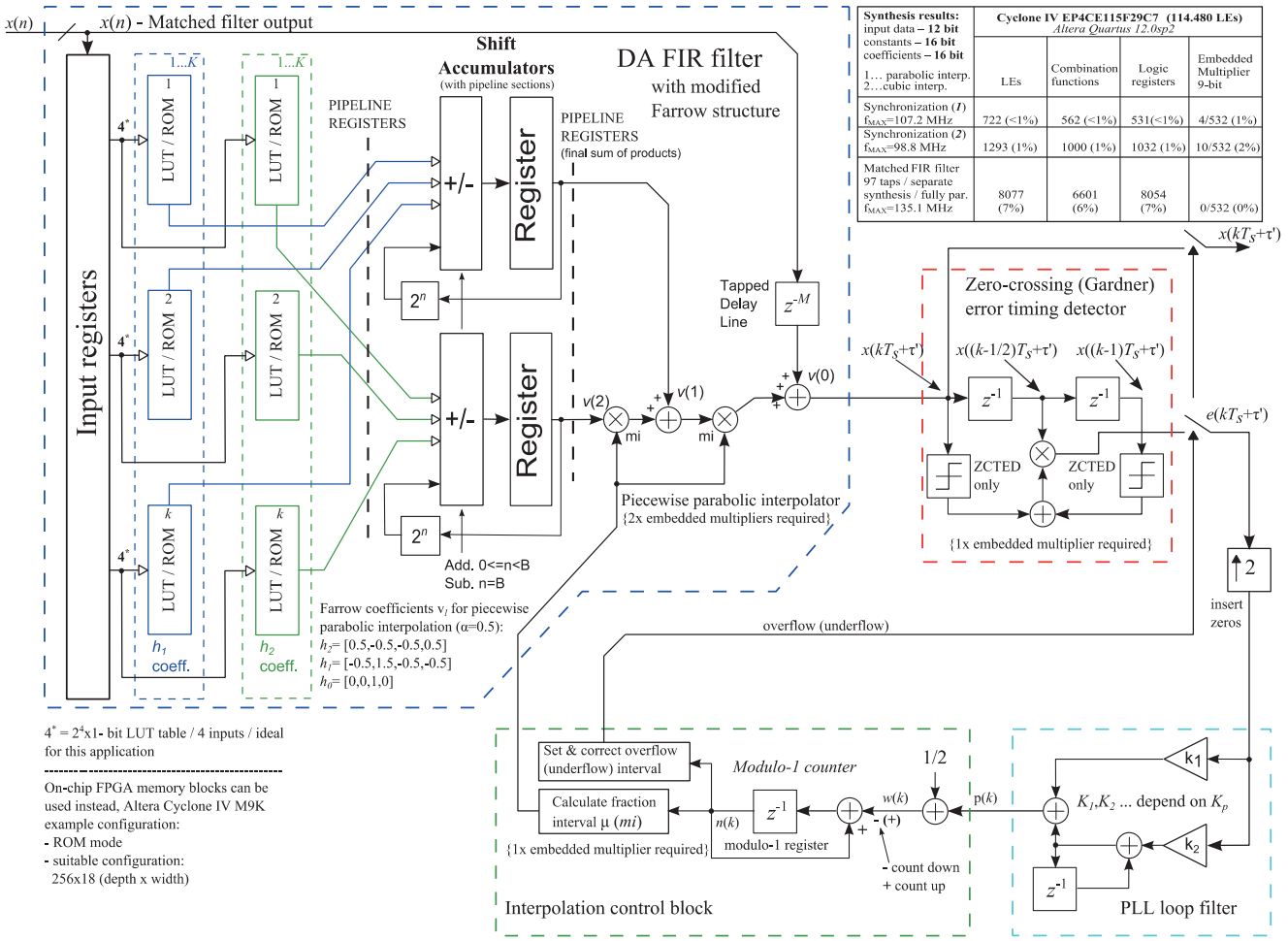

Fig. A1. Detailed structure of DA modified conventional symbol synchronization system with parabolic interpolator.

Acknowledgments

This project was supported by the Student Grant Scheme (SGS-2012-019) of the Faculty of Electrical Engineering, University of West Bohemia.

References

[1] DICK, CH., HARRIS, F. J., RICE, M. Synchronization in software radios - Carrier and timing recovery using FPGAs. In *IEEE Symposium on Field-Programmable Custom Computing Machines*. Napa Valley (CA, USA), 2000, p. 195–204. DOI: 10.1109/FPGA.2000.903406

[2] GAEDDERT, J., VOLOS, H. I., CORMIER D., REED, J. H. Multi-rate synchronization of digital receivers in software-defined radios. In *Proceeding of the SDR 07 Technical Conference and Product Exposition*. Denver (USA), 2007, p. 195–200.

[3] HARRIS, F. J., RICE, M. Multi-rate digital filters for symbol timing synchronization in software defined radios. *IEEE Journal on Selected Areas in Communications*, 2001, vol. 19, no. 12, p. 2346–2357. DOI: 10.1109/49.974601

[4] AWAN, M., KOCH, P. Combined matched filter and arbitrary interpolator for symbol timing synchronization in SDR receivers. In *IEEE 13th International Symposium on Design and Diagnostics of Electronic Circuits and Systems* (*DDECS*). Vienna (Austria), 2010, p. 153–156. DOI: 10.1109/DDECS.2010.5491797

[5] MENGALI, U., D'ANDREA A. N. *Synchronization Techniques for Digital Receivers*. New York: Plenum, 1997.

[6] RICE, M. *Digital Communications: A Discrete-Time Approach.* New York: Prentice Hall, 2009.

[7] MEYER-BAESE, U. *Digital Signal Processing with Field Programmable Gate Arrays*. Heidelberg: Springer, 2007.

[8] Altera Corporation, San Jose USA. *Internal Memory (RAM and ROM) User Guide* (datasheet). 64 pages. [Online] Cited 2015-03-08. Available at: www.altera.com/literature/ug/ug_ram_rom.pdf

[9] Altera Corporation, San Jose USA. *Memory Blocks in Cyclone IV Devices* (datasheet). 18 pages. [Online] Cited 2015-03-08. Available at: http://www.altera.com/literature/hb/cyclone-iv/cyiv-51003.pdf

[10] GARDNER, F. A BPSK/QPSK timing-error detector for sampled receivers. *IEEE Transactions on Communications*, 1986, vol. 34, no. 5, p. 423–429. DOI: 10.1109/TCOM.1986.1096561

[11] GARDNER, F. Interpolation in digital modems – Part I: Fundamentals. *IEEE Transactions on Communications*, 1993, vol. 41, no. 3, p. 501–507. DOI: 10.1109/26.221081

[12] ERUP, L., GARDNER, F., HARRIS, R. A. Interpolation in digital modems – Part II: Implementation and performance. *IEEE Transactions on Communications*, 1993, vol. 41, no. 6, p. 998–1008. DOI: 10.1109/26.231921

[13] Ettus Research, Santa Clara USA. *USRP N200 Networked Series Datasheet.* 2 pages. [Online] Cited 2015-03-08. Available at: http://www.ettus.com/content/files/07495_Ettus_N200-210_DS_Flyer _HR.pdf

[14] MONTAZERI, A., KIASALEH, K. Design and performance analysis of a low complexity digital clock recovery algorithm for software defined radio applications. *IEEE Transactions on Consumer Electronics*, 2010, vol. 56, no. 3, p. 1258–1263. DOI: 10.1109/TCE.2010.5606256

[15] LIN, CH., ZHANG, J., SHAO, B. A high speed parallel timing recovery algorithm and its FPGA implementation. In *2nd International Symposium on Intelligence Information Processing and Trusted Computing* (IEEE IPTC). Hubei (China), 2011, p. 63–66. DOI: 10.1109/IPTC.2011.23

[16] KIM, S. C., PLISHKER, W. L., BHATTACHARYYA, S. S., CAVALLARO, J. R. GPU-based acceleration of symbol timing recovery. In *Design and Architectures for Signal and Image Processing (IEEE DASIP)*. Karlsruhe (Germany), 2012, p. 1–8.

[17] HUA, J., ZHOU, L., CHEN, CH., JIANG, L. Synchronization for QDPSK Costas loop and Gardner algorithm using FPGAs. In *13th Internat. Conf. on Computer and Information Science*. Taiyuan (China), 2014, p. 27–31. DOI: 10.1109/ICIS.2014.6912102

About the Authors ...

Pavel FIALA was born in 1984 in the Czech Republic. In 2010 he received his M.Sc. from the Faculty of Electrical Engineering, University of West Bohemia in Pilsen. Since 2010 he has been a Ph.D. student and assistant at the faculty. His research interests cover software and cognitive radio. He specializes in digital signal processing using FPGAs.

Richard LINHART received his Ph.D. in Electrical Engineering from the Faculty of Electrical Engineering, University of West Bohemia in Pilsen in 2012. He is currently an Assistant Professor at the faculty. His research interests include wireless systems and space hardware design. He specializes in analog RF frontend design.