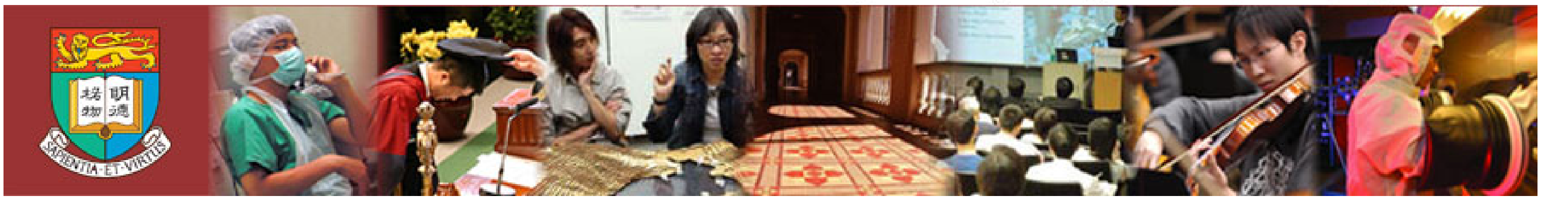

FPGA Implementation of Digital Timing Recovery in Software Radio Receiver

Yik-Chung Wu and Tung-Sang Ng
{ycwu, tsng}@eee.hku.hk

Department of Electrical and Electronic Engineering
The University of Hong Kong
Pokfulam Road, Hong Kong, China
Tel.: ++ 852 + 2857 8406   Fax: ++ 852 + 2559 8738

ABSTRACT

This paper describes an implementation of an all-digital timing recovery scheme. Squaring non-linearity is employed to generate the timing estimate and an IIR is used to extract the spectral component at symbol rate. Hardware design is performed using VHDL and realized in FPGA. The whole design can be fitted into an Altera EPF10K70 FPGA chip, with 95.5% utilization of logic elements and 22% utilization of memory bits. The implementation exploits features of FPGA, which enable easy implementation of look up table and variable data precision at different nodes.

I. INTRODUCTION

As the complexity of digital devices dramatically improved in the last decade, more and more functions in a communication receiver can be shifted from analog to digital. An obvious example is the development of software radio. In a software radio receiver, the received signal is sampled by a free-running local clock at IF and all the subsequent operations are performed in the digital domain. Traditional symbol timing recovery algorithm, which is performed by altering the timing phase of the sampling clock, needs to be replaced by fully digital realization.

When implementing an algorithm in hardware, we are facing with the tradeoff between performance and flexibility. ASIC, which possesses the advantage of low power consumption and high-speed, is only designed for a specific task. DSP, which can be programmed to implement different functions, requires many clock cycles to complete a task. FPGA lies in the middle of the two extremes. It is configurable and parallelism can be exploited to achieve high speed. Numerous applications in communication system implemented using FPGA are reported in the literature (e.g.. [6], [7], [8]). Apart from the flexibility and high performance, FPGA has two advantages: easy implementation of look-up table (LUT) and different precision can be easily accommodated at various nodes in the system such that required processing is exactly realized.

While VLSI implementation of digital timing recovery algorithm is popular, very few are implemented using FPGA platform. This paper presents a digital timing recovery scheme based on the squaring method [1], with an infinite impulse response (IIR) filter replacing the DFT operation [2]. The system was implemented using VHDL and realized in FPGA. Emphasis is being placed on how the features of FPGA are exploited in the implementation of LUT and variable data precision at different nodes.

This paper is organized as follows. The system will be described in section II. Section III presents how the property of user-definable precision is exploited, and section IV discusses how the look-up tables are used to simplify the design. Simulation results are presented in section V. Conclusions will be drawn in section VI.

II. SYSTEM DESCRIPTION

The overall system is shown in Figure 1. The sampled signal is first filtered by the digital matched filter and then spitted into two paths. The lower one is for estimating the unknown timing offset and the upper one, which is a first-in-first-out (FIFO) buffer, is for buffering the samples before the timing estimate is determined. In the offset time estimating path, an IIR filter is used to extract the harmonic at symbol rate [2]. Finally, the estimated timing information is output to the interpolator. In the following, some important components will be discussed in detail.

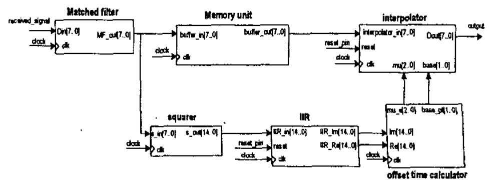

Figure 1. Block diagram of the overall system

a. *Interpolation*

Interpolation in timing recovery is the process of calculating the sample values in between the existing ones (i.e., compute samples $r(k-\mu_m T_s)$ from $r(k)$), such that optimum decision can be made from synchronized samples. The key equation [3] to the interpolation process is

$$x(m_k T_s - \mu_k T_s) = \sum_{i=I_1}^{I_2} x[(m_k - i)T_s] h_I[(i+\mu_k)T_s].$$

An interpolant is computed using $I = I_2 - I_1 + 1$ adjacent input samples about the base-point $x(m_k T_s)$ and I samples of the impulse response of the interpolation filter identified by the fractional interval μ_k.

Cubic interpolator, which is a member of polynomial-based approximating interpolation filter, can work well in typical modem applications [4]. Cubic interpolator can be implemented either using online calculation or LUT method. For online calculation, Farrow [5] proposed an efficient structure for calculating interpolants. For LUT method, μ_k has to be quantized into, say, L levels. Then the sets of impulse response samples correspond to each quantized μ_k are pre-computed and stored in a LUT. The correct set of impulse response samples is addressed by the fractional interval μ_k. The advantage of this method is that the computing burden is independent of the impulse response. However, the recovered clock suffers from a timing jitter of maximum value T_s / L.

b. *Squaring Timing Recovery*

The received signal for a linear modulation (PAM, QAM, PSK) is given by

$$r(t) = \sum_{n=-\infty}^{\infty} a_n g_T(t - nT - \varepsilon T) + n(t),$$

where a_n is the transmitted data symbol, $g_T(t)$ is the transmission signal pulse, T is the symbol duration, $n(t)$ is the Gaussian white noise with power spectral density N_o and ε is an unknown delay but assumed slow-varying.

After the received signal passes through the RF front end, where out of band noise is rejected, $r(t)$ will be sampled at rate $1/T_s = N/T$ and filtered by a digital matched filter with impulse response $g_R(kT/N)$. That is, the output of the digital matched filter is given by

$$\tilde{r}_k = \sum_{n=-\infty}^{\infty} a_n g(kT/N - nT - \varepsilon T) + \tilde{n}(kT/N),$$

where $g(m) = g_T(m) * g_R(m)$ and $\tilde{n}(kT/N)$ is the filtered and sampled noise.

M. Oerder and H. Meyr [1] proposed that the unknown timing delay can be generated by computing the complex Fourier coefficient at the symbol rate for every segment of LN samples (L symbols) of $\tilde{r}_k^2$. That is, the estimate is given by

$$\hat{\varepsilon}_m = -\frac{1}{2\pi} \times \arg\left(\sum_{k=mLN}^{(m+1)LN-1} \tilde{r}_k^2 e^{-j2\pi k/N}\right). \qquad (1)$$

One obvious implementation of (1) is the use of DFT. However, as only the Fourier coefficient at symbol rate is needed, the computation of other Fourier coefficients is unnecessary. M. Rahnema [2] showed that if $N = 4$, the computation of discrete Fourier coefficient at symbol rate is equivalent to the output of an IIR filter, with system transfer function $H(z) = jz^{-1}/(1-jz^{-1})$, at time $n = 4L$. This method reduces the computation of discrete Fourier Transform (DFT), which is a very hardware demanding process, to a simple filtering operation.

c. *Determination of interpolation control parameter*

The interpolator, however, does not use $\hat{\varepsilon}_m$ directly because the estimated timing delay $\hat{\varepsilon}_m$ is in term of T. An interpolator needs the fractional interval μ_m, which is in term of T_s, and the identification of basepoint set n_m, which is an integer. Therefore, $\hat{\varepsilon}_m$ must be converted to fractional offset time μ_m and basepoint set n_m. The relationship between $\hat{\varepsilon}_m$, μ_m and n_m is given by

$$\hat{\varepsilon}_m T = \mu_m T_s + n_m T_s.$$

Rearranging terms gives

$$\hat{\varepsilon}_m \times T / T_s = \mu_m + n_m. \qquad (2)$$

The timing delay estimate $\hat{\varepsilon}_m$ can be divided into five sub-ranges and each sub-range is handled individually. This is shown in Table 1. The cases for $\hat{\varepsilon}_m \geq -0.125$ are straightforward. For the case $-0.375 \leq \hat{\varepsilon}_m < -0.125$, n_m should be equal to -1 according to (2). Since the signal is sampled with rate $4/T$, basepoint at -1 is equivalent to basepoint at 3. The same principle applies to the case of $-0.5 \leq \hat{\varepsilon}_m < -0.375$.

$\hat{\varepsilon}_m$	n_m	μ_m
$-0.125 \leq \hat{\varepsilon}_m < 0.125$	0	$4 \times \hat{\varepsilon}_m$
$0.125 \leq \hat{\varepsilon}_m < 0.375$	1	$4 \times (\hat{\varepsilon}_m - 0.25)$
$-0.375 \leq \hat{\varepsilon}_m < -0.125$	3	$4 \times (\hat{\varepsilon}_m + 0.25)$
$0.375 \leq \hat{\varepsilon}_m < 0.5$	2	$4 \times (\hat{\varepsilon}_m - 0.5)$
$-0.5 \leq \hat{\varepsilon}_m < -0.375$	2	$4 \times (\hat{\varepsilon}_m + 0.5)$

Table 1. Conversion from $\hat{\varepsilon}_m$ to n_m and μ_m

III. USER-DEFINABLE PRECISION

One of the features of FPGA is the flexibility to enable designers to define different precision at various nodes in the system. This allows the design to match with the required processing precision and minimize the use of resource. In the following, some subsystems implemented using different precision at various nodes are described.

a. Matched filter

The matched filter is a root raised cosine filter with roll off factor equals 0.3. It is a 17 taps FIR with coefficients quantized to 8 bits. The filter architecture is shown in Figure 2 with Table 2 showing the data formats at different nodes. Taking advantage of the coefficients symmetry, the symmetric taps can be added together before multiplied by the coefficients. Assuming the input data samples are 8-bit numbers with 7 bits representing fraction (i.e., it's magnitude is smaller than unity), the result from addition may extend one more bit. Therefore, in order to avoid overflow, 9 bits are needed in node A, with 7 bits representing fraction.

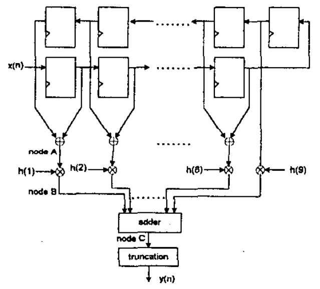

Figure 2. Filter architecture for the matched filter

	Data format
x(n)	8 bits (x.xxxxxxx)
node A	9 bits (xx.xxxxxxx)
h(n)	8 bits (xx.xxxxxx)
node B, node C	17 bits (xxxx.xxxxxxxxxxxxx)
y(n)	8 bits (x.xxxxxxx)

Table 2. Data format at various nodes in the matched filter

Since the filter coefficients are 8 bits with 6 bits representing fraction, a 17-bit data with 13 bits representing fraction results after multiplication (node B). Moreover, as $\sum_{n=1}^{17}|h(n)| = 5.6719$, the output from the final summation will have a maximum value of 5.6719. This can still be represented by the same 17-bit word, so the data format at node C is the same as that in node B. At the output of the filter, the result is truncated to 8 bits with fractional point left shifted 3 bits (i.e., divide the result by 8) such that the same format results at output $MF\_out[7..0]$ as that of input $Din[7..0]$.

b. IIR filter

The implementation of the IIR filter is shown in Figure 3, along with the internal register names. Theoretically, the output of an IIR filter will grow unbounded as index of output increases. However, in this application, only the output at $n = 4L$ is needed and the IIR filter will be cleared for another estimate. Since the output from the matched filter will always be smaller than 0.709 (5.6719/8), the output of the squarer will always be smaller than 0.503 ($[5.6719/8]^2$). Using matlab simulation, for the input to the IIR filter with magnitude smaller than 0.503, the magnitude of the output will always be smaller than 1 for $L \leq 64$. Therefore, if a segment of 64 symbols is used for timing estimate, the data formats for the input, outputs and the internal registers of the IIR filter are the same.

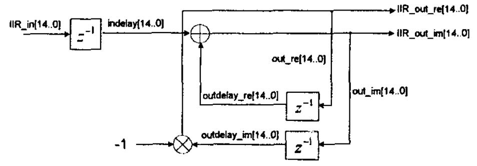

Figure 3. IIR filter used in timing delay estimation

c. Interpolator (online calculation)

For interpolator implemented using the Farrow structure [5], the output is given by

$$y(kT) = \{[v(3)\mu_k + v(2)]\mu_k + v(1)\}\mu_k + v(0)\,. \quad (3)$$

This equation shows that if the results from each multiplication with μ_k are not truncated, a very long register is needed to store the intermediate result. If truncation is done on each intermediate result, analysis of quantization effect has to be carried out in order to determine how many bits should be truncated in each step. Figure 4 shows the linear noise model [9] for the Farrow structure.

Assuming there is no quantization error in the Farrow coefficients, the first source of quantization error is introduced if we truncate $v(n)$ to a smaller number of bits, which is denoted by e_v in Figure 4.

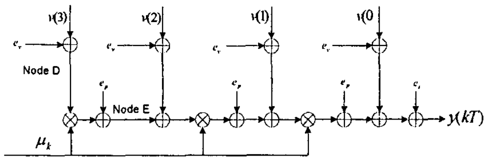

Figure 4. Linear noise model for the farrow structure

The second source of quantization error occurs when the product after multiplication with μ_k is truncated, which is denoted by e_p. The final source of error is the truncation before the interpolator's output, which is denoted by e_t. Analysis shows that the total quantization error at the output of the interpolator is given by

$$e = e_v(\mu_k^3 + \mu_k^2 + \mu_k + 1) + e_p(\mu_k^2 + \mu_k + 1) + e_t \cdot$$

Since $-0.5 \le \mu_k < 0.5$, it follows that

$$e < 2e_v + 2e_p + e_t .$$

If all the internal registers in the Farrow structure are 8 bits long (byte) with 7 bits representing fraction, the total error at output error e is bounded by

$$e < 1/2^5 + 1/2^7 .$$

If truncation is performed at the output only, the total error is bounded by

$$e < 1/2^7 .$$

Consider a compromise between the above two cases, suppose $v(n)$ is not truncated, the product after multiplication with μ_k is truncated to 13 bits with 11-bit fraction and the final product is truncated to 8 bits with 7-bit fraction before output. Then the total error at the output is bounded by

$$e < 1/2^{10} + 1/2^7 .$$

The total error at the output in this case is about the same as that of truncation at the output only. However, the length of the internal registers is much shorter. Therefore, this truncation approach was implemented for online calculation of the interpolator.

IV. LUT EXPLOITATION

Another property of FPGA is the ease in which LUT can be implemented. This property can be exploited to greatly simplify the circuit. Two examples are described in the following.

a. *Interpolator (LUT approach)*

If a little bit jitter can be tolerated, the parameter μ_k can be quantized and the Farrow structure can be replaced by a simple 4 taps FIR filter with the coefficients (impulse response samples $h_I[(i + \mu_k)T_s]$) pre-computed and stored in a LUT. Figure 5 shows a simplified diagram of the interpolator using the LUT approach. In this case, μ_k is quantized into 8 levels. Figure 5 also includes a multiplexier, which is controlled by parameter $base\_pt[1..0]$, for determining which sets of samples to be used in interpolation. Actual implementation in FPGA shows that the LUT method uses about 30% less logic elements when compared with the online calculation method.

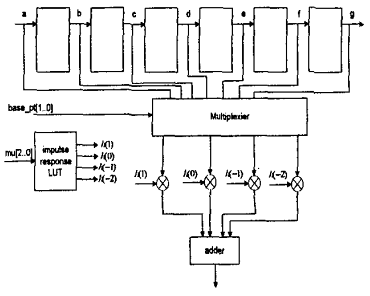

Figure 5. Interpolator using LUT method with multiplexier

b. *Calculation of interpolation control parameters*

LUTs are used to calculate the control parameters for interpolation. The block diagram for this process is shown in Figure 6, with Table 3 showing the data format at various nodes. The process of determining timing delay estimate $\hat{\varepsilon}_m$ is implemented by four steps. First, the sign bits and magnitude of the IIR filter outputs are separated. Second, absolute value of the imaginary part is divided by absolute value of the real part. In this step, only the first quadrant is considered. Third, the arctan value of the quotient is obtained by LUT, which is shown in Table 4. The use of arctan LUT, whose resolution is determined by the number of quantization levels of μ_k, avoids the difficult implementation of arctan calculation in FPGA. Fourth, the quadrant in which the angle is located is determined by the sign bit extracted in the earlier step, according to Table 5. The output from the quadrant LUT is the timing delay estimate defined in (1) and is quantized into 32 levels. Finally, the timing delay estimate is converted to fractional offset time μ_m and basepoint set n_m by the offset time conversion LUT, which is identical to Table 1. The use of LUT in this step reduces the calculation of (2) into simple additions and shifts operations.

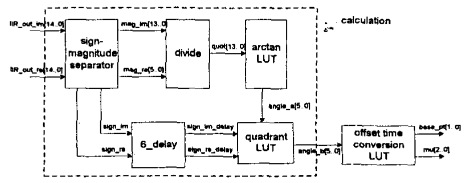

Figure 6. Block diagram for the calculation of interpolation control parameters

	Data format
IIR_out_re, IIR_out_im	15 bits (x.xxxxxxxxxxxxxx)
mag_im	14 bits (.xxxxxxxxxxxxxx)
mag_re	6 bits (.xxxxxx)
quot	14 bits (xxxxxx.xxxxxxxx)
angle_a, angle_b	6 bits (x.xxxxx)
base_pt	2 bits (xx)
mu	3 bits (.xxx)

Table 3. Data format at various nodes in fractional offset time calculation module

V. SIMULATION RESULTS

After each individual module are compiled and tested, they are integrated and compiled together. The whole receiver part can be fitted into an Altera EPF10K70 FPGA chip, with resources allocated to various components shown in Table 6. The overall design is a little bit smaller than the sum of individual design blocks since when compiling individual blocks, some logic elements may not be fully utilized. The overall receiver design occupies 95.5% logic elements and 22% memory bits of the EPF10K70 FPGA.

VI. CONCLUSIONS

This paper presented an implementation of the digital square timing recovery using FPGA. The effects of roundoff noise inside the interpolation filter have been analyzed. Look-up tables (LUT) are employed to replace some computationally intensive tasks. The flexibility of assigning different precision at various nodes in the system has been demonstrated.

ACKNOWLEDGEMENT

This work was supported by the Hong Kong Research Grants Council and by the University Research Committee of The University of Hong Kong, Hong Kong.

REFERENCES

[1] M. Oerder, H. Meyr, "Digital filter and square timing recovery," *IEEE Trans. Commun.*, vol. 36, pp. 605-611, May 1988.

[2] M. Rahnema, "Symbol timing recovery and tracking method for burst-mode digital communications," *US Patent*, Patent Number: 5870443.

[3] F. M. Garnder, "Interpolation in digital modems — part I: fundamentals," *IEEE Trans. Commun.*, vol. 41, pp. 501-507, Mar. 1993.

[4] L. Erup, F. Gardner, R. A. Harris, "Interpolation in digital modems — part II: implementation and performance," *IEEE Trans. Commun.*, vol. 41, pp. 908-1008, Jun. 1993.

[5] C. W. Farrow, "A continuously variable digital delay element," *Proceedings of ISCAS 88*, pp.2641-2645.

[6] H. S. Park, K, Y, Sohn, D. H. Kim, "The implementation of modulator using FPGA technology for W-CDMA WLL," *Proceedings of IEEE ASIC Conference and Exhibit 1997*, pp.79-83

[7] D. H. Lee, A. Choi, J. M. Koo, J. I Lee, B. M. Kim, "A wideband DS-CDMA modem for a mobile station," *IEEE Trans. Consumer electronics*, vol. 45, No. 4, Nov. 1999, pp.1259-1269

[8] J. M. P. Langlois, D. A. Khalili, R. J. Inkol, "A high performance, wide bandwidth, low cost FPGA-based quadrature demodulator," *Proceedings of the 1999 IEEE Canadian conference in electrical and computer engineering*, pp.497-502

[9] A. V. Oppenheim, R. W. Schafer, "Discrete-time signal processing," Prentice-Hall, 1989

Quotient range	angle_a *($\theta/2\pi$)*
Quotient ≥ 10.1523	0.25
10.1523 > Quotient ≥ 3.293	0.21875
3.293 > Quotient ≥ 1.8672	0.1875
1.8672 > Quotient ≥ 1.2148	0.15625
1.2148 > Quotient ≥ 0.8203	0.125
0.8203 > Quotient ≥ 0.5313	0.09375
0.5313 > Quotient ≥ 0.3008	0.0625
0.3008 > Quotient ≥ 0.0977	0.03125
0.0977 > Quotient	0

Table 4. Arctan look-up-table

sign_im	sign_re	$\theta/2\pi$	angle_b ($-\theta/2\pi$)
+	+	angle_a	-angle_a
+	-	0.5-angle_a	-0.5+angle_a
-	+	-angle_a	Angle_a
-	-	-0.5+angle_a	0.5-angle_a

Table 5. Quadrant look-up-table

	Logic elements	% of EPF10K70
Overall receiver design	3575	95.5%
Matched filter	1189	31.75%
FIFO buffer	332	8.86%
Squarer	183	4.88%
IIR filter	257	6.86%
Timing offset calculation	403	10.76%
Interpolator with multiplexier	1243	33.2%

Table 6. Resource allocation to various subsystems