King Fahd University of Petroleum and Minerals

College of Computer Science and Engineering

Computer Engineering Department

Final Report for the Internal Project (IN000287):

Development of Digital Circuit Techniques for Clock Recovery and Data Re-Timing for High Speed Non-Return-to-Zero Source-Synchronous Serial Data Communications

Muhammad E. S. Elrabaa

P.O.Box 584, Dhahran-31261
Saudi Arabia
Email: elrabaa@kfupm.edu.sa

JUNE 2007

Abstract

Novel circuit techniques for performing clock recovery and data re-timing functions for high-speed source synchronous data communications, such as in burst-mode data transmission have been developed. Two types of circuits have been developed; one for general synchronous Non-Return-to-Zero (NRZ) serial data communications and another for source-synchronous NRZ serial data communications. These two new techniques are fully digital, non-PLL-based, and capable of retiming the output clock with the received data within two data transitions. The absence of analog filters or other analog blocks make their area much smaller than conventional circuitry. They can also be described by any hardware description language, simulated, and synthesized into any digital process. This enables them to be ported from one technology to another and support system on a chip (SOC) designs. The performance of these circuits have been verified with Tspice® simulations using a 0.13µm, 1.2V CMOS technology. The results of this work have been published in several conference and journal publications.

Key Words: Clock Recovery, NRZ serial data communications, Source-Synchronous Serial Communications, Systems-on-a chip, Digital CMOS Circuits.

Table of Contents

1. Introduction

This report details the findings and results of a two-year research project funded by King Fahd University of Petroleum and Minerals under grant number IN000287. In section 2, some background information on the topic of clock recovery and data retiming is provided. These show the motivation behind this work. In section 3 …

2. Background and Motivation

The fast expansion of multimedia (Audio/Video) over the Internet in recent years has caused an explosion in the data transfer volume over wide-area networks (WANs) and local-area networks (LANs). This led to a rapid migration from copper wires to optical fibers as transmission media for high-speed digital-transmission schemes such as synchronous optical networks/synchronous digital hierarchy (SONET/SDH). This in turn necessitated the development of low-cost high-speed clock recovery circuits (CRCs) in the repeaters and receivers that can accurately extract clock signals from non return-to-zero (NRZ) source-synchronous serial bit streams (Figure 1). The CRC must maintain synchronism between the generated clock and the data in the presence of data phase noise (jitter), supply and temperature fluctuations. Also, in a point-to-multipoint operating scenario (as in optical networks), the CRC must be agile in extracting synchronized clocks for different data packets arriving from different sources. This is very important for burst-mode source-synchronous serial data communication systems (such as ATM and Gigabit Ethernets network switches).

Also, the high data rates and reliability have made serial communications the most preferred inter-chip communication method. Applications range from backplane communications, to fiber data transmission. With the emergence of large systems-on-chips, new design methodologies are being adopted to cope with design complexity and wire delays. For the 90 nm technology, different architectural explorations have shown that wire delays of up to 1.5ns are not unusual [1]. For 5 GHz clock, this delay exceeds seven clock cycles. Also, the Semiconductors Industry Association estimates that by 2010 cross-chip wire delays are going to exceed 12 clock cycles [2]. On top of this, ASICs are being assembled from pre-designed, usually heterogeneous in nature, blocks (i.e. IPs) that are interconnected together. Each block has its own operating frequency and communication needs. This necessitated a paradigm shift to enable quick timing closure of the whole chip; namely the use of networks-on-Chip [3]. These networks are made of routers (switches) and links. As the on-chip global interconnect technology moves from ad-hoc wiring or bus-based to switch/router based schemes, the demand for low overhead, low-power and portable serial communication circuits (as IP blocks for link implementation) will be ever increasing.

Two types of NRZ synchronous serial data communications are possible. In the first type, illustrated in Figure 2, NRZ data (**D_{Tx}**) is sent by the transmitter using a clock (**Clk_{Tx}**). At the receiver side, a clock-recovery circuit (**CRC**) would extract the clock (**Clk_{Rx}**) from the received data, re-time this clock with the data and use it to sample the data. Additional signal processing/conditioning such as equalization might be applied at the transmitter's side, receiver's side or both. This type of synchronous serial communication is more suited to inter-chip (i.e. off-chip) communications due to limitations on the number of chip's I/O pins. However it requires the CRC to perform both frequency and phase locking.

The second type NRZ synchronous serial data communications, called source synchronous, the transmitter sends the clock along with the data. This clock arrives at the receiver side out of phase with the data, hence the receiver re-synchronize it with the data before using it to sample the data. This would be also be a viable alternative to on-chip source synchronous parallel buses in deep sub-micron technologies where global wire delays amount to several clock cycles which necessitate careful design and elaborate skew control circuitry. This is the same argument that made serial communications replace parallel communications for inter-chip communications.

A source synchronous serial link (**S3L**) is illustrated in Figure 3. The FIFO buffers (synchronous at the transmitter side and asynchronous at the receiver side) would be required if the communicating entities operate at different clock speeds [4-8]. This type is also suitable for implementing on-chip communications (NoC links) with multiple-clock domains since there is no limitation on sending the clock along with the data and it only requires phase locking.

For on-chip source synchronous serial links (clock is sent with the data), as with inter-chip communications, the issue of re-timing the clock with the received data arises. Having a clock-recovery and data retiming circuit (CRC) would allow very high serial data rates and help with the timing closure of the chip. These circuits, setting in the repeaters and receivers, accurately extract the clock signals from non return-to-zero (NRZ) serial bit streams where they are embedded. Such circuit must maintain synchronism between the clock and the data in the presence of data phase noise (jitter), supply and temperature fluctuations. Also, they must be agile in extracting synchronized clocks for different data packets arriving from different sources.

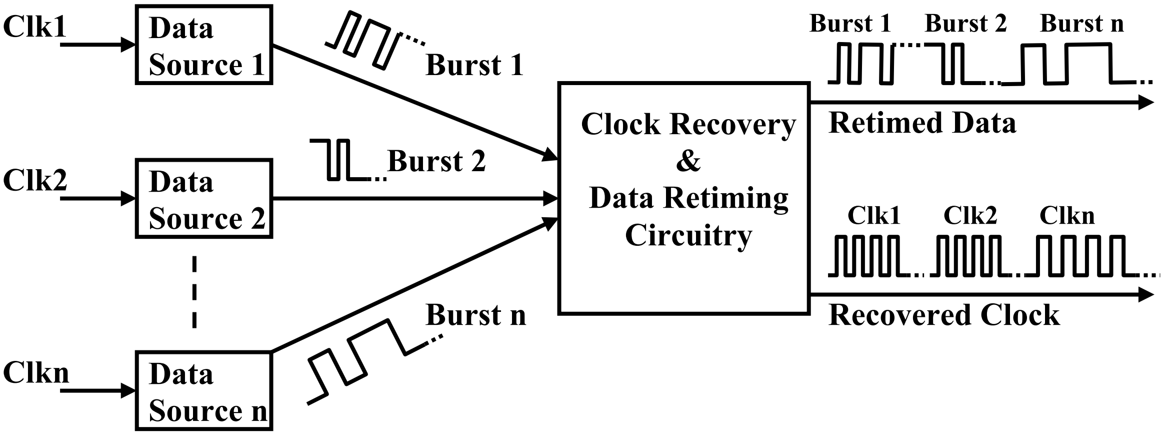


Figure 1: The role of the Clock Recovery Circuit (CRC) in a multi-source communication scheme such as in SONET/SDH.

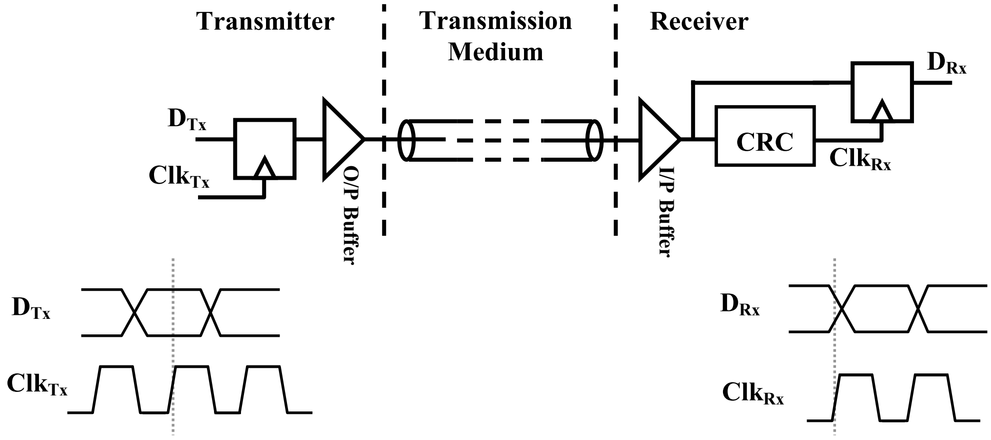


Figure 2. A typical synchronous serial Link with clock-recovery circuit at the receiver for Off-chip communications. The O/P and I/P buffers might implement some form of signal conditioning (such as equalization).

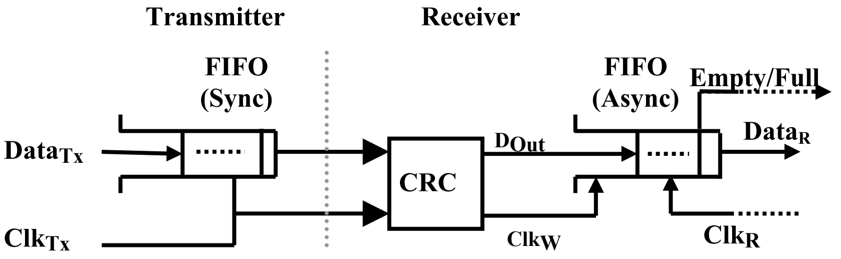


Figure 3. The general structure of a Source Synchronous Serial Link (S3L).

2.1 Clock Recovery Techniques

Conventionally, two major techniques were used to implement CRCs; 1) Using phase-locked-loops (PLLs) or delay-locked loops (DLLs) [9-12] and 2) Over-sampling [13-19]. The two techniques are reviewed below.

PLL/DLL Based CRCs:

PLL/DLL circuits represent automatic feedback control systems that lock the frequency and phase of the output clock to that of the input NRZ data. A PLL-based CRC is illustrated in Figure 4. A PLL-based CRC operates as follows; a Phase-Frequency Detector (**PFD**) generates two digital output signals, UP and DOWN, depending on the phase difference between the input data and the generated clock. A frequency or phase difference will be detected as a phase difference and the **PDF** will generate a steady stream of UP or DOWN signals. These signals cause the charge pump (**CP**) to output or source a current, which is converted to a voltage and filtered by the loop filter. This voltage controls the output frequency of the Voltage-Controlled-Oscillator (**VCO**). The **VCO**'s output, which is the recovered clock is then fed back to the **PFD.** If clock pulse symmetry is required (i.e. 50% duty cycle) the **VCO**'s output is divided by 2 before it is fed back to the **PFD** (in this case the VCO will be running at twice the clock frequency). The PLL's negative feedback forces the frequency and phase of the recovered clock to be the same as the bit rate and phase, the lock condition.

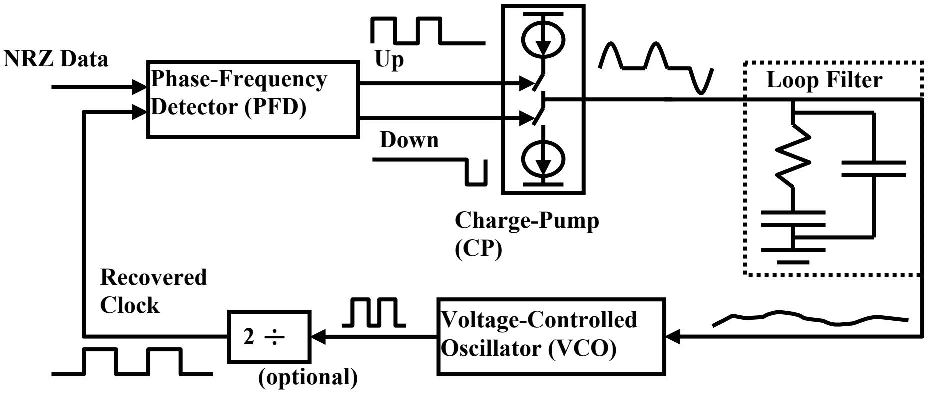


Figure 4. The general structure of a PLL-based Clock-Recovery Circuit (CRC)

PLLs, being mixed-signal systems, are very difficult to design because of the sensitivity of the analog blocks to noise generated by the digital blocks nearby. They also have to be re-designed every time they are ported to a different process. The large area of analog blocks limits the number of PLLs that can be incorporated on a SoC.

The job of a PLL is made difficult due to the nature of random NRZ data; they have no spectral contents at the bit rate or its even-order harmonics [9, 10]. This problem can be circumvented using edge detection techniques (sometimes referred to as input differentiation). Also, long streams of consecutive 1s and 0s represent a serious problem to traditional clock recovery circuits that utilize PLLs because they are viewed as lower frequency data (the PLL tries to re-lock to this new frequency). This result in data-dependent jitter in the PLL’s output. Data encoding schemes help reduce this jitter but does not eliminate it. To reduce this jitter further, the PLL’s bandwidth has to be limited which in turn limits the PLL’s capture range. Also, larger loop gain reduces jitter generation but reduces stability as well.

Additionally, analog PLLs suffer from additional shortcomings; large area (due to analog blocks), difficulty to port to other processes or supplies, high VCO operating frequency (double the data frequency) which in turn limits the data rate, phase error accumulation in the VCO, and long lock times due to the loop damping behavior.

A CRC based on a fully digital PLL was proposed in [20] to solve some of the analog PLLs problems. A high frequency clock (that must be synchronized to an external reference clock) is used as a reference clock for the digital PLL circuits. This limits the maximum data rate the circuit can handle and the resolution. Though it achieved a moderate resolution, it still has a large area (4393 gates) and potential stability problem (due to loop behavior). Many other researchers proposed fully digital or semi-digital solutions in an attempt to solve some of the analog PLLs problems [21-28]. The main strategy followed by these workers is to replace some of the PLL's analog blocks with a digital equivalent. Though these techniques retain many of the analog features, they suffer from poor resolution (more jitter), stability issues (two or more loops interacting), difficulty to port from one process to another (due to analog blocks) and/or large areas.

Over sampling CRCs:

The basic architecture of a CRC that uses over-sampling is shown in Figure 5 [29-32]. Several equidistant phases (*n*) of a reference clock are generated and then used to sample the incoming data. This requires the period of an *n*-stage oscillator to be locked to the data (clock) frequency. An alternative would be to generate multi phases of the data and use a single clock to sample them. A decision circuit then selects the appropriate clock phase and data value. The minimum number of samples required per bit is 3. The higher the over-sampling, the better the resolution and the bit-error-rate (BER) [32]. Several bits could be sampled simultaneously with de-multiplexing performed on the output data. The different clock phases are generated from the reference clock either; through dividers and logic gates, or using a PLL or DLL.

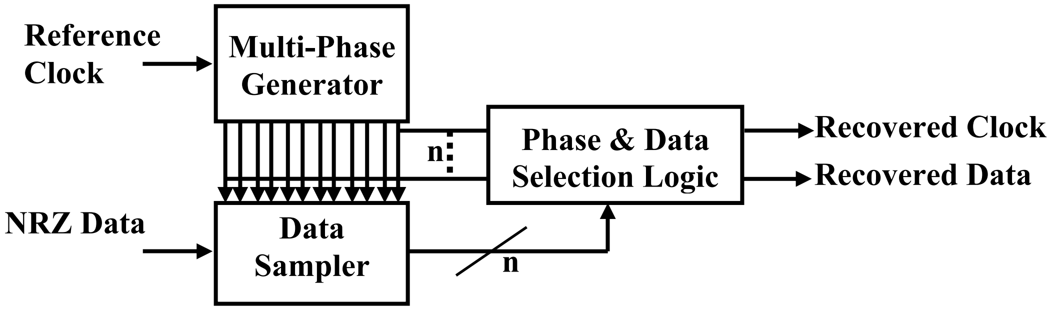


Figure 5. General architecture of an over sampling CRC.

The first method, with multiple clock phases, requires a clock reference with a much higher frequency than the bit-rate which limits the maximum bit-rate for a given technology. The second method requires the use of analog blocks that neither are easy portable nor can be replicated due to their large footprint. Both methods require the reference clock to have a frequency with a very precise relationship to the input data bit-rate. Any frequency mismatch will directly contribute to the jitter of the recovered clock.

Resolution can be enhanced by using two matched delay lines in the sampler; one for data and another for the clock [29]. This however, requires two DLLs which increases the area significantly.

3. Project Scope and Objectives

The main objective of this research was to devise fully digital circuit techniques for clock-recovery and data-retiming thus allowing cell-based designs that can be ported from one process to another. This would enable the use of such blocks in SoC designs.

Two techniques were to be investigated for the design of all-digital clock recovery circuits:

1. A variable length digital ring oscillator with both frequency and phase capturing. This would enable locking and data retiming within one data transition. Phase interpolation techniques were to be investigated to reduce frequency mismatch that can result from using digital ring oscillators with hard timing boundaries.

2. Direct Digital Frequency Synthesis (DDFS) techniques for synthesizing the required frequency combined with a digital variable-delay line for phase alignment. DDFS techniques were not used before in frequency locking in clock recovery systems.

The research was planed over a period of 24 calendar months starting from the time of approval of the project. A total of 9 tasks (summarized below) were planned to achieve the project objectives:

1. Development of a variable-length digital ring oscillator circuit,
2. Development of a phase interpolation technique,
3. Development of edge detection and wave-edge capturing techniques,
4. Integration of the above blocks with the required control circuitry to create a completely digital CRC based on variable length digital ring oscillators,
5. Investigation of digital direct frequency synthesizers (DDFS),
6. Development of a variable length delay line. This block can be combined with the DDFS to achieve both frequency and phase locking to the input data or be used in source synchronous serial communications,
7. Integration of the DDFS and variable delay line with any required control logic to form another implementation of a fully digital CRC,

8. Evaluation and comparison of both CRC implementations,

9. Generation of periodic and final project reports and authoring of publications for conferences/journals.

The schedule that was proposed for accomplishing the above tasks is shown in table 1 below.

Table 1: Project Time Table

	Months							
Tasks	**01-03**	**04-06**	**07-09**	**10-12**	**13-15**	**16-18**	**19-21**	**22-24**
Task 1	■	■	■					
Task 2		■	■					
Task 3	■	■						
Task 4		■	■	■				
Task 5			■	■	■			
Task 6				■	■	■		
Task 7					■	■		
Task 8							■	■
Task 9		■	■	■	■	■	■	■

4. Results and Accomplishments

Three major accomplishments resulted from this work; A fully-digital CRC that can extract the clock from a NRZ serial bit stream and re-times the data with the extracted clock, another CRC for source-synchronous NRZ data and investigation of the feasibility of using DDFS to make a CRC. The details of these research accomplishments are presented below.

4.1 Novel Fully-Digital CRC Circuit

The basic idea behind the developed digital CRC circuit is to use a variable length digital delay line to measure the input data's bit duration (i.e. the time between a positive edge and a negative edge). This delay is then used in a feedback loop to form an oscillator with an oscillation period equal to that delay (i.e. the oscillation frequency will be twice the maximum data rate). The resolution of the digital delay line is increased by allowing multiple feedback points or interpolation between stages.

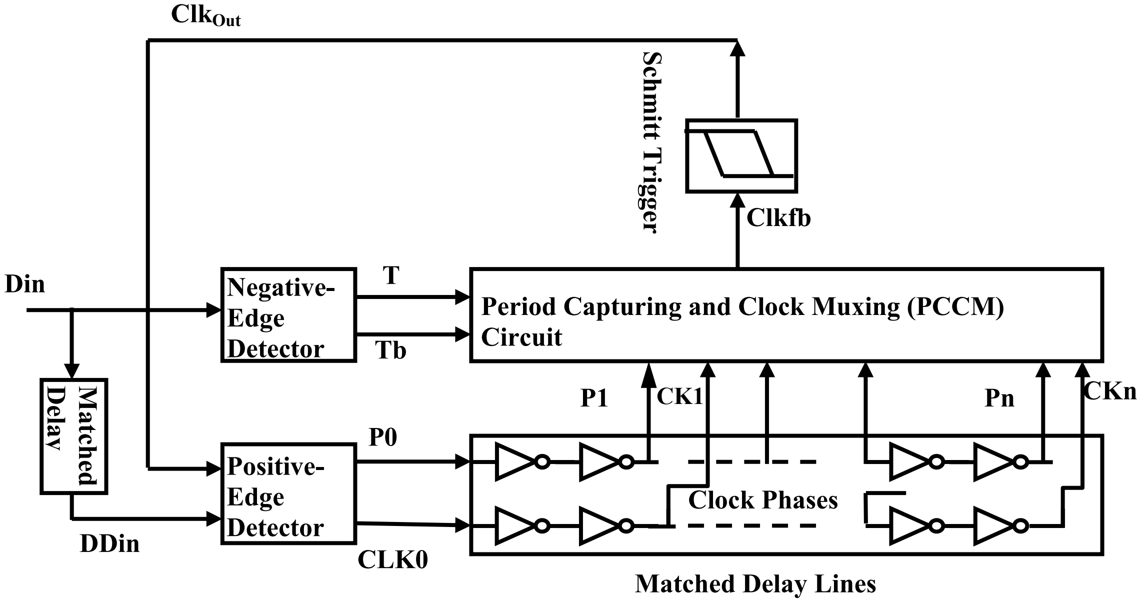


Figure 6. The architecture of the clock-recovery circuit proposed for off-chip serial communications.

The architecture of the proposed **CRC** is shown in Figure 6. When the incoming NRZ data (**Din**) exhibits a positive transition (0→1) for the first time, the positive-edge detector (**PED**) circuit produces a pulse (**P_0**) that would travel down the digital delay line. This takes a delay of T_{P0} (referenced to the positive edge of the input data):

$$T_{P0} = T_{FD} + T_{PED} \tag{1}$$

Where T_{FD} is the fixed delay and T_{PED} is the delay through the **PED**.

The delay line will provide several phases of this pulse (**P_1** to **P_n**). An identical delay line will also contain a replica of this pulse, providing unloaded clock phases (**CLK_0** to **CLK_n**). When the next negative NRZ data edge arrives, the negative-edge detector (**NED**) circuit will generate two complementary pulses, **T** and **T_b,** which will be used to latch in the appropriate pulse phase (one of **P_0** ~ **P_n**).

This latched pulse phase is then used as an enable signal for a mux to select the corresponding clock phase. More than one clock phase might be selected and latched, making the resulting feedback clock (**Clk_{fb}**) an interpolated version of the selected clock phases, enhancing the **CRC**'s resolution significantly. The fed-back clock would have a delay T_{clkfb} (again referenced to the positive edge of the input data):

$$T_{clkfb} = T_B + T_{NED} \tag{2}$$

Where $\boldsymbol{T_B}$ is the bit duration and $\boldsymbol{T_{NED}}$ is the delay of the **NED** circuit. This means that the captured delay in the variable delay line is $\boldsymbol{T_{DL}}$:

$$\boldsymbol{T_{DL} = T_{clkfb} - T_{P0}}$$

$$\boldsymbol{= T_B + T_{NED} - T_{FD} - T_{PED}} \quad (3)$$

The fed-back clock is then re-constructed using a simple Schmitt Trigger to form the output clock and is fed back to the 2nd input of the **PED** circuit. The output clock $\mathbf{Clk_{Out}}$ is produced after a delay $\boldsymbol{T_{clkOut}}$:

$$\boldsymbol{T_{clkOut} = T_{clkfb} + T_{ST}} \quad (4)$$

Where $\boldsymbol{T_{ST}}$ is the delay through the selection multiplexer and Schmitt Trigger. This closes the loop and from this point onward forms a ring oscillator with a total delay of $\boldsymbol{T_{CLK}}$:

$$\boldsymbol{T_{CLK} = \quad T_{PED} + T_{DL} + T_{ST}}$$

Substituting $\boldsymbol{T_{DL}}$ from (3):

$$\boldsymbol{T_{CLK}} = \boldsymbol{T_{PED} + T_B + T_{NED}} - \boldsymbol{T_{FD} - T_{PED} + T_{ST}}$$

$$\boldsymbol{= T_B + T_{NED} + T_{ST} - T_{FD}} \quad (5)$$

And if $\boldsymbol{T_{FD}}$ is set as such:

$$\boldsymbol{T_{FD} = T_{NED} + T_{ST}} \quad (6)$$

Then from (5), the oscillator's total delay $\boldsymbol{T_{CLK}}$ becomes:

$\boldsymbol{T_{CLK} = T_B}$ or 1 bit duration (i.e. bit cell)

The first $\mathbf{Clk_{Out}}$ pulse will reach the **PED** input (referenced to the first positive data edge), using (1-6):

$$\boldsymbol{t_{CLK1}} = \boldsymbol{T_{FD} + T_{PED} + T_{DL} + T_{ST}}$$

$$= \boldsymbol{T_{FD} + T_{PED} + T_B + T_{NED}} - \boldsymbol{T_{FD} - T_{PED} + T_{ST}}$$

$$= \boldsymbol{T_B + T_{NED} + T_{ST}}$$

$$= \boldsymbol{T_B + T_{FD}}$$

The second pulse would be at $\boldsymbol{t_{CLK2}} = 2\ \boldsymbol{T_B + T_{FD}}$

And the nth pulse $\boldsymbol{t_{CLKn}} = \mathrm{n}\ \boldsymbol{T_B + T_{FD}}$.

Hence the output clock pulses are not only at twice the input frequency but it is also aligned with the delayed version of the input data (DDin). This means that the CRC will continue to oscillate at twice the input's frequency even with the absence of any positive or negative input data edges. With each data transition the circuit re-times the clock with data, hence correct for any injected phase and/or frequency noise. The inverted version of **Clk$_{Out}$**, which is in the middle of the bit cell, is used to sample and latch in the delayed data bit (DDin), as illustrated in Figure 7.

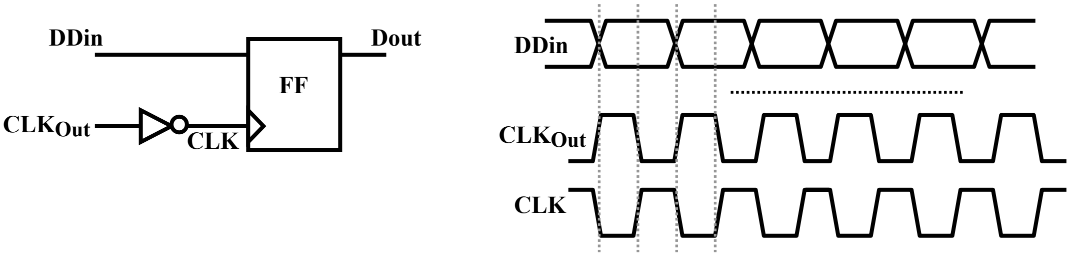


Figure 7. Timing relations between the delayed input data (DDin), CRC's output clock (CLK$_{Out}$) and sampling clock (CLK).

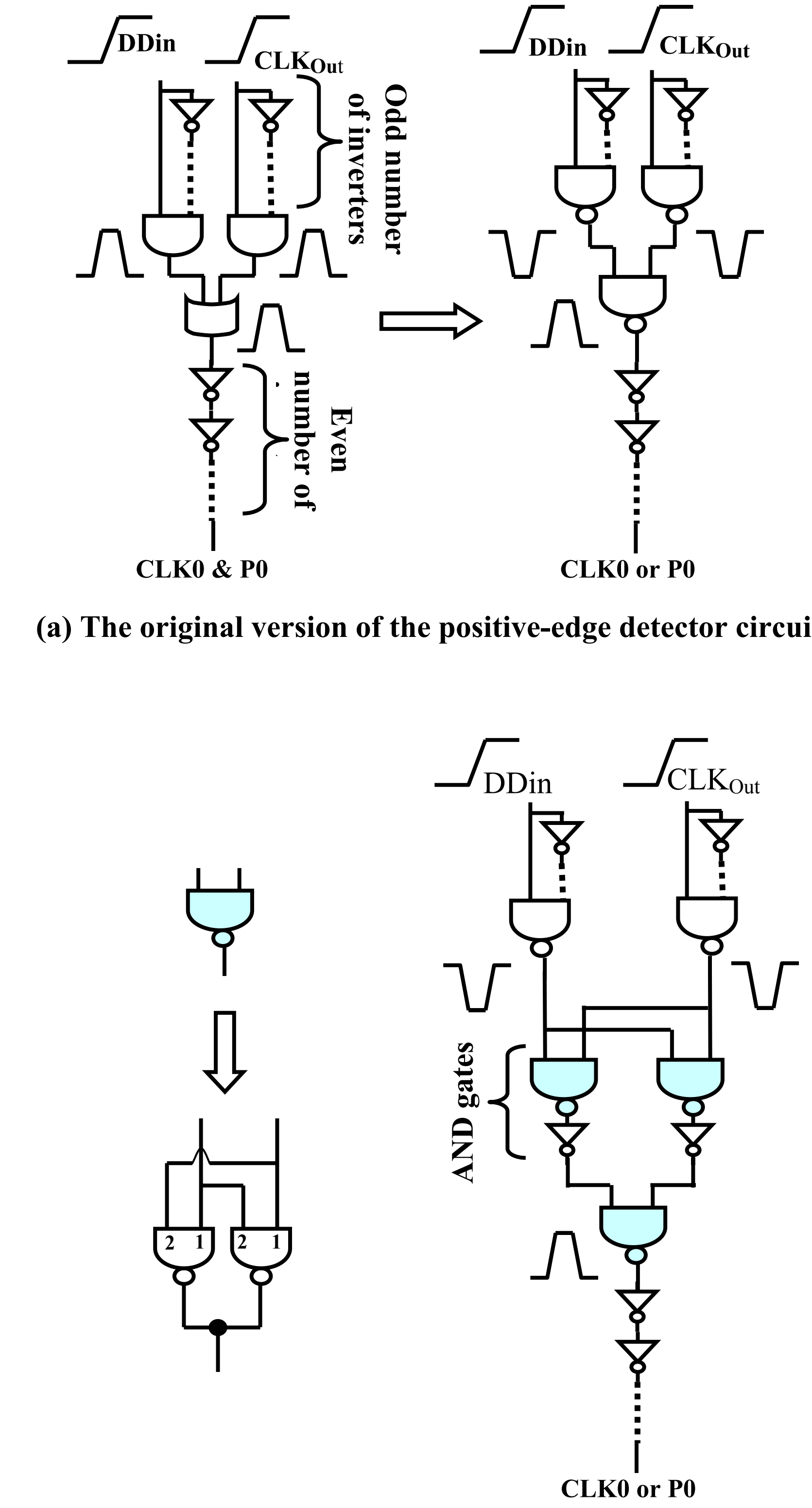
Circuit Components

The developed CRC circuit has five main components; a positive-edge detector circuit (**PED**), a negative-edge detector circuit (**NED**), two delay lines, a period capturing and clock muxing (**PCCM**) circuit, and a Schmitt Trigger. All blocks are implemented using standard CMOS circuits.

The PED circuit

The **PED** circuit generates two narrow output pulses, Po and CLK0, for every positive transition in the delayed NRZ input data (**DDin**) or the output clock (**CLK$_{Out}$**). These pulses would then be fed to the two identical delay lines. Figure 8 shows the development of the **PED** circuit from the basic conceptual circuit to the final implementation. 2 I/P AND gates are used to generate pulses from the inputs (DDin and CLK$_{Out}$) and their delayed versions. The delay, set by the odd number of inverters at the input of the AND gates, determines the width of the generated pulses. These pulses are Ored and then delayed by an even number of inverters to reduce the number of stages in the matched delay lines and the PCCM circuits. The number of these inverters depends on the intended data rate.

The conceptual **PED** circuit is first converted to CMOS NAND gates. Then an AND stage (implemented as NANDs followed by inverters) is added. This stage evens out the output pulses and eliminates the effects of any skew between DDin and CLK$_{Out}$. Finally, the NAND gates in the AND stage and the final OR stage (shaded in Figure 8) are implemented using 2 parallel-connected NANDs, as shown in the figure. The primary inputs are alternated between the inputs of the two NANDs to reduce the differences in delay between the two primary inputs. These simple actions reduce the jitter in the output clock significantly.



(a) The original version of the positive-edge detector circuit.

(b) The improved version of the positive-edge detector

Figure 8. The positive-edge detector circuit.

Figure 9 shows the input and output waveforms of the **PED** circuit at 2GBps data rate. It shows that when only one of CLK_{Out} or DDin exhibits a positive transition, the PED's output clock pulse (PClkOut) width remains the same. When both of these signals exhibit a positive transition simultaneously the output pulse is slightly wider. This is due to the reduced delay of the PED circuit at this input condition. If it was not for the actions taken above, this effect would have been much larger (as will be demonstrated later).

Two matched **PED** circuits with different pulse widths are used to generate CLK0 and P0. P0 is set to the minimum possible width of three inverters, while CLK0 is set to ~ TB/2. The two pulses are perfectly synchronized though.

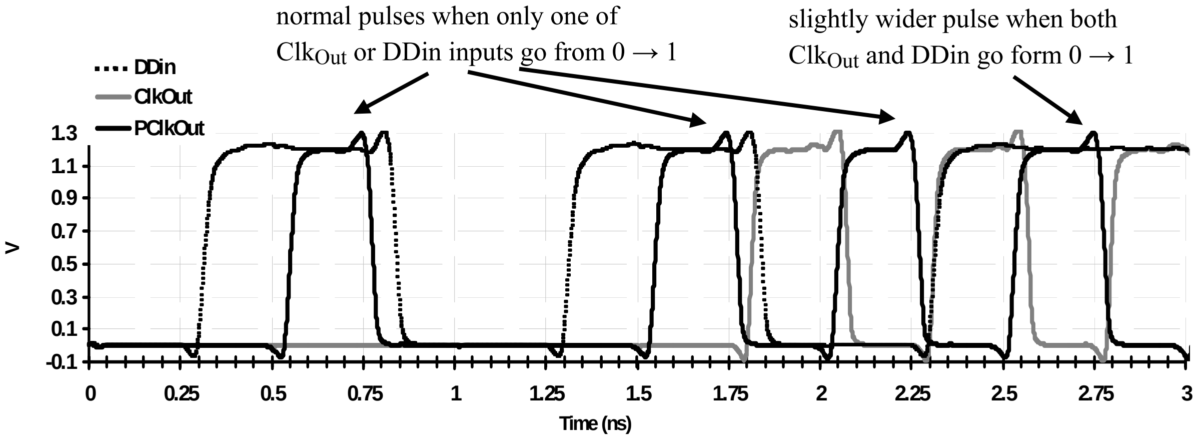


Figure 9. The input/output waveforms of the PED circuit. The output pulse (PClkOut) is slightly wider when both ClkOut and DDin inputs exhibit a simultaneous 0 → 1 transition.

The NED circuit

The **NED** circuit, shown in Figure 10, is similar to the **PED** except that it generates 2 complementary output pulses (T and Tb) only when the input data exhibits a negative transition. The circuit is designed such that T and Tb are truly differential signals (i.e. with simultaneous switching). Figure 11 shows the input (Din) and outputs of the **NED** circuit at 2GBps data rate. As the figure shows the output pulses T and Tb switch simultaneously. These pulses are used by the **PCCM** circuit to latch the appropriate phase(s) of the clock pulse (from CK0 to CKn). The matching delay matches the delay of PCCM and Schmitt trigger circuits.

The PCCM circuit

Figure 12 shows the circuit diagram of the period capturing and clock muxing circuit along with the simple Schmitt trigger used. The basic block is a pulsed flip-flop (**PFF**) whose inputs come from one of the identical delay lines and are triggered by the complementary pulses T and Tb. The output of each **PFF** controls a transmission gate.

When a clock pulse coincides with T and Tb, the **PFF** will latch a 1 and hence turn on the corresponding transmission gate which will connect the selected clock phase (from the second delay line) to the output. The delay lines are made of simple CMOS inverters.

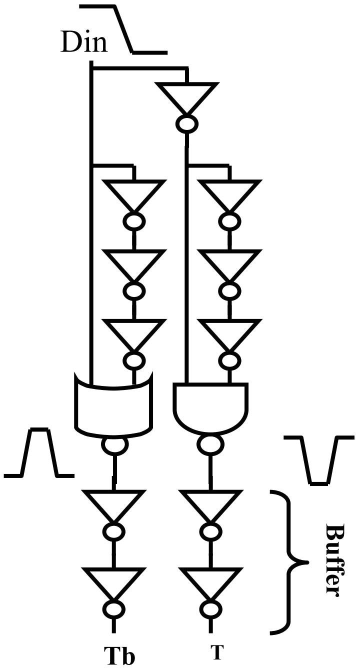


Figure 10. The negative-edge detector circuit.

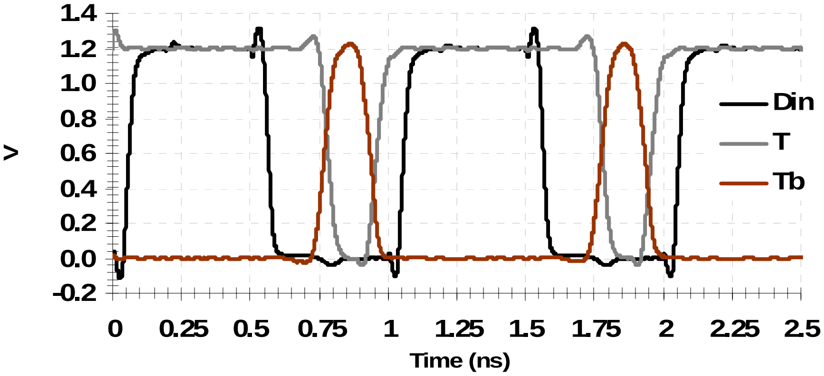
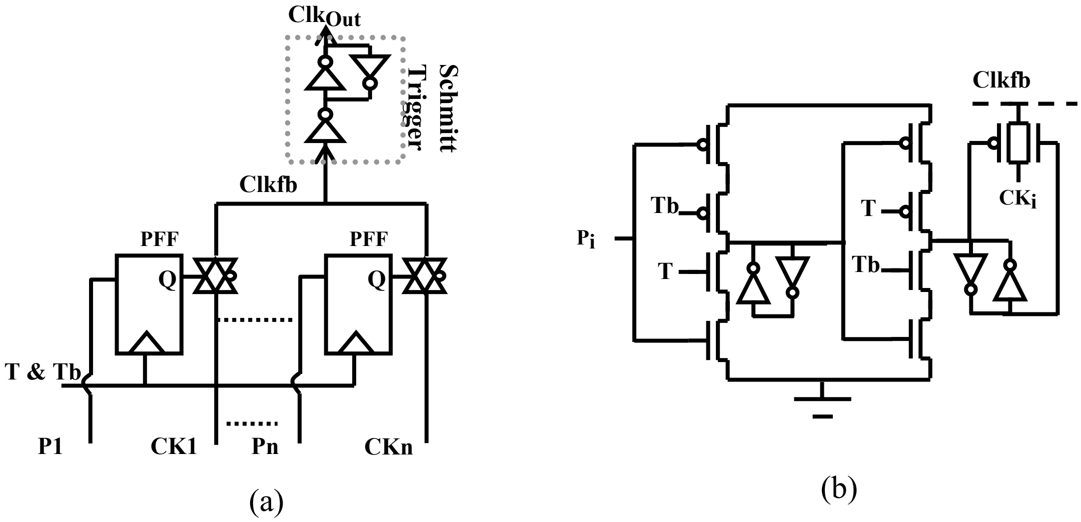


Figure 11. The NED input/output waveforms at 2GBps data rate.

The Matched Delay

The matched delay is made of an even number of inverters with a total delay equal to that of the **NED** circuit, the muxing circuit and the Schmitt trigger.

The total number of gates in the **CRC** is less than 100, most of which are inverters. This makes the circuit very compact allowing the instantiation of many copies for different serial links within the same chip.



(a)

(b)

Figure 12. (a) The PCCM Circuit's Schematic including the Schmitt Trigger and (b) The PFF schematic.

Performance Evaluation

Circuit simulations using T-Spice® and a 0.13µm, 1.2V CMOS technology were used to evaluate the operation and performance of the proposed CRC. Sizes of transistors in the circuit components were optimized for 2 GBPS operation, but it can still operate with any input frequency up to 2.5 GBPS. For lower frequencies the fixed delay in the PED circuit is increased keeping the PCCM circuit the same. First, the basic operation of the circuit is demonstrated by applying NRZ data to the input. Figure 13 below shows how the CRC captures the data frequency at 2 GBPS and generates an output clock after two bit transitions. The data pattern used was 1011111111.

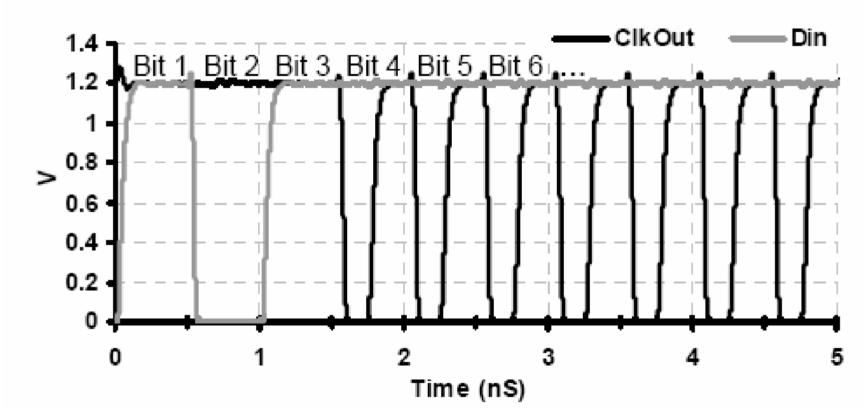


Figure 13. The input Data and output clock waveforms of the CRC at 2

Next, the response of the CRC to the injection of a large phase noise at the input data is evaluated. This is an important issue for CRCs which arises very often. Figure 14 shows how the circuit fully recovers the clock within two data transitions after a large phase noise (equivalent to a half bit cell) is injected into the input data. An analog PLL would have taken hundreds or even thousands of cycles to recover.

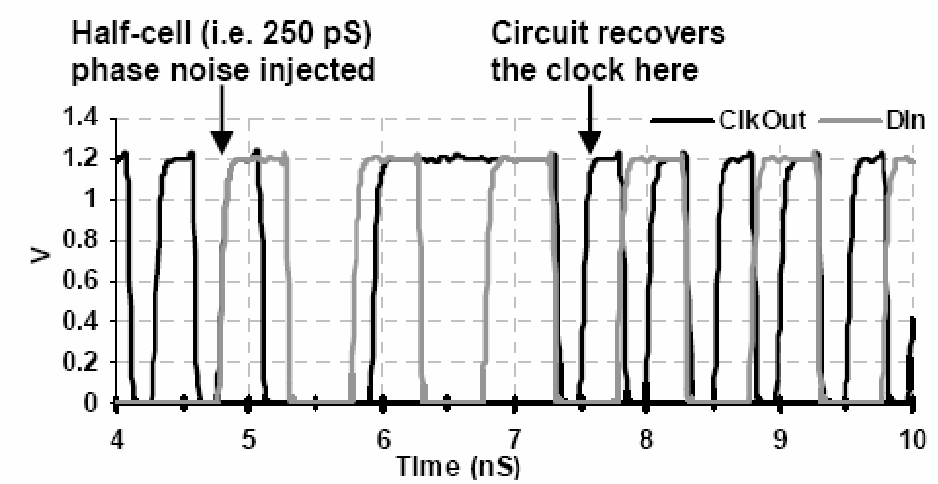
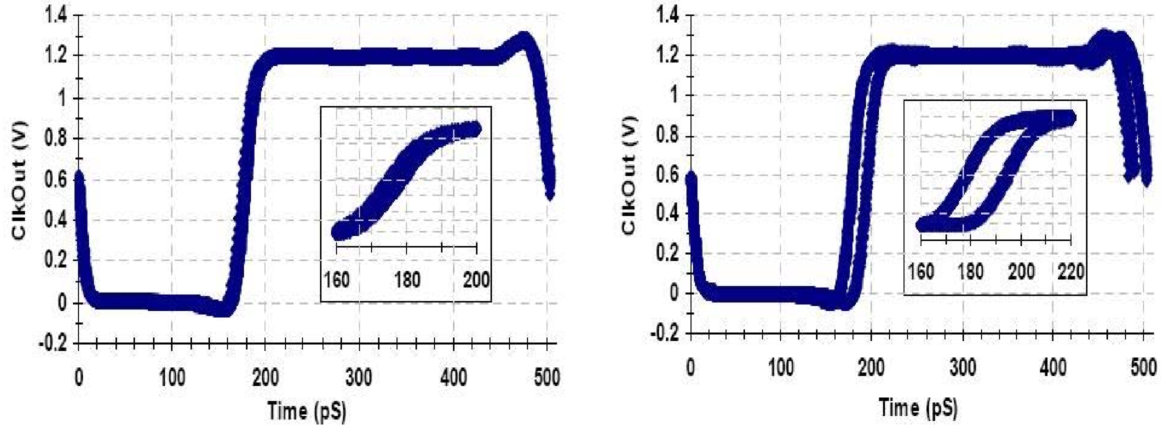


Figure 14. Output clock recovery after the injection of a large phase noise (half-cell) at 2 GBPS.

Another major concern of using a digital circuit with feedback in clock recovery might arise; that is the stability of the output frequency. To evaluate the stability of the output clock's frequency, two simulations were carried out; one with no data transitions after the initial clock frequency capturing (i.e. data stays constant after the two initial bit transitions) and another with a pseudo-random input stream (using 6-bit long pseudo-random patterns that are fed serially to the circuit). The output clock periods were laid out on top of one another in Figures 15 (a) and (b). These results were obtained with the improved version of the **PED** circuit, Figure 8(b), with the modifications that evens out the delays from the different inputs of the **PED** circuit to its output.

As this figure shows, the peak-to-peak period jitter in the case of no data transitions is less than 10 pS. This is an outstanding performance for an all-digital clock recovery circuitry. The results in Figure 15(b) show that, when there are data transitions, the output clock has two modes with about 10 pS difference in their periods. The peak-to-peak period jitter within each mode is still less than 10 pS. These two modes result from variations in the loop delay due to two modes of operation; when there is no positive data transition and when there is such transition. As Figure 9 shows, the delay of the **PED** circuit is slightly smaller when there is a positive input data transition (since in this case both inputs to the second NAND gate are going high). Without the improved **PED**, the two modes had about 50 pS difference between them. The improvements to the **PED** circuit reduced this number to 10 pS.

This bi-modal operation of the CRC, however, won't cause bit errors since the faster mode only occur when there is a positive input transition and the circuit immediately goes back to the normal mode when there is no data transitions. Hence there is no accumulation of phase error that might cause an error. It should be noted that the relative difference between the two modes is smaller for lower clock frequencies.



(a) Output Clock with no data transitions after the first two

(b) Output Clock with pseudo-random data

Figure 15. Output Clocks of the CRC (many cycles laid on top of one another) with original PED. The insets show a Zoom-in of the clock edge region.

Figure 16 shows the average output clock period for different data rates. The circuit components were only optimized once for the 2 GBPS data rate (0.5 nS bit cell). For other data rates, only the fixed delay part of the **PED** circuit was changed (by changing the number of inverters at its output). As the results in the figure indicate, the maximum frequency error is still below 0.8% for other data rates. The accumulated phase error due to the frequency error, however, is reset with every time a data transition occurs. This means that with proper data encoding that ensure adequate data transitions, the bit error rate can be set to a desired value.

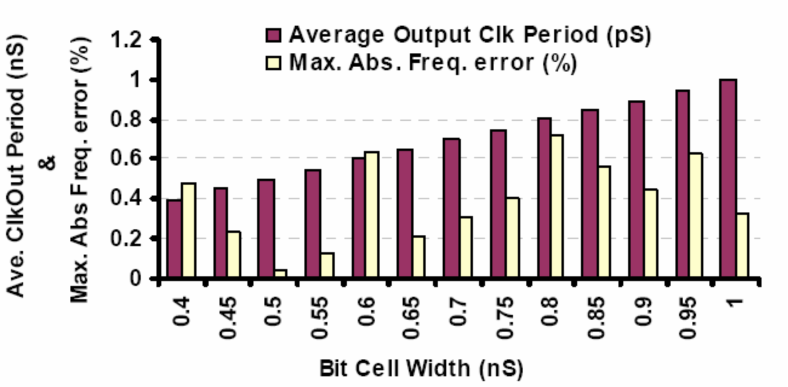


Figure 16. The average output clock Frequency and maximum absolute frequency error for different data rates.

4.2 Source Synchronous CRC

Since the previously developed had a very compact area, a second CRC was developed targeting on-chip source synchronous communications. The basic concept of the source-synchronous serial link (**S3L)** scheme was demonstrated in Figure 3. From, the transmitter side data is sent along with the transmission clock (**Clk**$_{\mathbf{T}}$). The FIFO on the transmitter side is synchronous (could be a simple single Flip-Flop). Both the transmitted data and clock are received by the **CRC** which re-times the clock with the data (**D**$_{\mathbf{Out}}$). This clock (**Clk**$_{\mathbf{W}}$) is used to write to the receiver FIFO. The CRC would also have a simple FSM to detect start bits and control writing to the FIFO. The asynchronous FIFO at the receiver side is required to facilitate data transfer between two clock domains (Transmitter and Receiver). The receiver reads data using its own clock (Clk$_{\mathbf{R}}$).

The architecture of this CRC is very similar to the off-chip version and is shown in Figure 17. The delay of the variable length digital delay line is adjusted such that a clock phase is aligned with the edge of the input data. The complement of this phase, which would then be in the middle of the bit cell (i.e. 90^o phase-shifted), is selected as the output clock. The delay line is made of two parts; a fixed delay (close to half a bit cell) and two matched delay lines that should have a total delay of at least one bit cell. The two matched delay lines carry two clock phases that are complement of one another (~ 180^o out of phase). The fixed delay was added to reduce the required stages in the variable delay line while maintaining the same resolution. Again all the different parts of the delay lines are all made up of identical inverters.

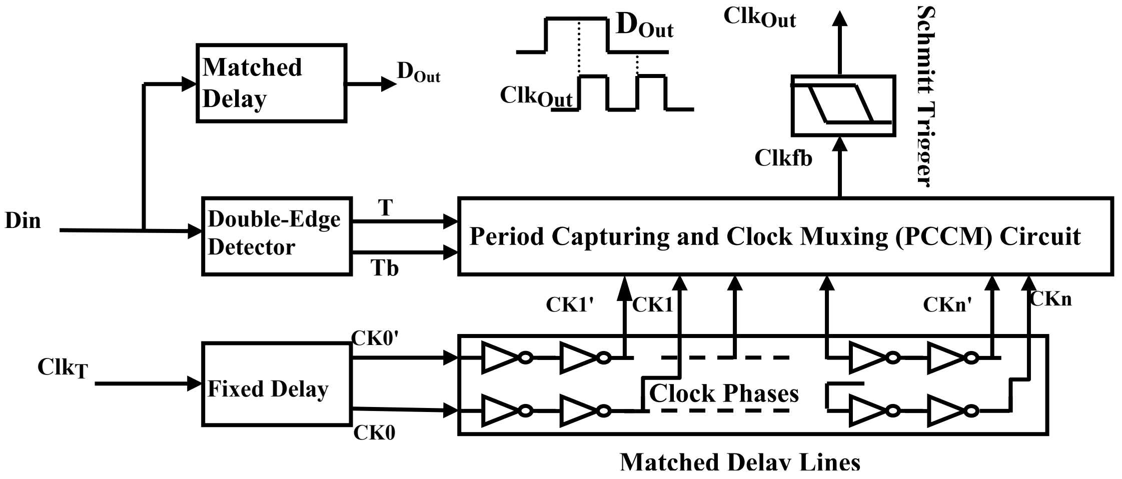


Figure 17. The architecture of the clock-recovery circuit for the on-chip S3L.

A double-edge detector (**DET**) circuit, shown in Figure 18, utilizes an XOR/XNOR to generate two complementary pulses each time there is an input data transition. This circuit could be implemented using a differential static XOR/XNOR circuit such as in [33]. However, the implementation shown in Figure 18 was chosen because it uses standard digital gates and produces complementary pulses (**T** and **Tb**) with equal delays. These pulses are used as trigger signals by the phase capturing and clock muxing (**PCCM**) circuit to capture the relative phase between the data and input clock. The **PCCM** circuit selects the appropriate phase(s) and output them to the Schmitt trigger to re-construct the clock signal.

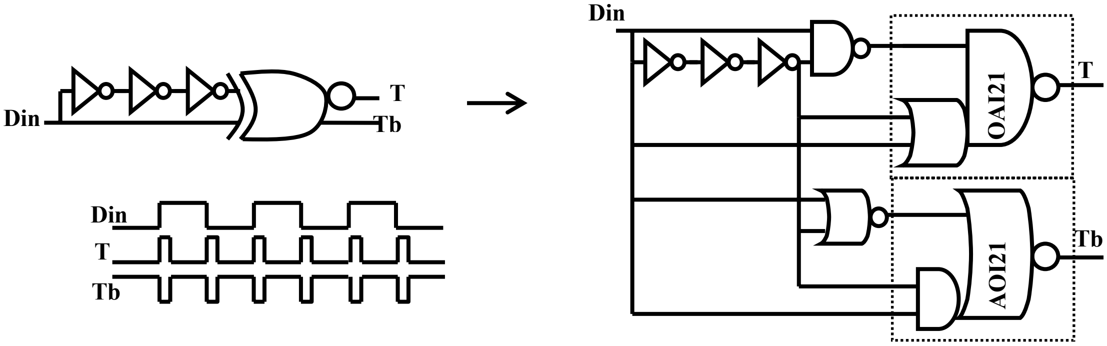


Figure 18. The Double-edge detector circuit implementation.

This CRC operates as follows;

- The fixed delay delays the incoming clock by about one half of a bit cell and produces two complementary phases which travel down the two matched delay lines,
- When an input transition occurs, the **DED** circuit generates the two complementary pulses, **T** and **Tb** that are applied as trigger signals to all the **PFF**s in the **PCCM** circuit,
- Each **PFF** captures the corresponding clock phase ($\mathbf{CK_i}$) at that instance. **PFF**s that capture the appropriate clock phase would then enable their muxes and pass the complement of the captured phase ($\mathbf{CK'_i}$),
- The Schmitt trigger reconstructs the output clock from the mixture of phases from all the enabled muxes. With more than one phase usually selected, the muxes in the **PCCM** circuit and the Schmitt trigger perform phase interpolation yielding higher resolution,
- The matched delay shown in Figure 12 is made up of even number of inverters that delays the input data (**Din**) by an equivalent delay of the **DED**, **PFF**, mux and Schmitt trigger. This makes the captured clock $\mathbf{Clk_{Out}}$ about 90^o phase-shifted with the data $\mathbf{D_{Out}}$. With each data transition the circuit re-times the clock with data, hence correct for any injected phase and/or frequency noise.

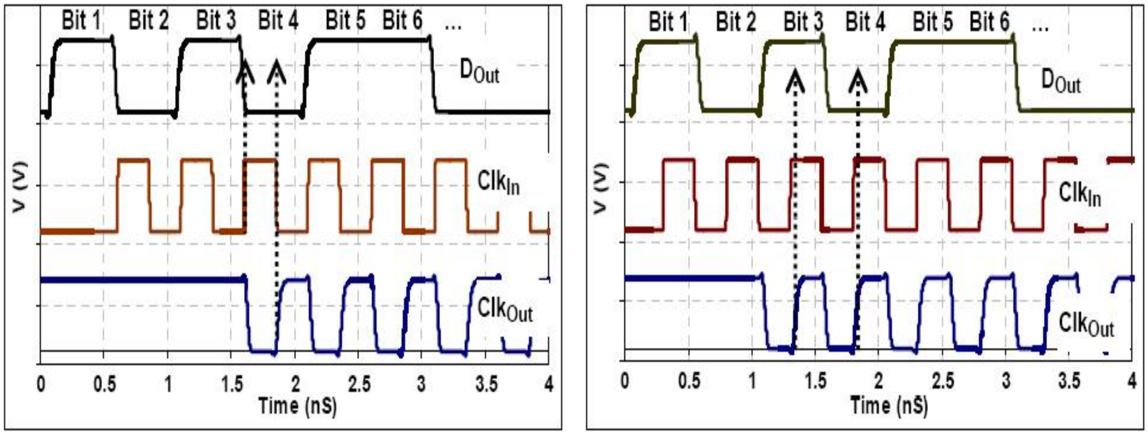
Again the total number of gates in this CRC is less than 100, most of which are inverters. This makes the circuit very compact allowing the instantiation of many copies for different serial links within the same chip.

Performance Evaluation

Again, circuit simulations using T-Spice® and a 0.13μm, 1.2V CMOS technology were used to evaluate the operation and performance of the proposed CRC circuit. Sizes of transistors in the circuit components were optimized for 2 GBPS operation, but it can still operate with any input frequency up to 2.5 GBPS. For lower frequencies, the fixed delay and the number of stages in the delay lines and PCCM circuit have to be increased. It should be noted that the data frequency is frozen at design time.

Figure 19 below shows how the CRC retimes the clock with Dout within 2 data transitions. Two scenarios were simulated in this figure; one with the input clock initially 85° phase-shifted with the data (i.e. 5° short of the required phase) and another with the input clock initially 0° phase-shifted with the data. In each case, the output clock is produced at the required phase.

The response of the circuit to a large phase noise (half a bit cell) injection into the input data was evaluated and the result is reported in Figure 20. The figure shows how the circuit first re-times the clock with the data with an initial -15° phase-shift between the input clock and data. The circuit then recovers the clock within one bit transition after a half-bit cell phase noise is injected into the data. An analog PLL would have taken hundreds or even thousands of cycles to recover.

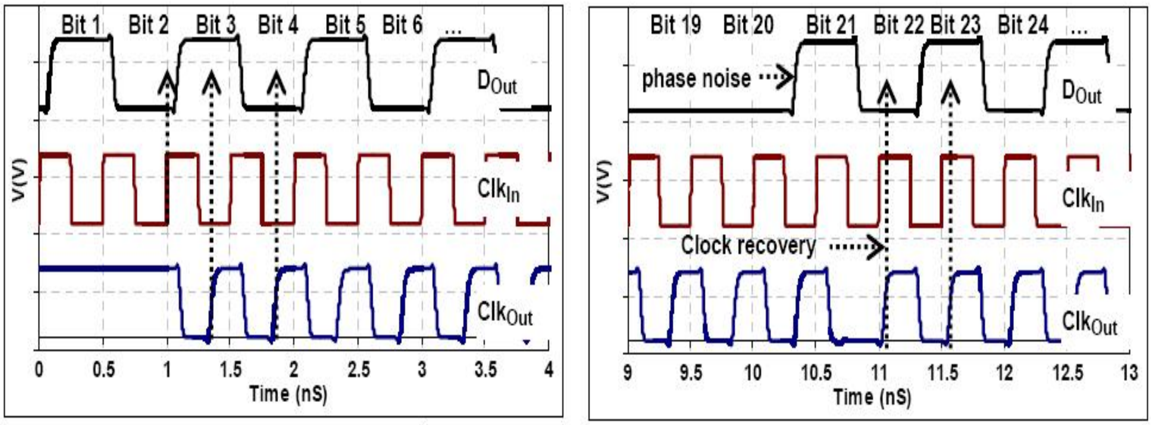


(a)

(b)

Figure 19. The input clock, output clock and data waveforms at 2 GBPS with two different phase shifts between the input clock and data; (a) 0° phase shift, (b) 85° phase shift.

To evaluate the stability of the output clock's frequency, simulations were carried out with a pseudo-random input stream (using 6-bit long pseudo-random patterns that are fed serially to the circuit). The output clock periods were laid out on top of one another and are plotted in Figure 21. As this figure shows, the peak-to-peak clock jitter is about 10 pS. It should be noted that this jitter would not accumulate since the CRC constantly re-times the clock with every data transition. Furthermore, the jitter is much smaller with no data transitions. This is an outstanding performance for an all-digital clock re-timing circuit.



(a) Initial clock re-timing with the input clock ~ −15° phase-shifted.

(b) Clock re-timing after an input phase noise is injected at Bit 21.

Figure 20. The CRC's response to a large phase noise injection (half a bit cell) in the input data.

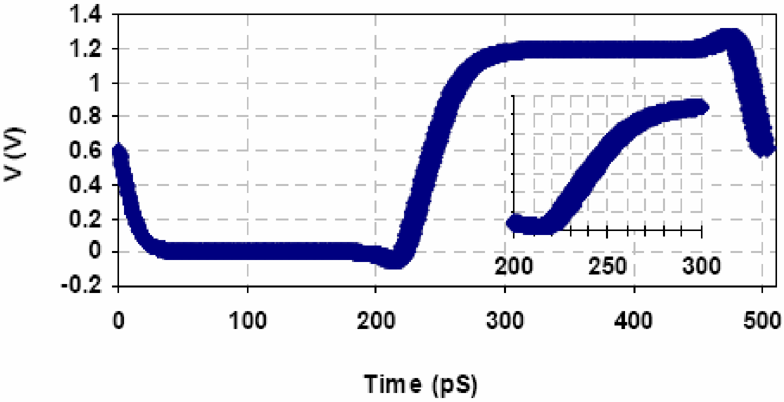


Figure 21. The output clock for a 6-bit long pseudo random input data. The inset shows the clock rising edge.

4.3 Investigation of Direct Digital Frequency Synthesis (DDFS) for Clock-Recovery

Many techniques were developed for synthesizing a sine wave or digital clocks with accurate frequency control for wireless applications [34-41]. The conceptual illustration of direct digital frequency synthesis (DDFS) is shown in Figure 22 below [34]. An n-bit accumulator is used to generate the phase data for the synthesized sine wave. It adds an external phase-specifying number N to its content every clock cycle. The resulting number, which represents the phase value of the sine wave, is used to access a ROM look-up table that stores the amplitude values. The m most significant bits of the accumulator output are used for accessing the ROM. DDFS usually utilize the inherent symmetry in the function to reduce the number of stored words in the ROM. Each amplitude value is represented by a k-bit word; hence the size of the LUT is **2^m x k** bits. A digital to analog converter (DAC) converts the k-bit words to analog values which are then filtered to smooth and remove some of the unwanted spurs (i.e. anti-aliasing). An optional comparator can be used to convert the output sine wave to a square (digital) wave. All blocks are controlled by an external clock with a frequency f_{Clock}. This clock could come from a PLL.

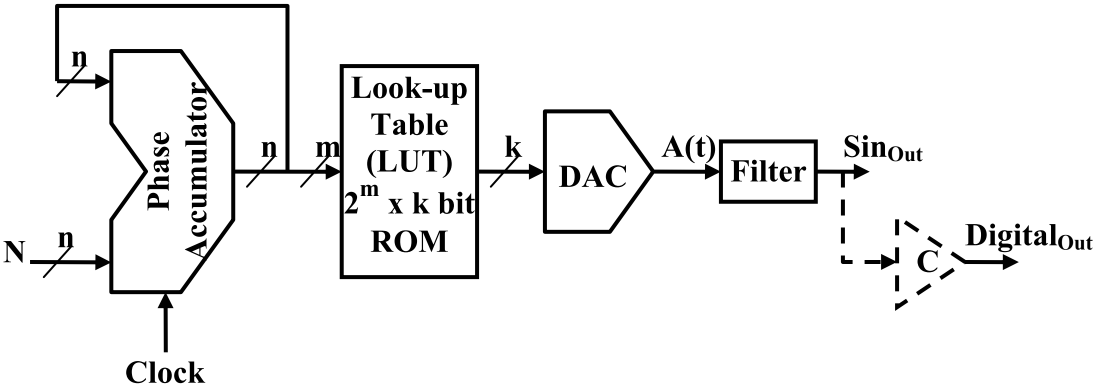


Figure 22. The basic architecture of a DDFS.

The period of the synthesized wave depends on the input N as:

$$T_{Out} = T_{Clock} \bullet \frac{2^n}{N}$$

Or the output frequency is:

$$f_{\text{Out}} = f_{Clock} \bullet \frac{N}{2^n}$$

Hence the maximum output frequency is ~ ½ f_{Clock}. The minimum period resolution is:

$$T_{\min} = \frac{T_{Clock}}{2^n}$$

This is also the maximum jitter due to the accumulator, i.e.:

$$J_{\max} = \frac{T_{Clock}}{2^n}$$

Thus, though the DDFS utilizes full digital blocks, it has the potential for synthesizing output frequencies with a resolution of less than an inverter's delay. Also phase transition from one frequency to another (i.e. when N is changed) is smooth and linear. These were the main reasons for investigating DDFS for clock recovery. The following limitations of DDFS have rendered it unsuitable for clock recovery:

1. In the case of clock frequency synthesis, this method requires a precise reference clock with at least double the maximum required frequency. This is a major limitation especially for the targeted high data rates. It should be noted that to keep the jitter (due to dithering) acceptable, the reference clock frequency has to be much larger than the maximum required frequency as indicated by the equations above,
2. Another alternative to having a precise reference clock was investigated. It involved using an on-chip ring inverter-based oscillator with the minimum possible number of stages to give the maximum possible reference frequency. The number of stages n_i is roughly:

   $$n_i \geq \max\{t_{rise}, t_{fall}\}/T_{Dave}$$

   Where t_{rise}, t_{fall} and T_{Dave} are the inverter's rise delay (time it take the output to rise from 10% to 90% of its maximum value), fall delay and average propagation delay. The worst case values of these parameters should be used for the technology at hand to ensure that the oscillator would sustain a complete (rail-to-rail) output clock.

   This means that the actual reference clock frequency will not be known, hence a feedback loop will be needed to adjust the n-bit phase-specifying number N at the input of the accumulator. Such a loop is demonstrated in Figure 23 below. A phase-frequency detector would detect the difference in phase/frequency between the generated clock and input data. A digital filter would then update N. The filter needs to be designed carefully to reduce both data-dependent jitter (jitter due to consecutive 0s or 1s) and jitter due to random input data jitter. This method still requires the reference clock to be much higher than the data frequency. Further more it requires significant design effort to ensure stability and proper jitter characteristics.

   For the technology used in this work (0.13 μm) the maximum reference frequency that can be reliably generated with an inverter-based ring oscillator was found to be less than 5 GHz, only 2.5x the targeted data frequency. The conclusion reached is that DDFS is clearly not suitable for clock recovery applications. Hence it was not investigated any further.

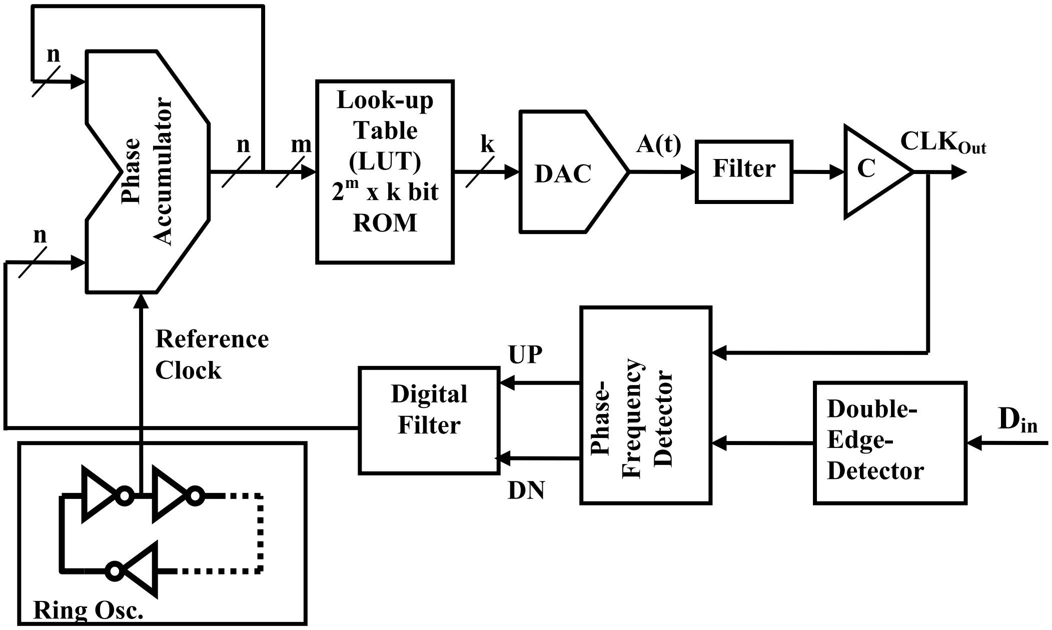


Figure 23. The basic architecture of a DDFS.

4.4 Implementing the Developed CRCs on an FPGA

One of the reviewers of the original proposal of this work had suggested prototyping the developed circuits using an FPGA board. After the development of the two CRCs (described in sections 4.1 and 4.2) this issue was investigated thoroughly. In fact many projects were conducted by PhD/Master students in a graduate course offered by the investigator to investigate the capabilities of FPGAs. Unfortunately, it turned out that the developed CRCs could not be implemented on FPGAs using the existing configurable logic blocks (CLBs). In all FPGAs, CLBs are implemented using look-up-tables (LUTs), typically with 4 inputs. That are connected using programmable interconnects. The developed CRCs require more fine-grained components (e.g. inverters). So when these CRCs are synthesized on an FPGA, one can not control pulse widths, or delay between different stages in the delay lines. In fact, multiple phases of the clock in the delay line can not be selected simultaneously, rendering phase interpolation useless. As a result, the developed CRCs could not be realized using FPGAs

5. Budget Utilization

A summary of the budget utilization versus allocated budget is shown in Table 2 below. A brief explanation of discrepancies between allocated budget items and expenditures is provided below. Many of these discrepancies resulted from two general reasons:

1. It took about 3 years from the time the proposal was submitted to the DSR (October 2002) till it was approved and allocated a budget (June 2005). Many of the things that were requested as part of the proposal were obtained through the department during that time,

2. Since principal investigator was new at KFUPM many of the budget items (such as secretary wages) were suggested by colleagues in the department. Latter on it turned out that some of these items were not really needed.

Discrepancies in manpower budget:

- The budgeted amount for graduate students was not fully utilized because of the difficulty of finding interested students. Although the work had been advertised among graduate students in Computer Engineering and Electrical Engineering departments, no student showed any interest. One graduate student showed some interest for a short time but then ceased to work on the project. It seems that graduate students simply do not like design work and prefer to write some simulation code for their research thesis.

- The budget amount for secretary exceeded the amount of work carried by secretaries. With the automation of many services and on-line submission of publications, there is very little work for a secretary.

- Though no fund was budgeted for a technician, the help of one was needed. As such SR 500 was re-allocated from the secretary item. Again this need arose after the department PC technician left and no replacement was hired.

Discrepancies in Equipments budget: FPGA Board

One of the proposal reviewers had suggested prototyping the developed circuits using an FPGA board. As a result SR 10,000 was allocated for an FPGA board with the provision (by the principle investigator) that it will not be purchased unless it is deemed feasible to implement the developed circuits on it. Since it was determined that the developed circuit can not be implemented on FPGAs (section 4.4), the board was not purchased.

Discrepancies in Software budget

At the time of writing the proposal, in 2002, a price quote for the required software was obtained for SR 13,000. By the time the actual purchase was carried out, the price has gone up to SR 17,700.

Difficulties in utilizing the allocated budget

The requirement imposed by the DSR to purchase all computer equipments and software through the ITC has proved to be very time and labor intensive. It took over 6 months to obtain such items, which hampers the research greatly. The investigator had to do most of the work himself anyway, but pushing others and making sure they do what they are suppose to do was a major additional effort.

Table 2: Summary of budget expenditures as of June 2007.

Item	Budgeted	Expended
Man Power:		
Principal Investigator	SR 28,800	SR 28,800
Graduate Students	SR 14,400	SR 500
Secretary	SR 2,000	SR 1,000
Technician	--	SR 500
Equipments:		
1. Desktop PC and printer	SR 7,500	SR 7,500
2. FPGA prototyping board	SR 10,000	--
Software: Tanner Tools	SR 13,000	SR 17,800
Stationary and Miscellaneous	SR 1,500	SR 202
Books and References (through library)	SR 2,000	SR 2,000
Conference Attendance	SR 10,000	SR 7,800
Total Project Cost[1]	**SR 89,200**	**SR 68,102**

[1] SR 100,000/- = US $ 26,600/-

References

[1] N. P. Carter and A. Hussain, “Modeling wire delay, area, power and performance in a simulation infrastructure,” IBM Journal of Research and Development, Vol. 50, number 2/3, p. 311, March 2006.

[2] International Technology Road Map For Semiconductors, ITRS, 2003 Edition. Semiconductors Industry Association 2006, http://www.itrs.net/Links/2006Update/2006UpdateFinal.htm.

[3] L. Benini and G. D. Micheli, “Networks on chips: A new SoC paradigm”, *IEEE Computer*, Vol. 35-1, pp. 70 – 78, January 2002.

[4] Daniel M. Chapiro, “Globally-Asynchronous Locally-Synchronous Systems,” PhD thesis, Stanford University, October 1984.

[5] T. Chelcea and S. M. Nowick., “ A low-latency FIFO for mixed-clock systems,” *Proc. IEE Computer Society Annual Workshop on VLSI (WVLSI’00)*, April 2000.

[6] Simon Moore et al., “ Point to Point GALS Interconnect,” *Proceedings of the Eighth International Symposium on Asynchronous Circuits and Systems (ASYNC.02),* 2002.

[7] A. Chakraborty and M. R. Greenstreet, “A minimal source-synchronous interface,” *Proc. 15th Annu. IEEE Int. ASIC/SOC Conf.*, pp. 443–447, 2002.

[8] A Chakraborty and M. R. Greenstreet, “Efficient self-timed interfaces for crossing clock domains,” *Proc. 9th Int. Symp. Asynchronous Circuits and Systems*, pp. 78–88, 2003.

[9] B. Razavi, Ed. "Monolithic phase-locked loops and Clock Recovery Circuits," IEEE Press, 1996.

[10] B. Razavi, “A 2.5-Gb/s 15-mW Clock Recovery Circuit, ” *IEEE Journal of Solid-State Circuits*, vol. 31, pp. 472-480, Apr. 1996.

[11] S. Anand and B. Razavi, “A CMOS Clock Recovery Circuit for 2.5-Gb/s NRZ Data,” *IEEE Journal of Solid-State Circuits*, vol. 36, pp. 432-439, Mar. 2001.

[12] T-Y. Hsu et al., “An All-Digital Phase-Locked Loop (ADPLL)-Based Clock Recovery Circuit,” IEEE JSSC, Vol. 34-8, pp. 1063-1073, 1999.

[13] D-L. Chen, “A Power and Area Efficient CMOS Clock/Data Recovery Circuit for High-Speed Serial Interfaces,” *IEEE Journal of Solid-State Circuits*, vol. 31, pp. 1170-1176, August 1996.

[14] P. Larson, “A 2-1600 MHz 1.2-2.5V CMOS Clock-Recovery PLL with Feedback Phase-Selection and Averaging Phase-Interpolation for Jitter Reduction,” *in Proc. of the 1999 IEEE International Solid-State Circuits Conference*, pp.210-211, 1999.

[15] Dao-Long Chen, “A Power and Area Efficient CMOS Clock/Data Recovery Circuit for High-Speed Serial Interfaces,” *IEEE Journal of Solid-State Circuits*, vol. 31, pp. 1070-1076, August 1996.

[16] J-K Kang, W Liu, and R. K. Cavin, III, “A CMOS High-Speed Data Recovery Circuit Using the Matched Delay Sampling Technique,” *IEEE Journal of Solid-State Circuits*, vol. 32, pp. 1588-1596, Oct. 1997.

[17] C-K K. Yang, R. Farjad-Rad, and M. A. Horowitz, “A 0.5-m CMOS 4.0-Gbit/s Serial Link Transceiver with Data Recovery Using Oversampling,” *IEEE Journal of Solid-State Circuits*, vol. 33, pp. 713-721, May 1998.

[18] R. L. Aguiar and M. Figueiredo, “Resource Constrained Clock Recovery on Programmable Logic Devices,,” *in Proc. of the 9th Int. Conf. on Electronics, Circuits and Systems (ICECS)*, pp.1111-1114, Sept. 2002.

[19] J-K Kang, “Performance Analysis of Oversampling Data Recovery Circuit,” *Institute of Electronics, Information and Communication Engineers (Japan) Transactions on Fundamentals*, vol. E82-A, No. 6, pp..958-964, June 1999.

[20] T-Y. Hsu et al., “An All-Digital Phase-Locked Loop (ADPLL)-Based Clock Recovery Circuit,” IEEE JSSC, Vol. 34-8, pp. 1063-1073, 1999.

[21] D-L. Chen, “A Power and Area Efficient CMOS Clock/Data Recovery Circuit for High-Speed Serial Interfaces,” IEEE Journal of Solid-State Circuits, Vol. 31, No. 8, pp. 1170-1176, August 1996.

[22] T. Saeki *et al*., “A 1.3-Cycle Lock Time, Non-PLL/DLL Clock Multiplier Based on Direct Clock Cycle Interpolation for Clock on Demand,” IEEE Journal of Solid-State Circuits, Vol. 35, No. 11, pp. 1581-1590, November 2000.

[23] M. Combes et al., “A Portable Clock Multiplier Generator Using Digital CMOS Standard Cells,” IEEE JSSC, Vol. 31-7, pp. 958-965, July 1996.

[24] M-J Lee, et al., “A Second-Order Semi-Digital Clock Recovery Circuit Based on Injection Locking,” *Proc. IEEE ISSCC’03*, 2003.

[25] H. Takauchi, et al., "A CMOS multichannel 10-GB/s transceiver," IEEE JSSC, Vol. 38-12, pp.2094-2100, December 2003.

[26] R. Farjad-Rad, et al., "1 33-mW 8-GB/s CMOS clock multiplier and CDR for highly integrated I/Os," IEEE JSSC, Vol. 39-9, pp. 1553-1561, September 2004.

[27] K. Krishna, et al., "A 0.6 to 9.6 GB/s Binary Backplane Transceiver Core in 0.13 μm CMOS," *Proc. IEEE ISSC'05*, pp. 65-66.

[28] J. Parker, et al., "A 15 mW, 3.125 GHz PLL for Serial Backplane Transceivers in 0.13 μm CMOS," *Proc. IEEE ISSC'05*, pp. 412-413.

[29] J-K Kang, W Liu, and R. K. Cavin, III, “A CMOS High-Speed Data Recovery Circuit Using the Matched Delay Sampling Technique,” *IEEE Journal of Solid-State Circuits*, vol. 32, pp. 1588-1596, Oct. 1997.

[30] C-K K. Yang, R. Farjad-Rad, and M. A. Horowitz, “A 0.5-m CMOS 4.0-Gbit/s Serial Link Transceiver with Data Recovery Using Oversampling,” *IEEE Journal of Solid-State Circuits*, vol. 33, pp. 713-721, May 1998.

[31] R. L. Aguiar and M. Figueiredo, “Resource Constrained Clock Recovery on Programmable Logic Devices,,” *in Proc. of the 9th Int. Conf. on Electronics, Circuits and Systems (ICECS)*, pp.1111-1114, Sept. 2002.

[32] J-K Kang, “Performance Analysis of Oversampling Data Recovery Circuit,” *Institute of Electronics, Information and Communication Engineers (Japan) Transactions on Fundamentals*, vol. E82-A, No. 6, pp..958-964, June 1999.

[33] M. E. S. Elrabaa, "A New Static Differential CMOS Logic with Superior Low Power Performance," Analog Integrated Circuits and Signal Processing, Vol. 43, No. 2, pp. 183-190, May 2005.

[34] B. Goldberg, *Digital Techniques in Frequency Synthesis*, 1996, McGrow-Hill, New York.

[35] D. Sunderland, *et. al*., “CMOS/SOS frequency synthesizer LSI circuit for Spread Spectrum communications,” IEEE Journal of Solid-State Circuits, Vol. SC-19, pp. 497-505, Aug. 1984.

[36] H. T. Nicholas III and H. Samueli, “A 150-MHz direct digital frequency synthesizer in 1.25-µm CMOS with -90dBc spurious performance,” IEEE Journal of Solid-State Circuits, Vol. 27, pp. 1959-1969, Dec. 1991.

[37] H. Nicholas, and H. Samueli, “The optimization of direct digital frequency synthesizer performance in the presence of finite word length effects,” in *42nd Annual Frequency Control Symp.*, 1988, pp.357-363.

[38] L. K. Tan, H. Samueli, and B. Kim, “A 200 MHz quadrature digital synthesizer/mixer in 0.8 µm CMOS,” IEEE Journal of Solid-State Circuits, Vol. 30, pp. 193-200, Mar. 1995.

[39] A. Bellaouar, *et. al*., “Low-power direct digital frequency synthesis for Wireless Communications,” IEEE Journal of Solid-State Circuits, Vol. 35, pp.385-390, Mar. 2000.

[40] H. Nosaka, *et. al*., “A Low-power direct digital frequency synthesizer using a self-adjusting phase-interpolation technique,” Journal of Solid-State Circuits, Vol. 36, pp.1281-1285, Aug. 2001.

[41] D. Calbaza, and Y. Savaria, “A Direct digital period synthesis circuit,” Journal of Solid-State Circuits, Vol. 37, pp.1039-1045, Aug. 2002.

Appendix A:

Publications Resulting from the Project

The following publications (click on any paper title to view the paper) resulted from this research project:

1. Muhammad E. S. Elrabaa, "Portable Clock Recovery Circuits (CRCs) For On-Chip and Off-Chip Serial Data Communication,” Accepted for publication in the Arabian Journal for Science and Engineering (AJSE).
2. Muhammad E. S. Elrabaa, “An All-Digital Clock Frequency Caputring Circuitry For NRZ Data Communications,” *Proc.13th IEEE International Conference on Electronics, Circuits and Systems* (ICECS 2006), Dec. 2006, Nice, France
3. Muhammad E. S. Elrabaa, “A Portable Clock Recovery Circuit (CRC) For Systems-On-Chip Serial Data Communication,” *Proc. International Conference on Microelectronics* (ICM06), Dec. 2006, Dhahran.
4. Muhammad E. S. Elrabaa, “A Digital Clock Re-Timing Circuit for On-Chip Source-Synchronous Serial Links,” *Proc. International Conference on Microelectronics* (ICM06), Dec. 2006, Dhahran.