High-Speed Clock Recovery for Low-Cost FPGAs

István Haller
Computer Science Department
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
haller@student.utcluj.ro

Zoltan Francisc Baruch
Computer Science Department
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
Zoltan.Baruch@cs.utcluj.ro

***Abstract*—High speed serial interfaces represent the new trend for device-to-device communication. These systems require clock recovery modules to avoid clock forwarding. In this paper we present a high-speed clock recovery method usable with low-cost FPGAs. Our proposed solution features increased speed and reduced size compared to existing designs. The method allows a maximum throughput of 400Mbps compared to the vendor supplied solution capable of only 160Mbps. The module was also integrated and tested within a serial transceiver system. Although the implementation is specific to a given vendor, the idea can also be applied to others devices because it uses only generally available components from most vendors.**

Keywords-clock recovery, serial communication, FPGA

I. Introduction

An important issue for embedded system designers is the task of high speed device-to-device communication. Traditionally synchronization between the devices was ensured by the use of a common clock for both the transmitter and the receiver modules. This required either a common clock source, or the transmission of a synchronization signal along with the data. The first solution is called a system-synchronous setup and it has the problem of requiring the designer to take into account the delays of both devices for the transmission. The second method is called a source-synchronous setup. Its advantage is simplicity and ease of use. The disadvantages of this method are due to the extra transmission lines required. The clock line also has more stringent constraints than the data line concerning noise, and the phase difference between the two lines is required to be kept at a minimum to ensure synchronization. Due to these problems operating frequency is limited and the production costs are increased.

By embedding the clock signal inside the data line the number of transmission lines can be reduced and the synchronization ensured while providing greater immunity to noise[1]. This technique has allowed the creation of the multi-gigabit serial communication protocols but can also help reduce power consumption and cost in embedded systems where device-to-device bandwidth is important.

The task to perform the extraction of the synchronization information from the data line is called clock and data recovery. It represents the most critical task in modern high performance serial communication systems as its capabilities limit the transmission bandwidth.

Literature presents three main types of clock and data recovery methods: PLL-based[2], optical[3], and over-sampling based[4]. PLL based methods use a phase locked loop controlled by a phase detector which is sampling the input sequence. Frequent changes are required for the lock to be achieved, but these are guaranteed by the communication protocols. The method is mainly dependent on the phase detection methodology. Currently published phase detection methods require two samples for every bit of data[5-7]. The disadvantage of this solution is the requirement of a controllable oscillator. Optical clock recovery based methods are a variation to the basic PLL based method, but they are applied directly on the optical stream. Although this method can operate at very high speeds, it can only be used over optical-fiber transmission. Over-sampling based methods use a higher clock frequency to sample the signal in multiple position. This method doesn't recover the clock signal, but it is able to recover the data. The usual over-sampling rate is 4-5 for high speed circuits and 3 for circuits used in gigabit transmission. The method is simple and robust, but it requires a higher frequency clock. This method is usually used in programmable devices when no dedicated hardware blocks are available, due to the fully digital design.

Currently two types of implementations are available for FPGAs. High-end models implement clock recovery in dedicated hardware components part of configurable multi-gigabit transceivers[1]. Low-end models have no such hardware blocks, but the vendors offer reference designs using on-board logic[4]. These designs use a 4-5X oversampling method to data recovery. Our proposed solution is a novel method to implement a PLL based clock recovery method inside an FPGA using only integrated components. By using the user controllable phase shift feature of the clock managers available also on low-cost FPGAs, we are able to provide clock recovery at twice the speed of the existing reference designs. Generally the clock recovery feature is provided by PLL devices, but it is not the case for FPGAs. Thus we were required to implement a user defined PLL digital loop around the existing components. The implementation is based on Xilinx technologies, but similar technologies are available also from the other major FPGA vendors such as Altera, Actel and Lattice.

978-3-9810801-6-2/DATE10 © 2010 EDAA

II. Related work

As presented previously, the vendors recommend oversampling based methods for low-cost FPGAs. We have selected the reference design proposed by Xilinx[4] due to our available hardware. The method uses simple oversampling based on 4 clock phases. The design requires 21 flip-flops and approximately 25 LUTs. The combinational logic requirements were approximated based on the number of inputs required for each separate combinational line. The architecture also requires multiple logic levels between registers, reducing the maximum operating frequency to 160MHz as noted in the documentation. As the figure was supplied by the vendor itself, we will take it as the reference maximum operating speed. Currently no faster method was found to be available for this family of chips.

III. Clock recovery loop architecture

The architecture is a basic feedback loop containing a sensor and an actuator. The phase detecting block acts as a sensor for the current setup, while the oscillator control module modifies the phase of the clock signal (Fig.1). The main difference from the classic PLL is that the system doesn't act on the oscillator itself, but it acts on the clock manager placed between the oscillator and the system. As a result the control interface is simplified and the other blocks from the loop are not required anymore.

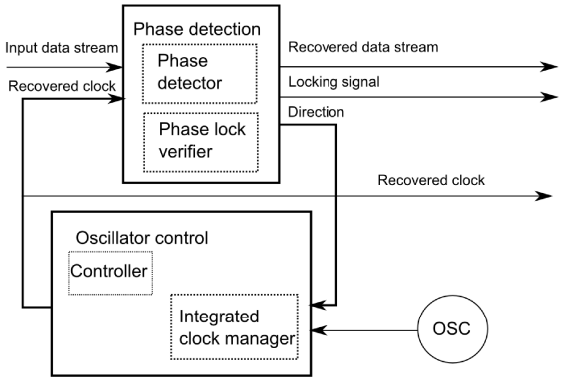

Figure 1. Loop architecture

A. Oscillator control

The novelty of our approach is the use of dynamic oscillator control using only integrated components. Both major FPGA vendors include a dynamic phase shift possibility in their clock manager. This feature is available in both the low-cost and high-end series, but it is not commonly used because it is not required in traditional designs. It has a maximum resolution of 20-100ps depending on model and operating frequency. For our research we have selected to use a Spartan3E chip, a low-cost solution from Xilinx. This chip features a dynamic phase shift resolution of 20-30ps independent of operating frequency. As all Xilinx FPGAs this model also uses a delay-locked-loop instead of a PLL for clock management. This has the advantage of a simple phase model based on time-steps. The phase shift also affects all of the clock manager outputs, thus enabling phase shifted quadrature clocks using a single component[8].

The phase adjustment operation requires multiple clock cycles for the clock manager. The amount is not know in advance, but the end of the operation is signaled on a dedicated output. A controller was designed to handle with this issue. A reduced complexity is preferred for this component for increased operating frequency. The implementation is a Mealy automaton, Fig. 2, with two states, one for sampling the phase detector and one for waiting after the clock manager. Sampling is performed into registers controlled by the automaton to ensure that no residual data remains from the transition phase. The final structure requires a total of 4 flip-flops and 5 LUTs.

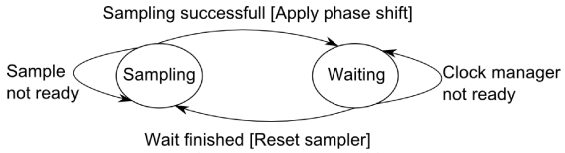

Figure 2. State machine

B. Phase detection

The phase-detection component provides the feed-back in the locking loop, thus its performance directly affects the clock recovery. Traditionally analog phase-detectors, are used which generate a command with the amplitude directly correlated to the measured phase-shift. This allows for a faster locking mechanism and a smoother operation once locking has been achieved[2]. The method requires analog components which are unavailable on FPGAs. Due to the technology constraints, we were required to use a fully digital phase detector. These are also called bang-bang type phase detectors as they only provide information on direction, but not on amplitude. Consequently this method can never provide perfect alignment, the system will operate with a continuous jitter around the phase lock.

For our work we have selected a slight variation[6] of the basic Alexander phase detector[5] using only half the sampling frequency. This method is called a half-rate binary phase detector and it enables support for double-data-rate transmission. It uses quadrature clocks for sampling the data line in three positions to gather data about the transitions on the data line. Note that although the oversampling methods presented in reference designs also use quadrature clocks, they are capable of providing only single-data-rate data recovery, thus having half the bandwidth of our approach. The phase detector is robust and simple requiring only three flip-flops and two XOR gates. The transition diagram is presented on Fig. 3. The main disadvantage of this method is that locking is achieved when the data line is phase locked with the second sampling edge. This violates the setup and hold times of the second register leading to metastability. As suggested by Xilinx literature[4] we are using a second row of registers to buffer the data and to limit the load on the metastable register. The structure of the detector can be seen on Fig. 4. The direct register-to-register transfer allows us to bring the data from the falling edge of the clock to the rising edge using a single step. Experiments prove that this solution is adequate and the system operates as required. The data recovery function is performed by the first and third registers which sample the two bits of data when the system is locked.

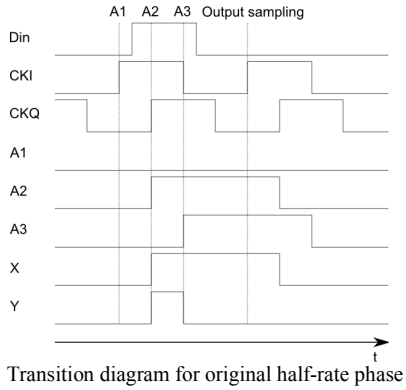

Figure 3. Transition diagram for original half-rate phase detector

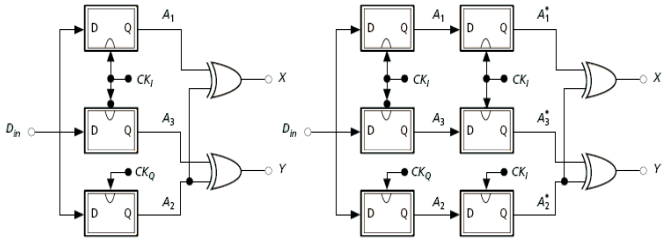

Figure 4. Original half-rate phase detector and new structure

C. Phase lock detection

Besides detecting the phase shift, an additional control component is required in the feed-back loop to signal phase lock. In our solution we have chosen to use a condition based on the recent phase-difference direction. This approach is searching for changes in direction which indicates that the second edge has passed over the data line edge. Using an ideal flip-flop model the transition suggests that the perfect locking position is between the current and the previous setup. In the real case setup and hold times are violated and the results are not predictable. We consider that the first and third registers are able to read accurate data when the timing constraints are violated for the middle register. This model is accurate as the setup and hold times are not overlapping for the three registers. Due to the bang-bang behavior of our detector the middle register should be kept in the non-deterministic region and its output should not be constant while locking is maintained. Currently the phase correction is maintained also after locking was achieved in order to have instantaneous response to temporary disturbances.

During our experiments we have discovered that a history of 16 consecutive states is required from the phase detector due to the real register behavior. This can be handled either by a shift register or a counter with custom control circuitry. The first method has the advantage of simplicity, requiring only direct register-to-register transfers and two logic levels for detecting the monotonous patterns. It requires a total of 16 flip-flops and 5 LUTs. The second method reduces size while using more complicated logic. It uses a 4bit counter and a state bit. The state stores the previous phase-difference and the counter counts the number of occurrences. The counter stops at 15 and is reset only on a change of direction. This method has the same behavior as the shift-register based approach while requiring only 5 flip-flops and 5 LUTs. The combinational paths are also reduced to a single level between registers. Both methods require an additional status bit and logic gate to handle initialization.

If continuous operation is trusted and the system is required to be simplified than a basic edge detector can be used to check for the first change of direction. This method requires only two flip-flops and a single LUT. This component will not affect the system's capability to perform clock recovery, just the possibility to detect and signal sudden changes due to which locking is temporarily lost.

IV. Results

A. Experimental setup

For our experiments we are using a Spartan3E-500 chip, a medium capacity low-cost FPGA from Xilinx. The architecture is based on 4 input look-up tables and single-bit configurable flip-flops/latches. The chip also includes 4 digital clock managers, each of which is capable of providing dynamic phase shift with a resolution of 20ps independent of the operating frequency. The device also provides LVDS and DDR support for high speed inter-chip communication. The logic capacity of the chip is of 9312 flip-flops and an equal number of LUTs. This platform was used to test and compare our solution with the reference design suggested by the vendor. The tests used DDR transmission at the highest frequencies possible for the system. The receiver and the transmitter were implemented on a single chip using an external loop-back for transmission.

B. Comparison of methods

The solution presented in this paper is also using the same 4 phases, but due to its half-rate design it is capable of recovering two bits of information for every clock cycle. Thus the transmission rate is double compared to the reference design when the operating frequencies are equal. Our design is also smaller, requiring only 16 flip-flops, 13 LUTs. Both methods require the additional clock manager for multiple-phase clocking thus no overhead is added for the dynamic phase shift. The difference is more significant for the combinational logic which helps reduce delay times in the design. The design also uses a single combinational level between successive registers, increasing maximum operating frequency. By using the Xilinx tool-chain the reported maximum frequency was 200MHz. As a result the transmission rate was increased by a factor of 2.5 compared to the reference design. The main disadvantage of our method is that it requires an initialization time. While the reference design is able to immediately locks onto the data stream, our method requires an initial locking procedure. As the phase shift can only be modified in increments of 20ps our method may require a large number of steps before phase-lock is achieved. Our test case used a clock frequency of 100MHz and a data line of 200Mbits with the external loop-back. The initialization phase required approximately 125 steps compared to 250, the maximum number possible for this setup. During experiments the total number of cycles required to performs these steps was on average around 500, with rare peaks of around 1000 cycles. For 100MHz these numbers represent a time-delay of 5us, respectively 10us. As the frequency

increases, the number of cycles required will decrease due to the fixed phase shift step. This step is required every time communication is stopped and restarted. This is due to the possible desynchronization of the transmitter and receiver. It is also a problem PLL based systems and it is usually avoided by transmitting idle signals when no data is present. As a result our method is recommended mainly for systems requiring a high continuous bandwidth while the reference design is better suited for small bursts of data. The results are summarized in Table 1.

TABLE I. COMPARISON OF SOLUTIONS

	FF	LUT	Frequency	Data-rate	Initialization
Xilinx	21	25	160MHz	160Mbps	none
Proposed	16	13	200MHz	400Mbps	5-10us

C. System integration

The module presented in this paper is not just a stand-alone component, it was designed as part of a high speed serial transceiver. The system is simplified implementation of the Gigabit-Ethernet protocol. It uses double-data-rate LVDS signaling to achieve maximum speed. The main limitation of system bandwidth was due to the clock recovery method. Thanks to our newly developed solution we were able to achieve a theoretical maximum transmission speed of 400Mbps instead of 160Mbps using the vendor proposed design. As a result the transmission throughput increased by a factor of 2.5. The transceiver uses 8b/10b encryption to ensure the requirements for the data line behavior are met. This encryption provides only 80% real data for the bandwidth, but it also provides numerous control symbols. Thus real bandwidth is reduced to 360Mbps. The system also features word-alignment based on the same control symbols as Gigabit-Ethernet. The symbol features a unique sequence of bits required to locate the word boundaries. Finally buffers are included both for the transmitter and the receiver to handle the different clocking domains and temporary data storage. A basic block-diagram of the system is presented on Fig. 5.

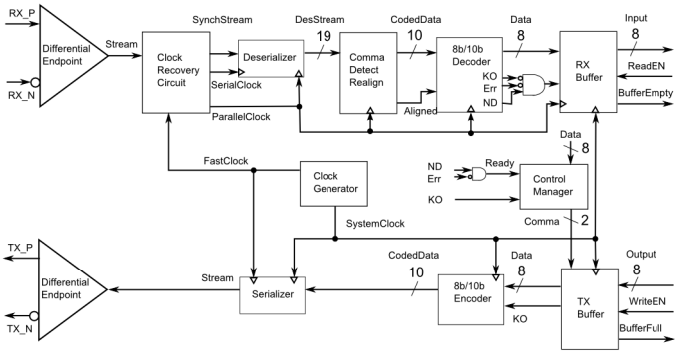

Figure 5. System architecture

The system is fully integrated into low-end FPGAs and has low resource requirements, under 4% of the logic blocks and 20% of the memory elements for our setup. The resource usage is also flexible as the 8b/10b encoder and decoder could be implemented using logic instead of the current memory based approach. When compared to clock recovery based methods, our system is faster due to DDR support. Also compared to clock forwarding based approaches it is slightly slower, as they can achieve 666Mbps of real data, but it has the advantage of simplicity and reduced power due to the reduction in the number of interconnecting wires.

The transceiver was tested at speeds up to 200MHz, 400Mbps, using the same experimental setup. The received data was also verified for correctness to ensure that the system operates flawlessly. A fixed byte pattern was sent to the transmitter and was compared with the data received. During our tests the system was stable and didn't exhibit any errors even when operated at speeds up to 200MHz. Also phase lock was achieved after every reset and was never lost.

V. CONCLUSIONS

We have presented a new method to provide high speed clock recovery for low-cost FPGAs. The method provides a considerable increase in speed compared to reference designs provided by the vendors. Compared to clock-forwarding transmission, our solution provides similar speeds while maintaining the advantages of clock embedding.

The design was also integrated into a transceiver system. The implementation proved to be robust and reliable, capable of operating at 400Mbps. Our transceiver can also be easily integrated into a larger system due to its small size and simple interface. This enables System-on-Chip designs to use high speed serial communication even on low cost FPGA chips.

REFERENCES

[1] Abhijit Athavale and Carl Christensen, “High-Speed Serial I/O Made Simple A Designer’s Guide, with FPGA Applications” , Edition 1.0, Xilinx Connectivity Solutions, 2005.

[2] Gardner, Floyd M., “Phaselock Techniques “, 3rd Ed., New Jersey, John Wiley & Sons Inc, 2005.

[3] Min Yong Jeon, Young Ahn Leem, Dong Churl Kim, Eundeok Sim, Sung-Bock Kim, Hyunsung Ko, Dae-Su Yee, and Kyung Hyun Park, “40 Gbps All-Optical 3R Regeneration and Format Conversion with Related InP-Based Semiconductor Devices,” ETRI Journal, vol.29, no.5, pp.633-640, Oct. 2007.

[4] Nick Sawyer, “Data to Clock Phase Alignment” , Xilinx Application Note XAPP225 (v1.2), 2007.

[5] J. D. H. Alexander., “Clock recovery from random binary data,” Electron. Lett., vol. 11, pp. 541–542, Oct. 1975.

[6] Behzad Razavi, “Challenges in the Design of High-Speed Clock and Data Recovery Circuits” , IEEE Communications Magazine Vol 40, Nr: 8, Page: 94-101, 2002.

[7] Shao-Hung Lin, Shen-Iuan Liu, “Full-Rate Bang-Bang Phase/Frequency Detectors for Unilateral Continuous-Rate CDRs“, IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 55, nr. 12, pp. 1214-1218, 2008.

[8] “Using Digital Clock Managers (DCMs) in Spartan-3 FPGAs”, Xilinx Application Note XAPP462 (v1.1) 2006.