

# Fast Asynchronous Shift Register for Bit-Serial Communication

Rostislav (Reuven) Dobkin, Ran Ginosar, Avinoam Kolodny

VLSI Systems Research Center, Technion—Israel Institute of Technology, Haifa 32000, Israel  
rostikd@tx.technion.ac.il

## Abstract

*A fast asynchronous shift register is used as the serializer and de-serializer in a novel bit-serial on-chip communication link. The link employs two-phase transition-based LEDR encoding. Acknowledgement is generated only at the word level, rather than bit by bit. The shift register is designed to achieve bit time of a single gate delay. It is based on a wave-pipelined control path and on transition latches. The circuit achieved 67 Gbps data rate when simulated on 65nm CMOS technology and was immune to in-die process variations of up to  $10\sigma$ .*

## 1. Introduction

Delays over global wires in large systems on chip (SoC) do not scale with technology. This problem challenges on-chip data communications in terms of latency, throughput and power. Large SoCs typically require multiple long data channels that interconnect far-away modules. Bit parallel data links provide high data rates at the cost of large chip area, routing difficulty, and high power. In addition, such links are often utilized only a small portion of the time, but dissipate leakage power at all times. Leakage is incurred at the line drivers and also at the repeaters, which are often necessary for long interconnects.

Bit-serial communication offers an alternative to bit-parallel interconnects, mitigating the issues of area, routability, and leakage power, since there are fewer wires, fewer line drivers, and fewer repeaters. However, in order to provide the same throughput as an N-bit parallel interconnect, the serial link must operate N times faster.

Point-to-point communications over long data links in large SoCs may cross multiple clock domains, requiring synchronization, whether the link is bit-parallel or serial. A common timing mechanism for serial interconnects injects a clock into the data stream at the transmitting side and recovers the clock at the receiver. Such clock-data recovery (CDR) circuits often require a power-hungry PLL, which may also take a long while to converge on the proper clock frequency and phase at the beginning of each

transmission. If the receiver and transmitter operate in different clock domains, the transaction must also be synchronized at the target, incurring additional time delay and power. Alternatively, an asynchronous data link employs handshake instead of clocks. Traditional asynchronous protocols are relatively slow due to return-to-zero (RTZ) requirement and the need to acknowledge transitions [1][2]. Recently proposed improvements of conventional asynchronous protocols share data lines, but their performance depends on wire delays [3]. We investigate a faster serial communication protocol which is designed to overcome the wire delay limitation.

A non-return-to-zero (NRZ) asynchronous protocol is investigated. Each bit is encoded by a single transition. The protocol avoids the usual per-bit acknowledgment, providing only per-word acknowledgement instead. The minimal inter-bit separation time is a single gate delay (FO4 inverter delay of the technology, e.g. 15 ps for high performance 65 nm process). The throughput is not bounded by wire delay any more, but rather by gate delay. Simulations of our circuits show correct operation at the maximal required data rate (67 Gbps on 65 nm CMOS).

The one gate delay data cycle requirement challenges the transmitter (serializer, line drivers), the channel wires, and the receiver. In this paper we focus on the shift-register in the serializer and de-serializer of the transmitter and receiver, respectively.

Section 2 reviews previously published high speed asynchronous shift registers and pipelines. Section 3 presents the architecture of the bit-serial data link. The fast serializer and de-serializer are described in Section 4, and the asynchronous shift-register design is detailed in Section 5. The circuit is studied in Section 6, and simulation results are discussed in Section 7.

## 2. Previous Work

Numerous high-speed asynchronous shift registers, FIFOs and related pipelining techniques have been published. The fastest designs employ QDI [2][4][5], GasP [6]—[9], and Wave-Pipelining [10]—[13]. Other fast circuits have also been proposed [14]—[17].

**Table 1: Data Cycle Mapping For Bit-Serial Versions of Several Pipelines**

Name	Data Cycle (# of FO4 Inv. Delays)	Data Cycle (# of Transitions)	Family	Reference
PCHB	18.6* (0.8 Gbps at 250nm)	14	QDI	[4]
PCFB	21.3* (0.7 Gbps at 250nm)	12	QDI	[4]
RSPCHB	16.6* (0.9 Gbps at 250nm)	14	QDI	[4]
WCHB	14.9* (1.0 Gbps at 250nm)	10	QDI	[4][5]
RSPCFB	14.9* (1.0 Gbps at 250nm)	10	QDI	[4]
1-of-4	10.0 (0.6 Gbps at 350nm)*	6**	QDI	[2]
STFB	9.3* (1.6 Gbps at 250nm)	6	QDI	[5]
RSDL	5.1* (4.2 Gbps at 180nm)	6	Dual Rail	[14]
asP* FIFO	3.6* (1.7 Gbps at 600nm)	6**	GasP	[6]
GasP FIFO	7.0* (1.5 Gbps at 350nm)	6	GasP	[7][8][9]
Wave-Pipelined Shift-Register	7.1* (3.0 Gbps at 180nm)		Wave-Pipe	[12][13]
Surfing	5.0 (2.1 Gbps at 350nm)*	6	Wave-Pipe	[10][11]
MCML	3* (6.7 Gbps at 180nm)			[16]
Analog Pipeline	2-3 (4.8 Gbps at 180nm)*			[14]
This Work	1 or less (67 Gbps at 65nm)	1		

\*Estimated data cycle/rate according to reported maximal throughput/minimal data cycle and values listed in Table 2.

\*\*Estimated.

**Table 2: Examples of One-Gate Delay Values Scaled with Technology**

Technology	FO4 Inverter Delay*	Rate Upper Bound
600 nm	165 ps	6.1 Gbps
350 nm	95 ps	10.5 Gbps
250 nm	67 ps	14.9 Gbps
180 nm	47 ps	21.3 Gbps
130 nm	33 ps	30.3 Gbps
90 nm	22 ps	45.5 Gbps
65 nm	15 ps	67 Gbps

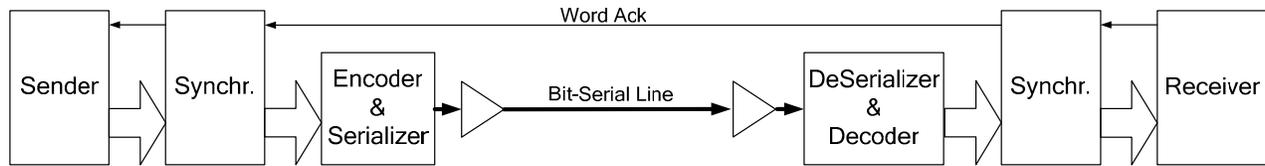
\*Delays are based on High Performance processes [18]

Table 1 lists different pipeline techniques according to their minimal data cycle. Some of the techniques, e.g. wave-pipelining, are not applicable to SR/FIFO implementations, since they cannot be stopped by the input control signal and cannot store all internal states (some waves disappear). Wave-pipelining is a latch-less technique that supports very high throughput, where only the final result at the pipeline output is sampled. The techniques that provide storage capability require data cycles longer than a single gate delay. Table 2 provides reference gate-delay values for the high-performance

variant of several technology nodes. These numbers are scaled for FO4 inverter delays, based on the FO3 NAND delay model provided by the ITRS [18]. They are used to compute data cycle in terms of the number of FO4 inverter delays in Table 1. Next, we introduce single gate-delay shift-register that meets the high-rate requirements.

### 3. A High-Rate Serial Communication

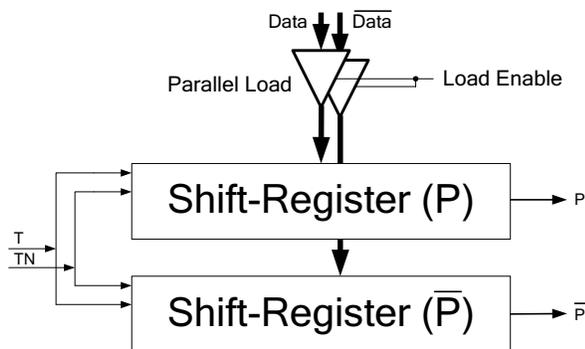
Throughput of a communication line may be enhanced by pipelining. Synchronous pipelines require clock distribution and incur flip-flop latencies. Asynchronous pipelines eliminate the clock and replace flip-flops by lower latency latches, but incur handshake overhead due to having to acknowledge each data transfer [2][4][5]. Non-pipelined methods include source-synchronous signaling, typically used for chip-to-chip interconnects [19]–[23]. The principal disadvantage is the need to embed a very fast clock in the data at the transmitter and to extract it using a PLL-based CDR circuit at the receiver. These circuits dissipate high power, require large area, are sensitive to jitter and are therefore less attractive for fast on-chip interconnects. Wave pipelining may help speeding up the pipeline thanks to eliminating some latches and clocks [24][25].



**Figure 1: Fast Bit-Serial Communication Scheme**

We propose a novel serial communication scheme (Figure 1) that employs low-latency synchronizers at the source and sink [26], two-phase NRZ Level Encoded Dual Rail (LEDR) asynchronous protocol (allowing non-uniform delay intervals between successive bits) [27][28], serializer and deserializer, line drivers and receivers, and differential channel encoding. Acknowledgment is returned only once per word, rather than bit by bit, enabling multiple bits in a wave-pipelined manner over the serial channel. The wires should be designed as waveguides, enabling multiple traveling signals. At signal propagation velocity of at least  $c/10$  on a well-designed wave-guide, and at a desired data rate of one bit per 15ps (the expected FO4 inverter delay at 65nm), a 1mm wire may carry at least two successive bits simultaneously.

Relative to LEDR, other asynchronous data encodings are slower. Four phase dual rail requires four successive transitions per data bit. Bundled data encoding is sensitive to the relative skew of the request and data lines, and requires about 1.5 transitions per bit (one on request and one half bit on average on the data line). The 1-of-4 protocol requires four successive transitions per two bits [2]. Differential LEDR requires only two simultaneous transitions per bit. It can also achieve higher reliability [29]. The drawbacks of the LEDR code include a possible asymmetric transition count over the two lines (causing channel saturation and thus limiting the maximal speed), the need for an encoder and a decoder, and the risk of interline skew which may limit data rate.



**Figure 2: Serializer. A Similar Circuit Drives the S Lines**

The LEDR code is defined as follows. A serialized sequence  $B(i)$  of bits is encoded into a sequence  $S(i)$ ,  $P(i)$  of State and Phase bits, respectively.  $S(i)=B(i)$  for all  $i$ . Given  $P(0)$ , if  $S(i+1)=S(i)$  then  $P(i+1)$  is the inverse of  $P(i)$ , otherwise  $P(i+1)=P(i)$ . Since we employ differential signaling, the serial channel comprises four wires ( $S$ ,  $S'$ ,  $P$ ,  $P'$ ).

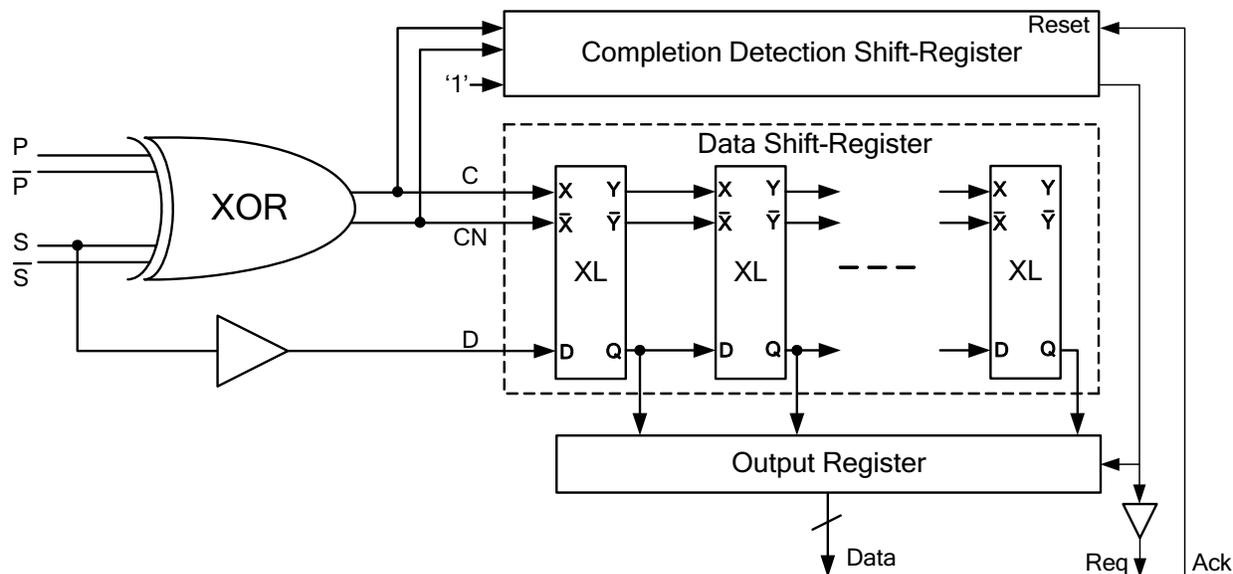
This paper focuses on fast asynchronous serializer and de-serializer. Both are based on a very fast shift-register. The architecture and circuit design of the modules are discussed in the following sections.

#### 4. Serializer and DeSerializer Architecture

The Serializer Module is shown in Figure 2. Encoded parallel word bits ( $P$  and  $S$ ) and their inverted values ( $P'$  and  $S'$ ) are first loaded in parallel into four shift-registers. Subsequently, for each incoming transition on  $T$  and  $TN$  control lines, each shift-register shifts one bit out. Note that the data is shifted both on rising and on falling edges of  $T$ . The control signals  $T$  and  $TN$  are generated using multi-phase clock generator similar to the one described in [30].

The de-serializer (Figure 3) consists of a dual-rail XOR gate for transition detection [31], a shift-register for data sampling and storage, another shift-register for completion detection, and a parallel-load output register. In LEDR encoding only one of the differential pairs at the input ( $S$ ,  $S'$ ,  $P$ ,  $P'$ ) makes a transition per bit. Each transition on either of the two differential pairs is translated into a transition on the  $C$ ,  $CN$  control lines. Each transition causes a single shift at the two shift-registers. Thus, the dual rail XOR retimes the input data.

The Data Shift-Register (Data-SR) is constructed of transition latches (XL). Bits propagate in the SR driven by transitions. For the sake of achieving the fastest possible shift-register operation, full handshake and acknowledgement are avoided. Control transitions propagate in a wave-pipelined manner, and are sensitive to skew and delay variations. All delays along the pipe are carefully balanced. Once all bits of a transmitted word have been stored, bit-parallel decoding can take place at a slower rate.



**Figure 3: DeSerializer & Decoder Module**

The upper shift-register in Figure 3 detects the completion of a full word reception. The Completion Detection SR is reset at the beginning of each word and has a fixed '1' at its input.

LEDR decoding is combined with de-serialization ( $S(i)=B(i) \forall i$ ; see Section 3). Once a transmission is completed, the Data-SR contains a decoded word that is sampled into the output register by a signal from Completion Detection SR. If a second (parallel) data SR is added too, P bits may also be recorded, enabling codes other than LEDR.

The proposed asynchronous SR is designed to provide a very high throughput, while storing the data and accommodating varying delays between successive bits. Below we describe the shift-register structure and its operation in detail.

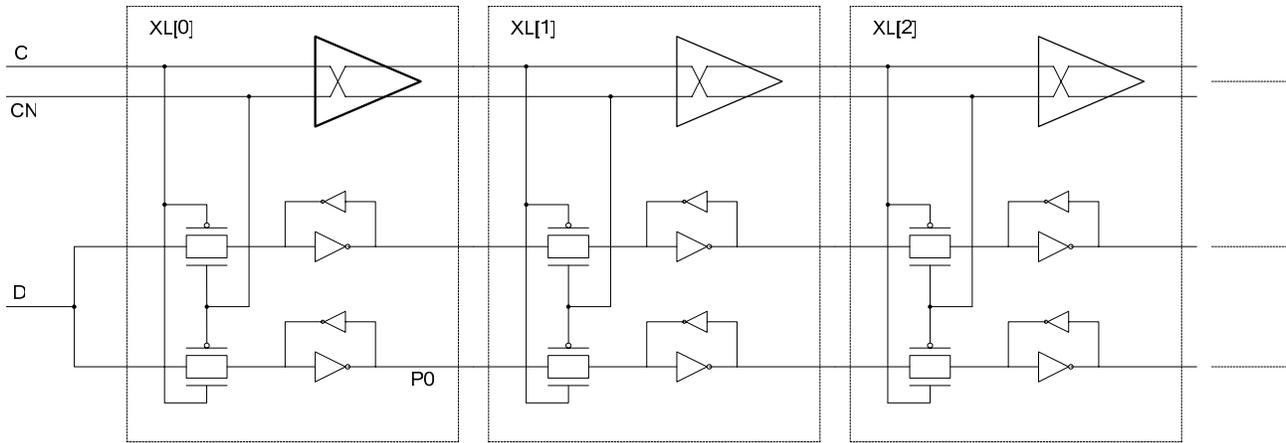
## 5. High Data Rate Asynchronous Shift Register

In order to obtain the desired speed we implement the SR as shown in Figure 4. The SR comprises transition latches (XL). Each XL has a differential control signal C, CN, a dual-rail inverting buffer for the controls, and two separate data paths with internal latches. Each latch consists of an inverter and a (weak) keeper. The differential control lines C, CN are connected to the transmission-gates (TG) of each XL, such that when one TG is open, the other one in the same XL is closed, and the situation is reversed in the next XL. The input data D

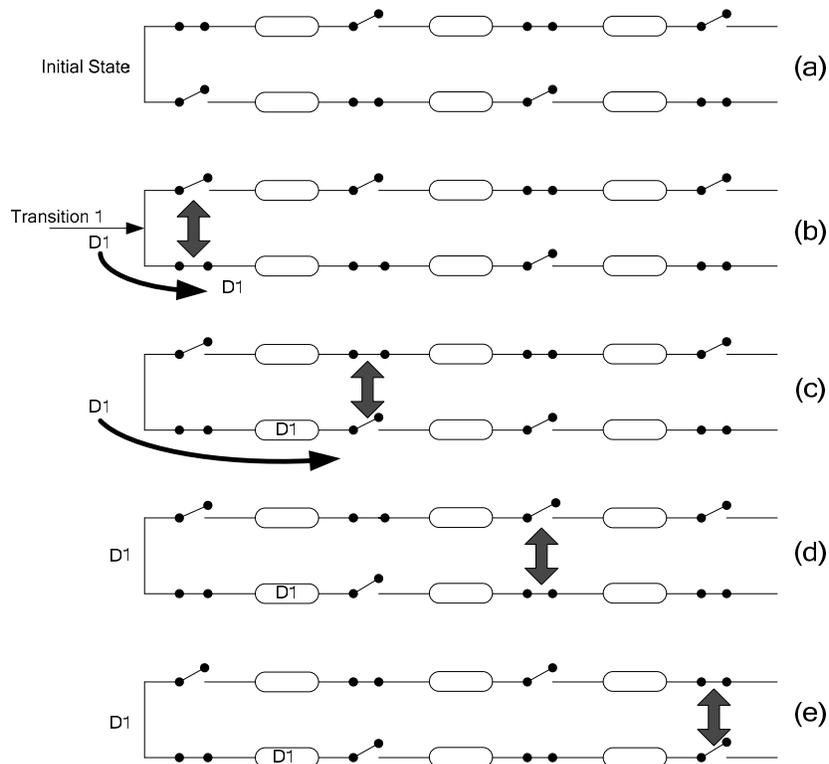
is forked into two parallel pipes before the first XL, so that even bits are written into the bottom data-path and odd bits are directed to the upper one. The control transitions on C, CN propagate without stopping through the control wave-pipeline shifting data in the pipe. Note that the data is sampled and shifted both for the rising and the falling edges of C/CN.

Figure 4 shows the shift-register of the de-serializer. The shift register in the serializer is very similar: Its two outputs are combined (through transmission gates) rather than the inputs. Data is loaded in parallel into the latches via additional tri-state buffers.

On the arrival of a new data item, and the corresponding transitions on C, CN, the data item is written into either the top or the bottom latch of the first XL. The control transitions continue to propagate, shifting the data items that are already stored in the SR. To enable this, the control transitions should propagate through the XL at least as fast as the data. While data propagation through the XL takes approximately one gate delay, the propagation of the control transitions can be made significantly faster. Our SPICE simulations of 65nm CMOS show that the control transition delay through XL is about twice shorter than the gate-delay, over all corners and all simulated in-die variations. This is achieved thanks to the fact that buffer fanout is much smaller than four, especially when the buffer size is large relative to the TGs. Therefore, during one data cycle of one FO4 gate delay, the control transitions manage to propagate over two pipe stages.



**Figure 4: One Gate-Delay De-Serializer Shift-Register (Receiver)**



**Figure 5: SR Operation – One Bit Received. The double arrow shows the moving control transition. (a) Initial state. (b) Arrival of bit D1 and control transition. (c) D1 latches at first stage, control transition moves to second stage. (d), (e) control transition keeps moving.**

We demonstrate the operation of the SR by means of Figure 5 through Figure 7. First, one bit D1 is sent to the SR (Figure 5). The bit propagates into the bottom data path and is stored in the first XL. By the time D1 appears at the XL output, the subsequent TG is already open, preventing data propagation into the next stage. The transition wave (double-arrow) propagates without

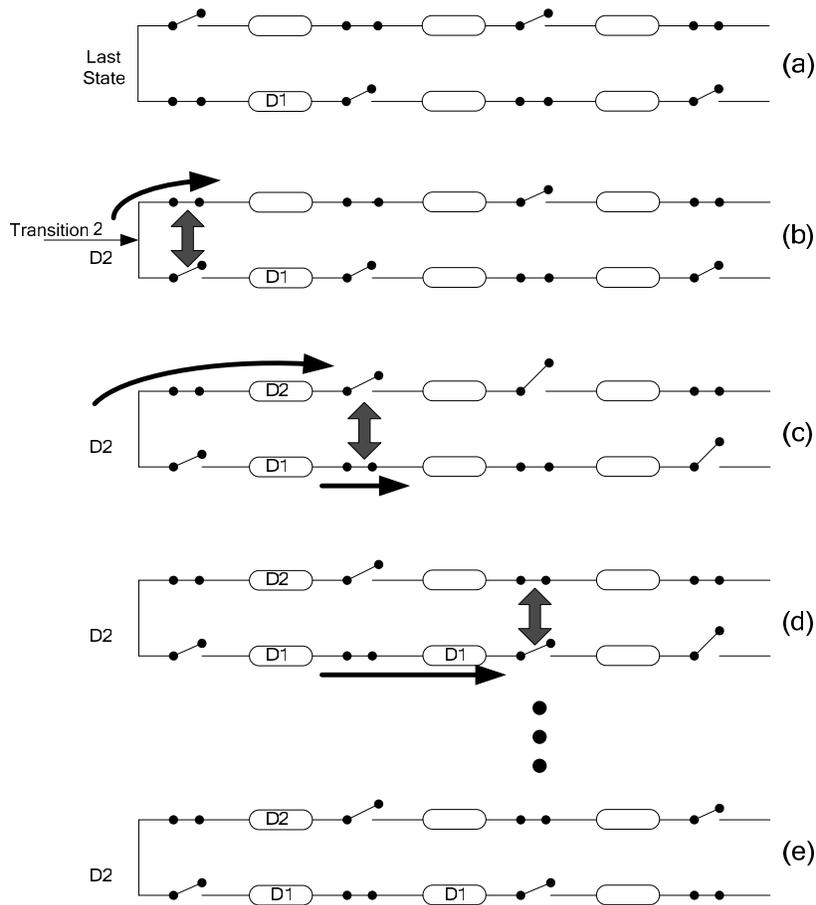
stopping through the SR control wave-pipeline, alternating switch connections at each stage. If there are no further transitions, D1 is stored in the first SR stage and the upper first latch is ready to receive a new datum.

In Figure 6 we demonstrate what happens when another bit arrives after a long time. The SR initial state here is the final state of Figure 5. The new transition copies D2 into

the first stage of the upper data path, opening the path to the second stage. In addition, it closes the TG between the first and second stages of the bottom path, enabling D1 propagation into the second stage. Then, operation is stopped again.

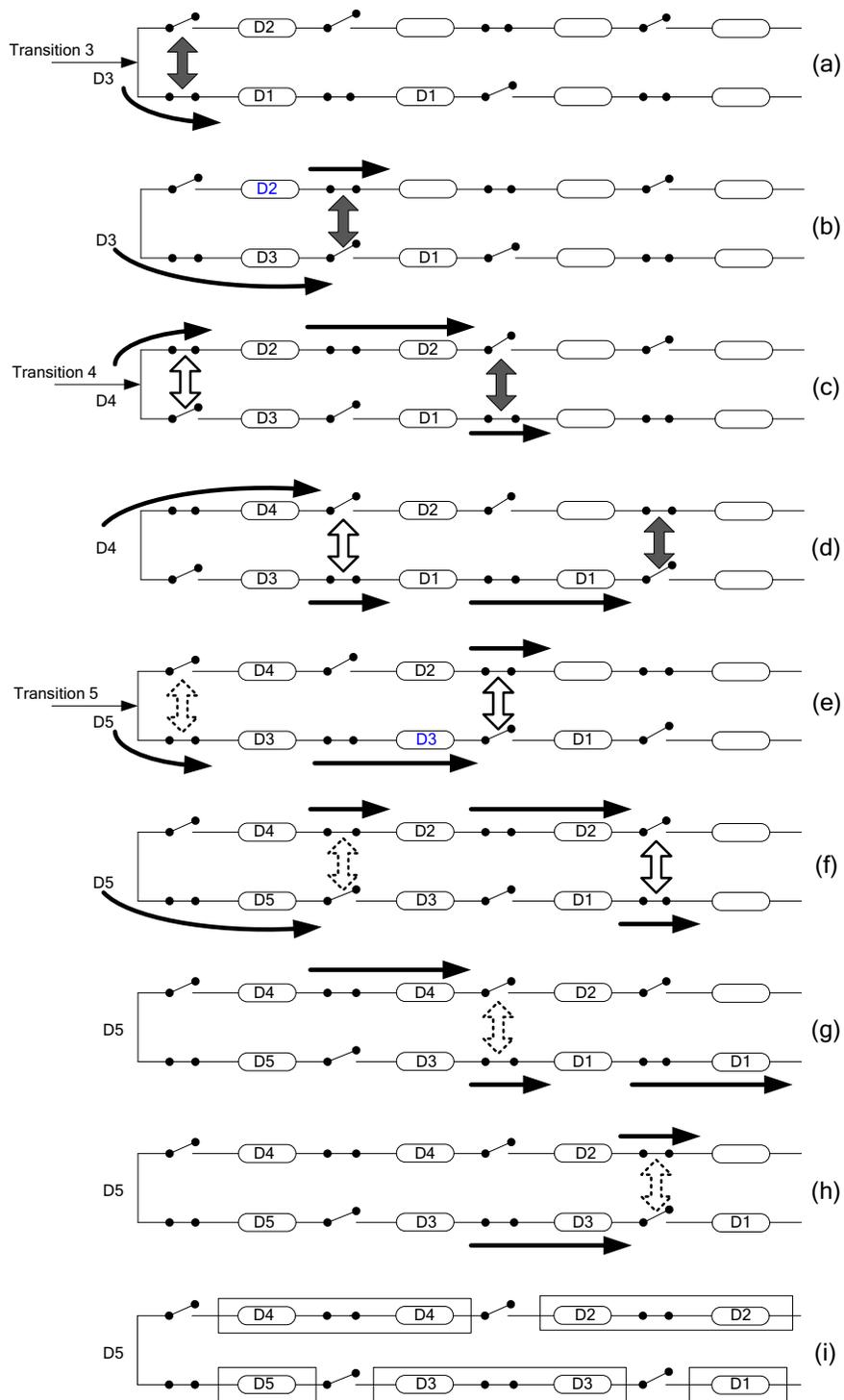
In Figure 7, inputs arrive at maximal rate, with one gate delay separation between successive bits. This demonstrates the maximal operating speed of the SR. Note that control transitions move twice as fast as the data. Three bits, D3, D4, D5 are received at the highest rate.

When working at the highest rate and observing any single datum, every other transition causes the datum to move one stage forward. Thanks to the split at the input of the SR, the data rate in each data path is twice as slow as the input data rate. Data sampling operation, however, lasts only one gate delay at each stage. When the SR is stopped in Figure 7, the bits are spread along the SR. All bits are stored in two consecutive latches, except for the first and last bits, which are stored in a single latch each. In order to store a word of length  $W$ , the SR requires  $W-1$  XLs.



**Figure 6: SR Operation – Second Bit Received.**

- (a) Last state. (b) Arrival of bit D2 and second control transition. (c) D2 latches at first stage, control transition moves to second stage, D1 transfers to second stage.**
- (d) Control moves on, D1 latched in both first and second stages (e) Final state**



**Figure 7: SR Highest Rate Operation.** The different type double arrows show successive control transitions. (a) Arrival of bit D3 (b) D3 overwrites first latch of D1. (c) Arrival of D4; D2 and D1 are shifted forward. (d) D4 sampled by first stage, D1 finishes right shift, D3 starts right shift. (e) D5 arrives, D3 fully copied, D2 starts the right shift. (f) D5 sampled in first stage. D4 starts shifting, D2 fully copied, D1 starts shifting. (g)-(h) continued shifts. (i) First and last bits occupy single latches. Other bits span two latches each.

## 6. SR Circuit Design

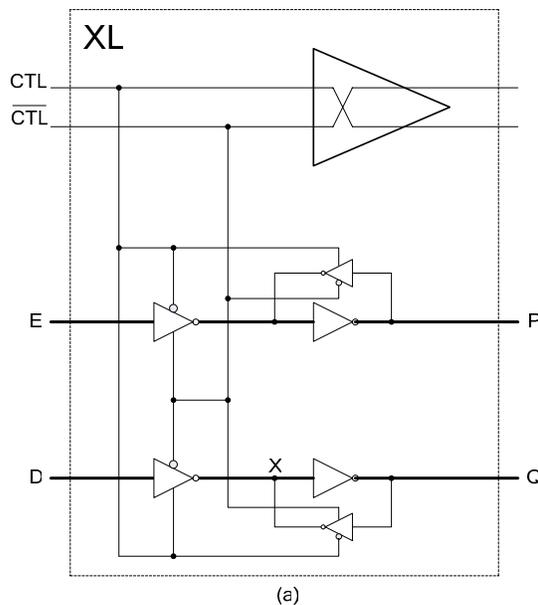
The control transitions are buffered in each XL. It is desirable to retime the two transitions, in order to eliminate relative skew accumulation. Such retiming can be achieved with a differential inverter. Alternatively, two simple inverters may be used (Figure 8b), as long as the resulting skew is acceptable. Another risk facing the use of inverter chains is the disappearance of extremely short pulses. These two factors could restrict the maximal SR length.

In the rest of this section we provide computational analysis for correct sizing of the SR transistors. The TG circuit (Figure 4) suffers from some ringing due to charge sharing and feed-through effects, and hence we employ tri-state inverters instead of the TG and the keeper, as in Figure 8a. Data is sampled at the input of each XL. Correct data sampling during the allocated time (one gate delay) depends on the drive capability of the tri-state inverter  $I$  and the size of the load. Control and data lines in Figure 8b (CTL[i-1], CTLN[i-1], D) are matched in terms of delay. The goal is to find the minimal tri-state inverter  $I$  size that allows correct data sampling.

$C$  and  $R_I$  values are identified as follows. Node A capacitance comprises the input gate capacitance of the forward inverter and the output diffusion capacitances of inverter  $I$  and the keeper.  $R_I$  is the resistance of the pull-up stack of the tri-state inverter  $I$ :

$$C = C_g^{pmos} + C_g^{nmos} + C_{diff}^{k-pmos} + C_{diff}^{k-nmos} + C_{diff}^{I-pmos} + C_{diff}^{I-nmos}$$

$$R_I = R_I^{pullup}$$



Assume a voltage step function at the input of the RC stage. The voltage at node A is ( $V_0$  is the initial voltage):

$$V_A(t) = V_{DD} + (V_0 - V_{DD}) \cdot e^{-t/R_I \cdot C} \quad (1)$$

We approximate an ideal rectangular pulse width  $T$  at the input of the RC stage as 10 ps, having in mind a real pulse width of 15 ps (one gate-delay as specified for 65nm technology). The ideal rectangular pulse is estimated to be less than one gate-delay due to the facts that rise and fall times are finite, and that the data (D) is usually somewhat delayed relative to the control (C, CN). Upon a rising edge, the voltage on node A must reach the following value for correct operation:

$$V_A^{MIN} > V_{DD} - V_T \quad (2)$$

Then,  $R_I \cdot C$  can be expressed as follows:

$$R_I \cdot C < -T / \ln(V_T / V_{DD}) = 7.2 \text{ ps}$$

While  $R_I$  depends only on the tri-state inverter transistors,  $C$  depends also on the latch inverters. Since we use pre-determined sizes for the two latch inverters, the given constraint on  $R_I \cdot C$  affects only the tri-state inverter transistors. Given  $R$  and  $C$  dependence on transistor sizes, and selecting minimal  $L$  for speed, we can determine the desired  $W$  for the four transistors of inverter  $I$ . In order to assure that the tri-state inverters ( $I$  and the keeper) do not slow down the wave-pipelined control, we use dual-rail buffers that are four times larger than the driven load. This simple approach allowed us to obtain a first working point, which can be optimized later.

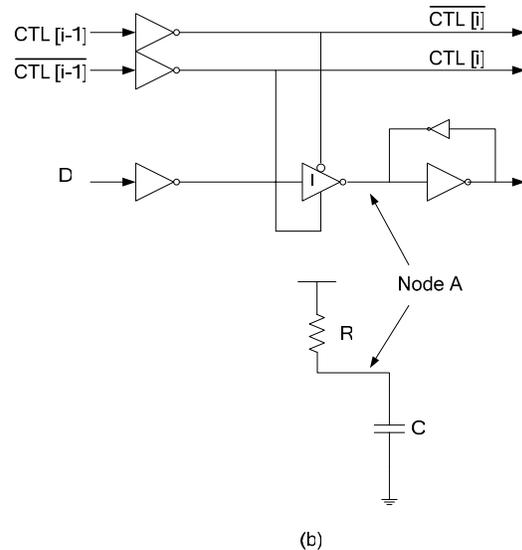
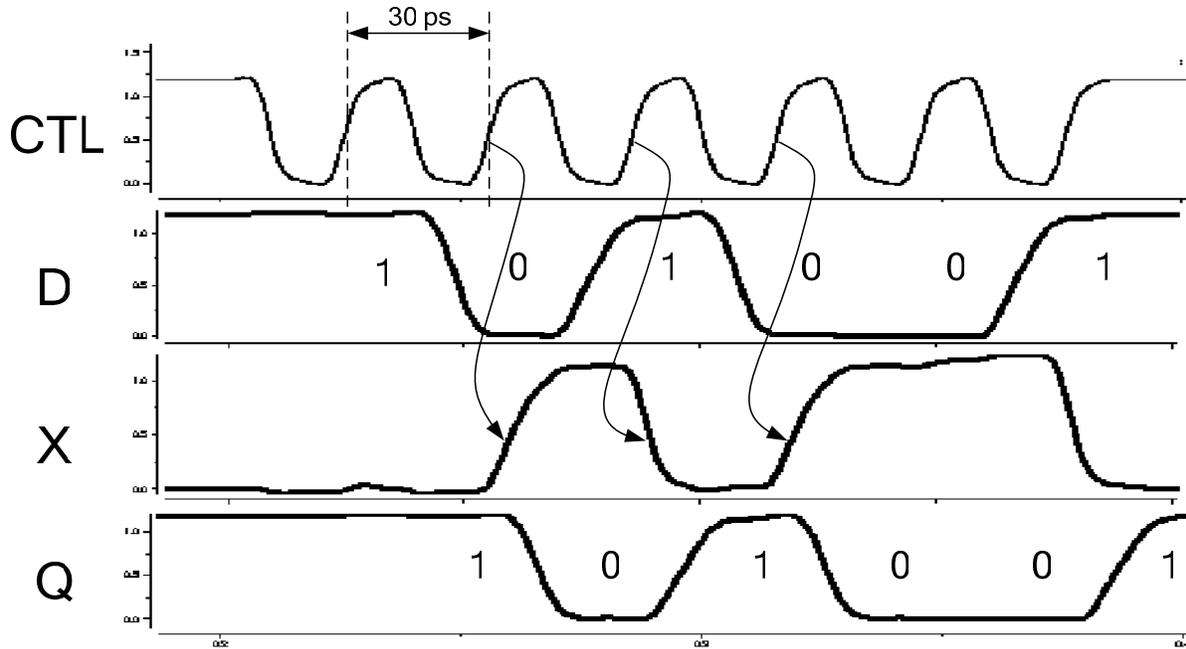


Figure 8: XL Data Sampling



**Figure 9: 15 ps Data Cycle Operation (Node names are defined in Figure 8).  
Input D is sampled during the high phase of CTL**

## 7. Simulation Results

SPICE simulations show correct operation of the shift-register at the target data cycle of 15 ps and target rate of 67 Gbps. The circuit was found operational at 24 PVT corners, ranging over 0.7–1.35 Volt and -10–110°C. At the slowest corner (slow N and P transistors, low voltage and high temperature) the data cycle slowed down to 1.6 times the corresponding gate delay. In all other corners the data cycle ranged between 0.6 and 1.6 times the corresponding gate delay. Monte-Carlo simulations for in-die variations demonstrated correct operation over a very wide range of variation; the circuit failed only at variations above  $10\sigma$  (ten times the standard deviation).

Figure 9 shows an example of simulated signal waveforms at typical operating conditions. Several data patterns were used during the simulation, causing worst case operation at different SR stages. The SR ceased to operate at a data cycle of 13ps.

The SR circuit demonstrates high immunity to delay variations, including those caused by in-die process variations, thanks to its asynchronous structure. The asynchronous wave-pipelined control line is not subject to delay uncertainty, unlike more complex wave-pipelined logic. This is because the control consists merely of a differential line, and the two complementary signal wires

are retimed at each stage by the dual-rail buffer (when non-differential inverters are used, this immunity may be limited by the inter-line skew). The data path is also quite resilient to delay variations: The data are switched and sampled at each stage, timed by control transitions. Thus, delay matching is limited to within a single XL stage.

## 8. Conclusions

We have shown a very fast asynchronous shift register, designed for high speed bit-serial on-chip interconnection channel. The channel employs two-phase transition based LEDR encoding and differential signaling. The channel operates like a fast wave-pipeline, with multiple bits traveling on the wires simultaneously. Acknowledgement is provided only at the word level, in order not to slow down the transmission. This scheme requires very fast serializers and de-serializers, consisting of the fast shift register described in this paper.

The asynchronous shift register can send or receive a new bit at a data cycle of a single FO4 inverter delay (15ps on 65nm process). Thus, it can shift data at the fast rate of 67Gbps. The shift-register is based on novel Transition Latches, which employ a wave-pipelined control structure and two parallel latches. On each control

transition at its input, the shift register shifts all stored data items by one stage.

The shift register is robust in terms of in-die variations. It has been simulated successfully on 65nm process with up to 10 $\sigma$  process variations, and over 24 PVT corners. The fast SERDES, made possible by this shift-register, is useful for high-bandwidth long on-chip interconnects, where bit serial communication is preferred thanks to reduced area, easier routing and reduced leakage.

## 9. Acknowledgement

We thank Eitan Grau, Doron Gershon, Omer Vikinski, Alex Lyakhov, Josh Rotshtein and Charles Dike from Intel for their assistance to this research. Comments of the anonymous reviewers helped to improve this paper significantly. This research was funded in part by Intel Corp., Semiconductor Research Corporation (SRC), and the iSRC consortium.

## References

- [1] J. Teifel, R. Manohar, "A High-Speed Clockless Serial Link Transceiver," Proc. ASYNC, pp. 151-161, 2003.
- [2] W. Bainbridge, S. Furber, "Delay Insensitive System-on-Chip Interconnect using 1-of-4 encoding", Proc. ASYNC, pp. 118-126, 2001.
- [3] R. Ho, J. Gainsley, R. Drost, "Long Wires and Asynchronous Control," Proc. ASYNC, pp. 240-249, 2004.
- [4] R.O. Ozdag, P.A. Beereel, "High-Speed QDI Asynchronous Pipelines," Proc. ASYNC, pp. 13-22, 2002.
- [5] M. Ferreti, P.A. Beereel, "Single-Track, Asynchronous Pipeline Templates Using 1-of-N Encoding," Conf. of Design Automation and Test in Europe, pp.1008-1015, 2002.
- [6] C.E. Molnar, I.E. Sutherland et al. "Two FIFO Ring Performance Experiments," Proceedings of the IEEE, pp. 297-307, 1999.
- [7] I.E. Sutherland, S.M. Fairbanks, "GasP: A Minimal FIFO Control," Proc. ASYNC, pp. 46-53, 2001.
- [8] J. Ebergen, "Squaring the FIFO in GasP," Proc. ASYNC, pp. 194-205, 2001.
- [9] J. Ebergen, J. Gainsley, J. Lexau, I.E. Sutherland, "GasP Control for Domino Circuits," Proc. ASYNC, pp. 12-22, 2005.
- [10] B.D. Winters, M.R. Greenstreet, "A Negative-Overhead, Self-Timed Pipeline," Proc. ASYNC, pp. 37-46, 2002.
- [11] S. Yang, B.D. Winters, M.R. Greenstreet, "Energy Efficient Surfing," ASYNC, pp. 2-11, 2005.
- [12] G. Lakshminarayanan, B. Venkataramani, "Optimization Techniques for FPGA-Based Wave-Pipelined DSP Blocks," IEEE TVLSI Systems, 13(7), 2005.
- [13] S.J. Lee, K. Kim, H. Kim, N. Cho, H.J. Yoo, "Adaptive Network-on-Chip with Wave-front Train Serialization Scheme," Proc. VLSI Circuits, pp. 104-107, 2005.
- [14] N. Saadallah, X. Kong, R. Negulescu, "High-Speed Reduced Stuck Dual Lock Circuits," Proc. ASYNC, pp. 219-228, 2004.
- [15] S. Fairbanks, S. Moore, "Analog Micropipeline Rings for High Precision Timing," ASYNC, p. 41-50, 2004.
- [16] T.W. Kwan, M. Shams, "Design of High-Performance Power-aware Asynchronous Pipelined Circuits in MOS Current Mode Logic," Proc. ASYNC, pp. 23-32, 2005.
- [17] W.P. Burlison, M. Ciesielski, F. Klass, W. Liu, "Wave-Pipelining: A Tutorial and Research Survey," IEEE TVLSI, 6(3), 1998.
- [18] International Technology Roadmap for Semiconductors (ITRS), 2003.
- [19] C. Svensson, J. Yuan, "High Speed CMOS Chip to Chip Communication Circuit," Proc. of ISCAS, pp. 2228-2231, 1991.
- [20] S. Sidiropoulos, "High Performance Inter-Chip Signaling," Tech. Rep. CSL-TR-98-760, Stanford Univ., 1998.
- [21] C.K.K. Yang, "Design of High-Speed Serial Links in CMOS", PhD Thesis, Stanford University, 1998.
- [22] W.F. Ellersick, "Data Converters for High Speed CMOS Links," PhD Thesis, Stanford Univ., 2001.
- [23] H.O. Johansson, J. Yuan, C. Svensson, "A 4 Gsamples/s Line-Receiver in 0.8 um CMOS," Proc. of Int. Symp. VLSI Circuits, pp. 116-117, 1996.
- [24] J. Xu, W. Wolf, "A Wave-Pipelined On-chip Interconnect Structure for Networks-on-Chips," Proc. of High Performance Interconnects Symposium, pp. 10-14, 2003.
- [25] J. Xu, W. Wolf, "Wave Pipelining for Application-specific Networks-on-Chip," International Conference on Compilers, Architecture, and Synthesis for Embedded System, Grenoble, pp. 10-14, 2003.
- [26] R. Dobkin, R. Ginosar, C.P. Sotiriou, "Data Synchronization Issues in GALS SoCs," Proc. ASYNC, pp. 170-179, 2004.
- [27] M.T. Dean, T. Williams et al. "Efficient Self-Timing with Level-Encoded 2-Phase Dual-Rail (LEDR)," Proc. ARVLSI, pp. 55-70, 1991.
- [28] D.H. Linder and J.C. Harden, "Phased Logic: Supporting the Synchronous Design Paradigm with Delay-Insensitive Circuitry," IEEE Trans. Computers 45(9), pp. 1031-1044, 1996.
- [29] F. Worm, P. Ienne, P. Thiran, "Soft Self-Synchronizing Codes for Self-Calibrating Communication," Proc. ICCAD, pp. 440-447, 2004.
- [30] M.J.E. Lee, "An Efficient I/O and Clock Recovery for TERABIT Integrated Circuits Design," PhD Thesis, Stanford Univ., 2001.
- [31] I.E. Sutherland, "Inverse XOR and XNOR Circuits," US Patent 5,861,762, 1999.