A Two-Phase Return-to-Zero (RZ) Asynchronous Transceiver Circuit for Pipe-Lined SoC Interconnects

Muhammad E. S. Elrabaa
Computer Engineering Department
King Fahd University of Petroleum and Minerals
Dhahran, Saudi Arabia
elrabaa@kfupm.edu.sa

***Abstract*—A new delay-insensitive two-phase asynchronous handshaking protocol has been developed. The new protocol utilizes return to zero data format which simplifies communication circuits design significantly. Robust transceiver circuitry that implement this protocol have been developed and simulated using a 0.13µm, 1.2V technology to verify their performance.**

I. INTRODUCTION

Current SoCs not only feature multiple clock domains but also integrate a wide range of blocks (IPs) with various data communication needs and patterns. In addition, SoC designs usually have very short time-to-market demands. This requires efficient design flows that can achieve time closure of the whole SoC in short times. As a result of these requirements two main new design paradigms have emerged to satisfy the communication needs of these SoCs while enabling a reasonable timing closure of the complete SoC design; Network-on-Chips (NoCs) [1-3] and Globally Asynchronous Locally Synchronous (GALS) systems [4-6].

NoCs research aims at developing scalable interconnect architectures that can provide a mean for routing data between SoC IPs with minimum latency over shared interconnects. While research on GALS aims at developing circuits, methodologies and models for interconnecting synchronous blocks with separate clock domains using asynchronous interconnects. Hence NoCs can be viewed as a special case of GALS. In any case, both shares the common problem of designing the point-to-point interconnect circuitry (repeaters, buffers, and pipeline stages) between routers and/or IP blocks. Hence developing high performance robust interconnect circuitry is essential for current and future SoCs.

GALS are categorized into three types based on their communication schemes [6]; pausible clocks; asynchronous, and loosely synchronous. Pausible clock systems stop (or pause) the clock of the IP block during data transfer. This goes against the fundamental concept of decoupling 'computations' from 'communications' rendering this design style impractical. With each additional input channel, the percentage of idle time would increase even further. Loosely synchronous techniques would require some form of buffering (FIFOs) on the receiver and/or transmitter sides. Again coupling IP design with the communication (interconnect) design. This increases the SoC's design time significantly. Fully asynchronous interconnects offer the highest degree of robustness and decoupling of different SoC design activities. However, latency and throughput are major concerns. Due to handshaking, each datum transfer would require at least two round trips. Interconnect pipelining and repeaters can improve latency and throughput.

Many researchers have proposed new solutions to improve latency and throughput of asynchronous pipelines [7-11]. In [7,8] control pulses are used instead of traditional transition-coded control. This allows faster acknowledge at the expense of more complex circuit design to precisely control pulse widths and math the wire delays. Other researchers proposed a form of wave-pipelining called surfing interconnects [9-11] where they remove two way handshaking altogether. This adversely affects the robustness of circuits and increase the design time significantly. By trading off design time (complexity) for speed another important feature of asynchronous interconnects is sacrificed, flow control. Asynchronous handshaking not only ensure proper timing of valid data but it also allows receivers to control the flow of data, an essential feature in SoCs. Using FIFO buffers instead of handshaking as proposed in [11] would require flow control at higher levels of the protocol stack. Surfing interconnects resembles source synchronous communications with the request signal being used to strobe the data at the receiver and repeaters with adjustable delays as delay lines. Efficient

This work have been supported by King Fahd University of Petroleum and Minerals through grant # IN070367

source synchronous on-chip serial communication circuits have been proposed in [12,13] where the data and clock are re-timed at the receiver side instead of repeaters along the control line as in [11]. Again flow control would have to be handled at higher levels of the communication protocol stack, something that SoC IPs might not be designed for.

Another concern with asynchronous interconnects is the use of non-standard CMOS circuits. Hence developing robust asynchronous circuits that can be used as 'plug-and-play' hard macros is highly desirable. This can be achieved through the use of delay-insensitive design techniques.

In this work a robust pipelined asynchronous interconnect system is proposed. The proposed interconnect system combines a new handshaking protocol with an efficient delay-independent circuit implementation that keep the delay to a minimum. The new handshaking protocol is introduced in section II followed by the developed circuits that implement it in section III. Simulations results that verify the operation of these circuits are provided in section IV followed by conclusions in section V.

II. THE NEW HANDSHAKING PROTOCOL

In a typical asynchronous pipeline, Figure 1, data is transferred from one stage to the next via a sequence of handshaking signals. A stage would latch a datum when it receives a Request (**Req**) signal from the preceding stage while the next stage had already indicated that it had latched the previous datum (by de-asserting the Acknowledge signal). Traditionally, there have been two main handshaking protocols for asynchronous data exchange; four-phase handshaking and two-phase handshaking. When combined with dual-rail data encoding these protocols yield delay-insensitive (or at least Quasi-delay-insensitive) operation. The four-phase protocol, illustrated in Figure 2(a), uses a return-to-zero (**RZ**) data format requiring 4 steps (or *trips*) to complete a single datum transfer. The transmitter initiates a datum transfer by driving one of the pre-charged data lines low (or high depending on the pre-charged value). The receiver detects the difference between the data lines using a simple CMOS gate, generates the request, latches in the data if the acknowledge signal coming from the next stage is low and force its own acknowledge high. This signals the transmitter that the transfer is successful and it responds by pre-charging the data lines which is detected at the receiver as the request signal going down. The receiver now responds by lowering its acknowledge signal indicating to the transmitter that it is ready for a new data. Since data is level-encoded, conventional circuits can be used in the transmitter and receiver. The two-phase protocol is very similar except that it uses a non-return-to-zero (**NRZ**) data format (no pre-charging) requiring only two steps to complete a datum transfer as shown in Figure 2(b). For this protocol, data is transition encoded which require special circuitry to detect and handle the two possible transitions.

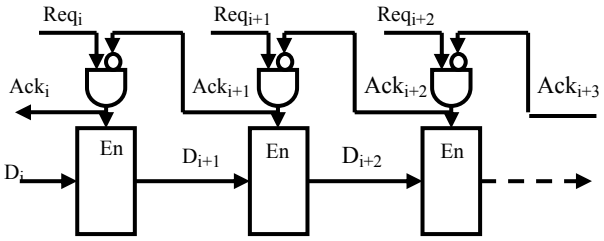
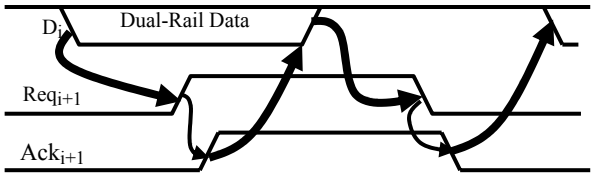
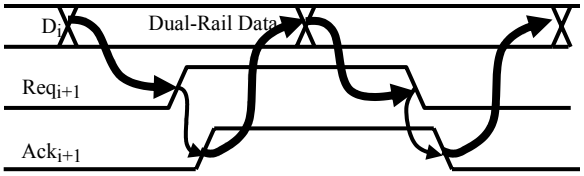


Figure 1. A typical asynchronous pipeline. For dual-rail or 1-in-n encoding, the Req signal is generated from at the receiver.



(a) Four-phase (RZ) handshaking.



(b) Two-phase (NRZ) handshaking.

Figure 2. Traditional asynchronous handshaking protocols with dual-rail data encoding for delay insensitive operation. Di is the data transfer initiated at the transmitter. Both Req and Ack signals are generated at receiver side. Thick arrow curves indicate trips from transmitter to receiver or vise versa.

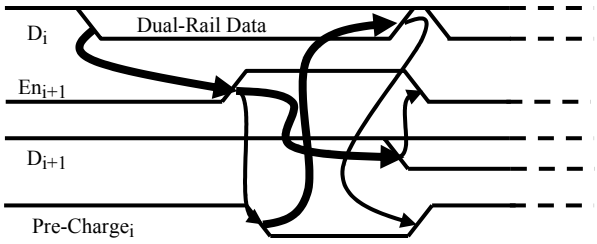


Figure 3. The proposed two-phase (RZ) handshaking. Di is the data transfer initiated at the i^{th} stage. Each data segment is discharged from the transmitter side and pre-charged from the receiver side. Only two trips are required per datum transfer.

Noting that both request and acknowledge signals are generated at the receiver, a new handshaking protocol has been developed that combines the four-phase data level-encoding (i.e. RZ) with the two-phase data exchange steps (*trips*), as illustrated in Figure 3. When a new data initiated at stage ***i*** is received at stage ***i+1***, an enable signal is generated at stage ***i+1*** (En$_{i+1}$). This enable signal would initiate the transfer of data to the next segment (***i+1***th segment) and at the same time activate a pre-charging signal (Pre-Charge$_i$) that would pre-charge the preceding data segment (***i***th segment). This overlaps the transfer of data to the ***i+1***th segment with the pre-charging of the ***i***th segment. So

within two *trips* the data is transferred, similar to conventional two-phase signaling. Because data lines are pre-charged between transfers, then simple level-sensitive circuits can be used, simplifying the circuit design significantly and enabling higher performance. Also, since a data line can only go down, there is no need for an actual data latch. The enable signal can be simply used to drive the data line low using a single NMOS switch, again simplifying the design and reducing the latency of the repeater. Hence each data segment in the pipeline is discharged from the transmitter side and charged from the receiver side. The developed protocol and circuits ensure delay-insensitive operation with no contention between the discharging and charging circuitry on the same data segment.

III. Circuits Description

Figure 4 shows the circuit details of the repeater (transceiver) on one of the dual data lines. The circuit for the other line is similar with Di replaced by Di~ and Di+1 by Di+1~. It has four components as shown in Figure 4(a); a data driver circuit for the next data segment, an enable circuit to generate the control signal (En) for the data driver, a pre-charging driver for the preceding data segment, and a pre-charging control circuit that controls the pre-charging driver. The data driver circuit is a simple NMOS switch with a weak keeper to hold the data line low when the enable signal goes down. The enable circuit, Figure 4(b), has a behavior similar to a Muller-C element. It would assert the enable signal only when the input data (D_i) becomes low while both output data lines are high (indicating that previous data has been transferred). Only when the next data segment (D_{i+1}) is discharged the En signal is de-asserted.

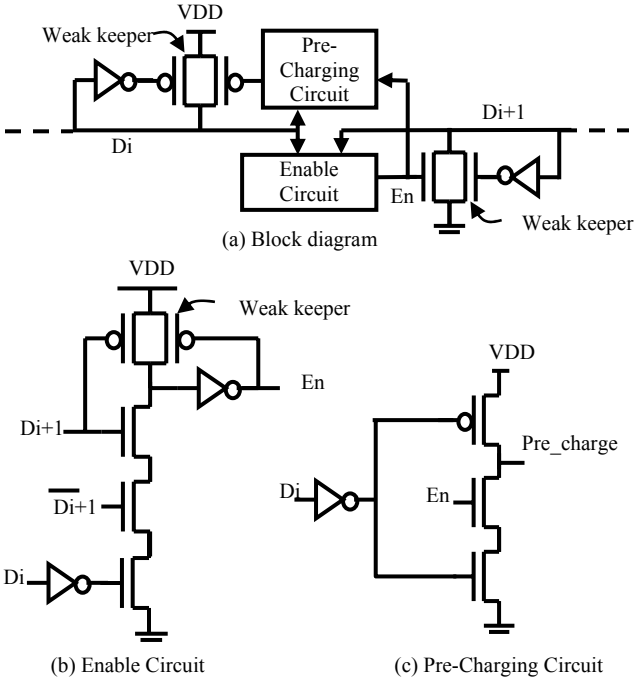


Figure 4. Asynchronous transceiver Circuitry

En would remain low as long as the next segment is low. A weak keeper is added to hold the enable signal low when all data segments are in the pre-charged (high) state. The pre-charging driver for the preceding data segment is a simple PMOS transistor and a weak keeper to hold the data line high. The circuit that controls the pre-charging driver, Figure 4(c), would produce a low signal when both En and Di are low. When the preceding segment is charged (i.e. Di becomes high), the pre-charging signal goes high and the data line is held high by the weak keeper.

Unlike previous work, the widths of the En and Pre-Charge pulses are automatically set by the timing behavior of the data lines and need no special circuit sizing.

IV. Circuit Simulations

Spice simulations using a 0.13µm, 1.2 V CMOS technology were used to verify the operation of the proposed circuits. Figure 5 shows the test pipeline consisting of three stages asynchronous transceivers, a data producer and a data consumer. Wire segments in between are modeled using lumped RC circuits that approximately represent 100µm wires. Transistors were simply sized to achieve 50 ps fall times and 100 ps rise times. No further optimization was carried out to illustrate the robustness of the circuits. Figure 6 shows the simulation waveforms of a single stage transceiver. It shows how the transceiver circuits achieve the appropriate sequence of events on input data, En and pre-charge signals, and output data. It shows also at this wire length, the throughput is ~ 5 Gbs.

To test the complete asynchronous pipeline the following scenario has been simulated; 1st the producer produces data at a constant rate (every 2 ns) while the consumer does not consume any data, Figure 7(a). The figure shows how the pipeline is filled after the injection of 4 data items (all data lines D1-4 are now low).

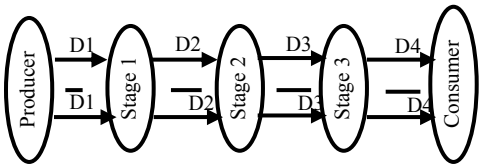


Figure 5. The setup used to test the new transceiver circuit. Wire segments between stages are 100µm long and represented as a lumped RC circuit.

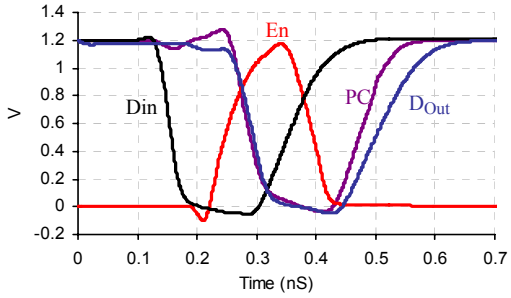


Figure 6. Signal waveforms of one of the transceiver stages. PC is the pre-charge control signal.

Next, the consumer starts consuming data items also at a rate of 2ns. Whenever the consumer consumes a data item (indicated by the pre-charging of D4), all the data in the pipeline move one step forward as evident from the consecutive pre-charging and discharging of the data lines in Figure 7(b). The producer continues to inject data at the same rate the consumer is consuming them, keeping the pipeline full while data move along the pipeline. The 2ns injection/consumption rate was used to have uncluttered waveform graphs that clearly show the movement of data along the pipeline. At this wire length, the injection rate could have been made as small as 200 ps.

V. CONCLUSIONS

A new two-phase asynchronous handshaking protocol that utilizes dual-rail RZ data encoding has been developed. Allowing simple circuit implementations that keep minimize latenc. Efficient robust circuit implementation of the protocol has been realized and tested using SPICE simulations. With almost no circuit optimization, the new transceiver circuit can achieve a throughput of 5Gbs with wire lengths of ~100µm. The robustness and delay-insensitivity of the developed circuitry would help decouple computations from communications in the SoC design process, significantly increasing the design productivity.

ACKNOWLEDGMENT

Facilities support by King Fahd University of Petroleum and Minerals is highly appreciated.

REFERENCES

[1] L. Benini and G. D. Micheli, "Networks on chips: A new SoC paradigm", *IEEE Computer*, Vol. 35, No 1, pp. 70 – 78, 2002.

[2] W. J. Dally and B. Towles, "Route packets, not wires: On chip interconnection networks", *Proc. 38thDesign Automation Conference*, pp. 684–689, June 2001.

[3] J. Henkel, W. Wolf, and S. Chakradhar, "On-chip networks: a scalable, communication-centric embedded system design paradigm", *Proc. 17th Int'l Conf. VLSI Design*, 2004, pp. 845-851.

[4] D. M. Chapiro, "Globally-Asynchronous Locally-Synchronous Systems", PhD thesis, Stanford University, October 1984.

[5] S. Moore, G. Taylor, R. Mullins, and P. Robinson, "Point to Point GALS Interconnect", *Proc. 8th Int. Symp. Async. Cir. & Sys.* (ASYNC'02), pages 69–75, 2002.

[6] P. Teehan, M. R. Greenstreet, and G. Lemieux, "A survey and Taxonomy of GALS Design Styles", IEEE Design and Test of Computers, pp. 418-428, September-October 2007.

[7] I. Sutherland and S. Fairbanks, "GasP: A Minimal FIFO Control", ASYNC'01, pp. 46–53, 2001.

[8] R. Ho, J. Gainsley, and R. Drost, "Long Wires and Asynchronous Control", ASYNC'04, pp. 240–249, 2004.

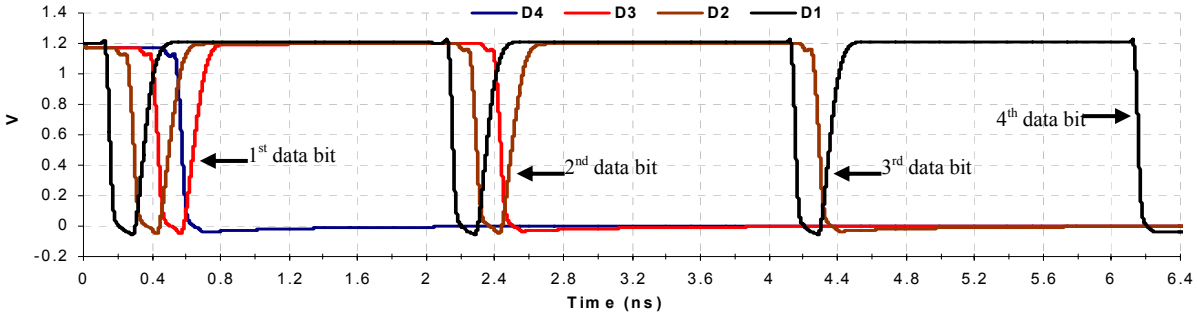
[9] B. D. Winters and M. R Greenstreet, "A Negative Overhead, self-timed pipeline", ASYNC'02, pp. 32–41, 2002.

[10] S. Yang, B. D. Winters, and M. R Greenstreet, "Energy Efficient Surfing", ASYNC'05, pp. 2–11, 2005.

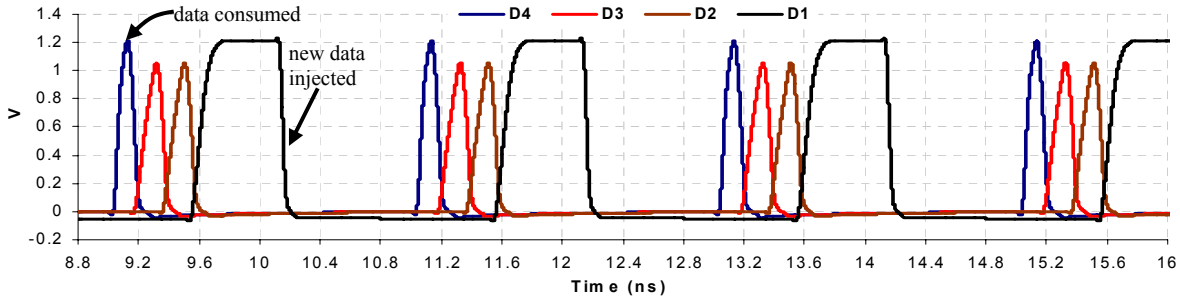
[11] M. R Greenstreet and J. Ren, "Surfing Interconnect", ASYNC'06, pp. 1–9, 2005.

[12] M. E. S. Elrabaa, "A Digital Clock Re-Timing Circuit for On-Chip Source-Synchronous Serial Links", *Proc. IEEE Int. Conf. on Microelectronics*, pp. 206-209, 2006.

[13] M. E. S. Elrabaa, "Portable Clock Recovery Circuits (CRCs) For On-Chip and Off-Chip Serial Data Communication," AJSE Journal, pp. 109-117, Dec. 2007. http://www.kfupm.edu.sa/publications/ajse/



(a) Data is being injected by the producer (D1) at a rate of 2ns while the consumer is not consuming any data at all. After four data injections the pipeline is full and can not accept any new data (all data lines D1-D4 are low).



(b) Data is being injected by the producer (D1) and consumed by the consumer (D4) at the same rate of 2ns. The data is moving along the pipeline at a speed of ~200 ps/stage.

Figure 7. The data waveforms along the asynchronous pipeline stages for the two communication scenarios.