Asynchronous Bit-stream Compression (ABC)

Arkadiy Morgenshtein, Avinoam Kolodny, Ran Ginosar

Electrical Engineering Department, Technion – Israel Institute of Technology, Haifa, 32000, Israel

[arkadiy@tx.technion.ac.il]

Abstract - Asynchronous signaling is used for high-speed data communication in large Systems-on-Chip in. The bandwidth limitations of serial link dictate a need for real-time compression techniques. In this paper we propose a new technique of Asynchronous Bit-stream Compression (ABC), based on Level Encoded Dual-Rail protocol. The ABC method is based on transitions added to LEDR protocol which allow simple identification of the compression code and ease its separate treatment in the receiver. This compression allows a significant saving in the transmission time and power without losing data. The concept of ABC is described in this paper together with the proposed architecture of its hardware components. Simulations results are presented for several data patterns with various differentiation rates. Application of ABC results in reduction of transmission time by 9% to 54% depending on type of source data.

I. INTRODUCTION

Data communication in large Systems-on-Chip can be performed by serial links in order to reduce the wiring area and power Error! Reference source not found.Error! Reference source not found.[1][3]. In order to maintain similar throughput as in case of parallel link, the data in serial link has to be transferred at much higher bit-rate. This may be problematic if synchronous communication protocol is considered, because of the difficulty in implementation of ultra-fast clock generator and accompanying circuits needed for synchronization. Asynchronous signaling is used in order to allow high-speed communication without the need for clock generation [3]Error! Reference source not found.. One of the popular asynchronous protocols is Level Encoded Dual Rail (LEDR) protocol which eliminates the need for handshake signaling [5].

The relatively low bandwidth of a single wire is the major limiting factor in serial link performance. In order to increase the throughput of the data in the serial link and to utilize the limited bandwidth in the most efficient manner, encoding techniques can be applied to the data. One of the most effective techniques for overcoming the bandwidth limitations of serial link is the real-time compression [7].

In this paper we propose a new technique of Asynchronous Bit-stream Compression (ABC), based on utilization of previously unused transitions in the state machine of Level Encoded Dual-Rail protocol. The proposed technique allows significant savings in terms of transmission time and power consumption in asynchronous serial link, by effectively detecting and compressing sequences of identical symbols in the bit stream.

The paper is composed of the following sections. The Asynchronous Bit-Stream Compression technique is described in Section II. Section III presents the architecture of ABC. The results of ABC application to variety of data patterns are given in Section IV. The work is summarized in Section V.

II. ABC CONCEPT

The concept of LEDR is illustrated in Fig. 1a. The basic LEDR state machine is based on two signals of state (S) and phase (P). If the value of the signal is different from the previous state, S is toggled; if the value is same as before, P is toggled. In this manner, the transitions between the sequent symbols can be identified, while only one of the signals S or P is toggling at each transition. Note, that in LEDR machine none of the transitions occur between states with change in more than one signal at same time.

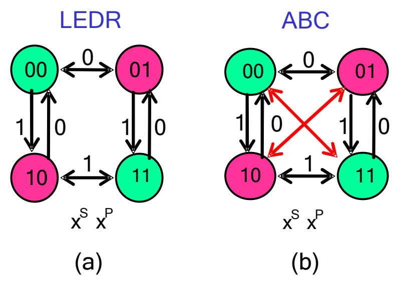

Fig. 1. Comparison of ABC and LEDR state machines

The state machine for communication protocol with Asynchronous Bit-stream Compression in presented in Fig. 1b. ABC is based on LEDR state machine, while utilizing the four missing transitions in the LEDR. The additional transitions (00↔11 and 10↔01) are implemented to encode the sequences of identical signals. These transitions are used to indicate the beginning of the compression and its end in the data flow. The fact that the transitions are different from the standard LEDR protocol allows simple identification of the code, without need for additional signaling. The implementation of the ABC signals has relatively low hardware penalty due to basing on the existing state machine.

The application of the ABC technique can be described using the following algorithm:

a. *Identify a sequence of identical bits*
b. *Mark the beginning of the sequence by one of the ABC transitions*
c. *Transmit the length of the sequence*
d. *Mark the end of the encoding by one of the ABC transitions.*

The Asynchronous Bit-stream Compression is exemplified in Fig. 2. In the LEDR protocol the data is encoded using State and Phase signal. In case of appearance of long sequence of bits with identical value, the Phase signal continues triggering to represent the different bits. In this case, both the transmission time and the power consumption caused by Phase transactions can be saved by applying the ABC technique. As can be seen from the example, when a sequence is identified, it is encoded by ABC to a string containing the number of signals in the sequence framed by two specific ABC transitions. The new ABC transitions allow simple identification of the code and ease its separate treatment in the receiver. This compression allows a significant reduction of number of transitions in the link without loss of data.

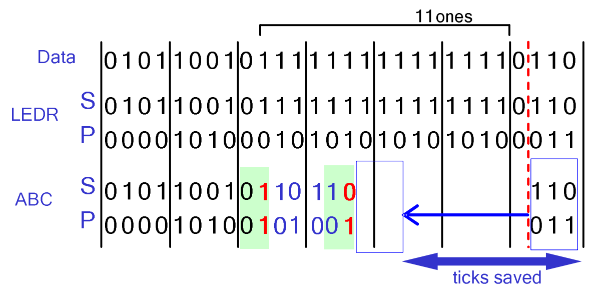

Fig. 2. Example of transitions saving in ABC.

III. ABC ARCHITECTURE

The ABC algorithm requires implementation of specific hardware elements that will be responsible for the processing of the data before and after the transportation along the link. In this section we describe the architecture of the ABC transmitter and receiver.

ABC Transmitter - The architecture of ABC transmitter is shown in the block diagram in Fig. 3. The transmitter is composed of three major modules – the sequence detector, the controller and the ABC state machine. The block diagram contains also the input register and the serializer, which are typical components of the serial link.

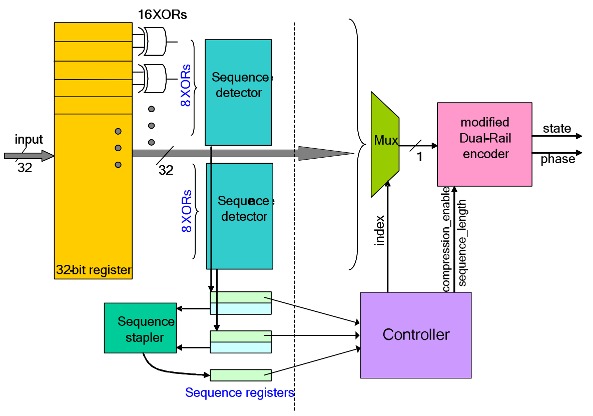

Fig. 3. Block diagram of ABC transmitter.

The architecture of ABC transmitter is presented here for 32-bit register. The modules described below are developed for this data length. However, the concept can be easily extended to larger number of data bits.

Sequence detector – The detection is preformed using two sequence detectors operating in parallel in order to reduce the processing time. Each sequence detector is scanning a 16-bit part of the register by a “window” containing eight 2-bit XOR gates. Each XOR compares a pair of sequent bits in the register. Once a sequence of identical bits is identified, the sequence detector stores the index of sequence and its length in the *sequence registers*. Note that a single sequence can start in one part of the register and end at the other. In this case it is identified by the sequence detectors as two sequences and the indexes of each sequence in the each part are stored by the related sequence detector in the sequence registers.

Controller – Once the indices of the sequences are stored in the registers, there is a need to check if these are separate sequences, or parts of a longer sequence. This operation is performed by the *sequence stapler*. When the sequences parts are combined, the final indexes of all sequences and their lengths are stored in registers. The indexes are used during the transmission by the sequence counter for signaling to the MUX and the encoder when the compression starts (signals *index* and *compression_enable*) and what is the length of the sequence to be encoded (signal *sequence_length*).

ABC state machine – The state machine of Dual Rail encoder is modified to contain the additional transitions of ABC as was shown in Fig. 1b. The encoder has two operation modes: a) regular LEDR protocol for uncompressed data, b) ABC mode for compressed data. The transition between the modes is performed according to the signal from the controller. The beginning and the end of the compression are symbolized by ABC transitions. After the compressed sequence is transmitted, the controller sets the index of the MUX to point on the bit following the sequence in the register and the transmission continues in regular LEDR mode until the index of next sequence is reached.

ABC Receiver – The architecture of ABC receiver is shown in Fig. 4. The presented structure contains the output register of the link. The operation of the receiver in this case combines the decoding of the data from dual-rail to single-bit flow controlled by clock generated from the S and P signals, conversion of the data from serial to parallel format and decompression of the sequences that were processed by ABC.

Clock generator – This module translates the transitions in S and P signals into clock pulses. The clock is used for synchronization of the data storage in the output register, as well as the controlling the state machine. The additional function integrated in the clock generator is the identification of the ABC transitions used for signaling of compression beginning. When the ABC transition is identified the *compression_start* signal is toggled in the input to FSM, causing it to switch to a different operation mode.

Enabling decoder – The conversion of the data from serial to parallel is performed here by providing the *Data* signal to all the cells of the register and controlling the *enable* signal of each cell in order to write the data to the related location in the register. These enable signals are set by the enabling decoder. When the data is received in regular LEDR mode, the enabling decoder allows only one enable signal to be activated in every cycle. When ABC mode is activated for decompression, the enabling decoder activates the enable signals starting from the index *comp_from* till the index *comp_till*. This type of operation allows a fast storage of the sequence data during the decompression, while all the cells get the same value in a single clock cycle.

Receiver FSM – The structure of the receiver final state machine is shown in Fig. 5. The operation starts at the *Count* mode, in which with each clock pulse an internal counter is increased by one. The counter controls the enabling decoder in order to enable one register cell for writing during each clock cycle. When ABC compression is identified and *comp_start* signal is toggled, the FSM switches to *Comp_Decode* mode. It stays in this mode until all the bits of sequence length code are received. After the length is known, the FSM switches to *Comp-Write* state and creates two signals for the indexes of sequence beginning and end – *comp_from* and *comp_till*. These signals are entered to the enabling encoder to enable the related cells in the register for writing. At the next clock the FSM returns to the regular operation mode.

Design Considerations – The proposed architecture of the ABC ink interfaces has some specific parameters that have to be considered during the design of the link. While the packet size was considered here as 32 bit, the design considerations described here can be easily adjusted to different packet sizes as well.

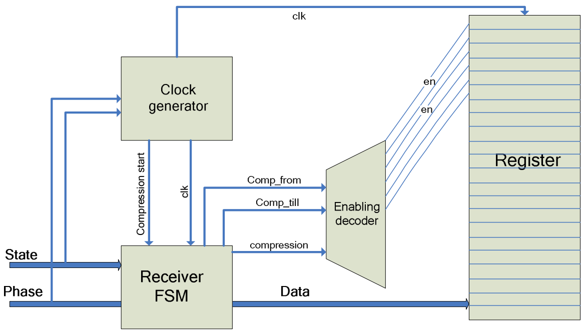

Fig. 4. Block diagram of ABC receiver.

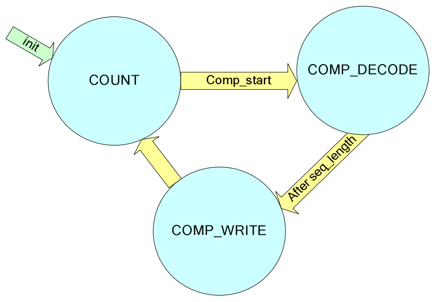

Fig. 5. Receiver final state machine

As was shown, the detection of the sequence is performed using 8-bit sets of XOR gates on 16-bit portion of the data. The detection process will last for 8 clock cycles. This consideration may influence the design of the system while several architecture alternatives can be proposed.

a. The ABC transmitter can be modified by addition of a register. In this case two packets will be contained in the transmitter. The detection of the sequences in one packet can be performed simultaneously with the transmission of the other packet. In this manner, a maximal throughput will be maintained, traded off with increased area and power consumption.
b. The transmission of the packet can be delayed by 8 clock cycles in order to allow the scanning of all the bits and detection of the sequences. In this case, no additional register is needed, but there is a penalty in terms of transmission time.
c. The transmission time, area and power can be saved by compromising on the compression efficiency of ABC. The first 8 bits of the packet can be transmitted without compression, while at the same time, scanning and detection of the sequences is performed on the remaining 24 bits.

In this work we adopt the third option for ABC transmitter implementation. This architecture defines the maximal length of the sequence that can be detected and treated as function of packet length, sequence detector size and the size of XOR "window":

$$L_{seq_{\max}} = L_{packet} - \left(L_{Seq\_detector} - L_{XOR\_window}\right) \qquad (1)$$

The number of sequences that can be detected is also a parameter that can be controlled during the design. The maximal number of sequences that can be detected in a packet is a function of the relation between the sizes of sequence detector, the packet length and the XOR "window":

$$N_{seq_{\max}} = \frac{L_{packet} / L_{Seq\_detector}}{L_{Seq\_detector} / L_{XOR\_window}} = \frac{L_{packet} \cdot L_{XOR\_window}}{L^2_{Seq\_detector}} \qquad (2)$$

In this work, each sequence detectors is capable of identifying up to two sequences. As can be seen, in the proposed transmitter architecture up to four sequences can be detected in 32-bit packet. These numbers, of course, can be changed for different architecture, or different implementation of the sequence detector.

IV. RESULTS

ABC system containing transmitter, receiver and 32-bit registers was designed using VHDL code. The evaluations of the system showed that transmission time of the uncompressed packet is 655ns. In case of a packet with maximal compression rate with 24-bit sequence of identical bits out of 32 bits in the packet, the transmission time was reduced to 295ns, resulting in 55% improvement.

The specific improvement rate in transmission time of each packet depends on the number and lengths of sequences in the packet. The example in Fig. 6 shows the results of simulation of ABC with a series of random packets with various number and lengths of sequences. As can be seen, the compression rate differs among the packets, while the transition time is between maximum 655 nsec and minimum 295 nsec.

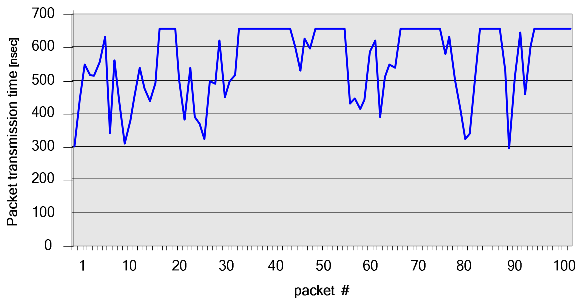

Fig. 6. Receiver final state machine

The effectiveness of ABC compression can be best exemplified by a visual example, while applying the compression to image data. In order to evaluate the effectiveness of ABC, three images were chosen with various differentiation levels (Fig. 7). The images were represented in 32-bit format and the value of each pixel was treated as a separate packet. The data blocks were transmitted through the serial link, while measuring the transmission time with and without the ABC compression.

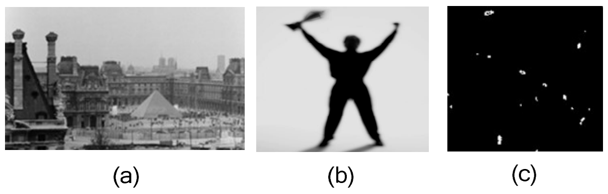

Fig. 7. Images with differentiation varying from high (a) to low (c) used for compression effectiveness evaluation.

Image	*(a)*	*(b)*	*(c)*
Image size [kb]	25	10	6.5
TX original [ms]	0.51	0.20	0.13
TX by ABC [ms]	0.47	0.13	0.06
TX reduction [%]	9	36	54

Table 1. Results of ABC effectiveness evaluation.

The results of ABC compression are presented in Table 1. The transmission time was reduced by 9% to 54% for images with high and low differentiation, respectively. The reduction of the number of transitions by ABC would also reduce dynamic power.

V. SUMMARY

Asynchronous Bit-stream Compression (ABC) was proposed in this paper, based on Level Encoded Dual-Rail protocol. The compression allows a significant saving in the transmission time and power without losing data. The concept of ABC is described in this paper together with the proposed architecture and design considerations. Simulations results are presented for several data patterns with various differentiation rates. Application of ABC reduces the transmission time by 9% to 54% depending on type of source data.

REFERENCES

[1] I. Saastamoinen, T. Suutari, J. Isoaho, J. Nurmi, "Interconnect IP for gigascale SoC", *ECCTD*, pp. 116-120, 2001.

[2] T. Suutari, J. Isoaho and H. Tenhunen, "High-speed Serial Communication With Error Correction Using 0.25 µm CMOS Technology," *ISCAS*, pp. 618-621, 2001.

[3] I.B. Dhaou, E. Dubrova, H. Tenhunen, "Power efficient inter-module communication for digit-serial DSP architectures in deep-submicron technology", *Multiple-Valued Logic*, pp. 61-66, 2001.

[4] A. Morgenshtein, I. Cidon, A. Kolodny, R. Ginosar, "Comparative Analysis of Serial and Parallel Links in Networks-on-Chip", *SoC*, Finland, pp. 185-188, 2004.

[5] R. Dobkin, I.Cidon, R.Ginosar, A.Kolodny, A. Morgenshtein, "Fast Asynchronous Bit-Serial Interconnects for NoC", *CCIT* #529, Technion.

[6] M.T. Dean, T. Williams et al. "Efficient Self-Timing with Level-Encoded 2-Phase Dual-Rail (LEDR)," *ARVLSI*, pp. 55-70, 1991.

[7] S. Ogg, B. Al-Hashimi, "Improved Data Compression for Serial Interconnected Network on Chip through Unused Significant Bit Removal", *VLSID*, pp. 525-529, 2006.