Asynchronous ADC/Microprocessor Interface

Tao Zhu, Xiaohua Kong and Radu Negulescu
Dept. of Electrical and Computer Engineering
McGill University, Montreal, Quebec, Canada
{taozhu, kong, radu@macs.ece.mcgill.ca}

***Abstract*-System-on-chip technology raises challenges when smooth interfacing between different functional systems is required. This paper presents the design of an asynchronous interface between irregular sampling and asynchronous microprocessor. The target of this design is to interface an asynchronous microprocessor with front-end devices with high-speed and low-power operation modes depending on the sampling rate. We use GasP FIFOs [1], as well as several custom-designed modules to implement this interface. Simulation results in Hspice for a CMOSP18 technology shows stable operation of up to 3.1 Giga Data Items / sec (GDI/s).**

I. INTRODUCTION

The development of system-on-chip technology provides the opportunity to integrate multiple functional systems on a single chip. The most demanded properties for on-chip systems are low power, high speed, and immunity to variations of temperature and power supply, because many of them are battery powered and must work successfully in different environments. Challenges often arise when smooth interfacing between different functional systems is required [2].

The design presented in this paper aims to interface an asynchronous microprocessor [3] with front-end devices. We present our experiment on interfacing with an irregular sampling asynchronous A/D converter (A-ADC) [6]. In this design, we use a GasP FIFO and interface blocks, which can sample digital signal from the front-end A-ADC and send data to the back end asynchronous microprocessor in high speed. We harness the benefits of asynchronous circuits' low power and high speed, particularly GasP [1] circuits and the single-track [5] handshaking protocol. Furthermore, our design provides different data loading modes to reduce communication activity between the front-end to the microprocessor for energy saving when the input data rate is low.

Our design can accommodate an input data rate up to 3.1GDI/s.

II. BACKGROUND

Several modern asynchronous circuit families use handshake channels to synchronize data transmission locally between blocks. A handshake channel is a bundle of data and control wires between a sender and a receiver, complying with a communication protocol.

Data transmission between two GasP stages is shown in Fig. 1. GasP circuits, proposed by a team from SUN Microsystems Laboratories [1], were aggressively optimized for high operating speeds by reducing asynchronous pipeline control to its minimal form. GasP circuits use the single-track [5] handshake protocol that minimizes switching activity by having only two control transitions per handshake cycle and by supporting simple level-dependent logic in the sender and the receiver. More importantly GasP circuits support modular design by enforcing delay constraints only inside individual modules and along handshake channels which bundle data and control information.

A GasP stage (shown in Fig. 1) consists of an asynchronous control unit (GasP unit) and the data path under its control. Wires between GasP units are bi-directional and shared by two adjacent GasP units; these wires are named state conductors. A keeper (weak inverter loop) on each state conductor stores the state of the stage (full or empty). Triangles inside the blocks in Fig. 1 represent ports at the interface of GasP units. Black triangles indicate that the port is activated and white triangles indicate that the port is inactivated. When all the ports of a GasP unit are activated, the unit issues a pulse to the datapath and the data is latched in.

A new class of asynchronous Analog-to-Digital Converters (A-ADC) has been proposed recently [6]. These converters realize an irregular sampling of the analog

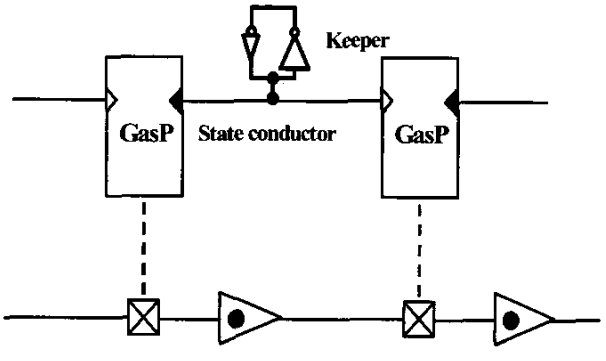

Fig. 1. GasP pipeline (After [4])

0-7803-8322-2/04/$20.00 ©2004 IEEE.

signal to process according to its amplitude variations. Unlike classical Nyquist sampling, samples are not regularly spaced out in time, because the sampling rate depends on the signal variations: the sharper the signal, the closer the samples. Nevertheless, sampling is accurate because the time of the samples is recorded along with the samples themselves. This asynchronous A-ADC "offers a significant reduction in power consumption, complexity, area, and electromagnetic emissions" [6], thus can be used "in classical signal processing systems as a low-power conversion front-end". The authors highlighted that such an A-ADC design provides an opportunity to develop a fully asynchronous systems combining irregular sampling and asynchronous processing.

An asynchronous microprocessor design is proposed in [3] for high-speed, low power embedded applications. Compared with its synchronous counterpart using a similar datapath, the asynchronous microprocessor reduces the energy dissipation to approximately half in high-speed applications profiled to the SPECint92 benchmark.

III. Overview of High-Speed FIFO

The purpose of the interface design is to transfer data smoothly from an A-ADC to an asynchronous microprocessor.

Digital signals from the A-ADC are transmitted through a handshaking channel using 4-phase handshaking protocol. On the output side, the microprocessor implemented with GasP logic uses single-track handshaking.

We use a GasP FIFO and several custom designed blocks to implement the interface between A-ADC and microprocessor. Fig. 2 illustrates the signals at the interface of the FIFO and Fig. 3 shows the schematic of the FIFO. Only one bit of the datapath is shown in Fig. 3, but a wide range of datapath widths can be accommodated with a similar design because the buffer can be resized or cascaded with a large delay margin. Signals *Fifo_control <0:3>* actually are state conductors of GasP units and they are connected to the circuit that sets the state of the FIFO (not shown). For simplicity, keepers are not shown.

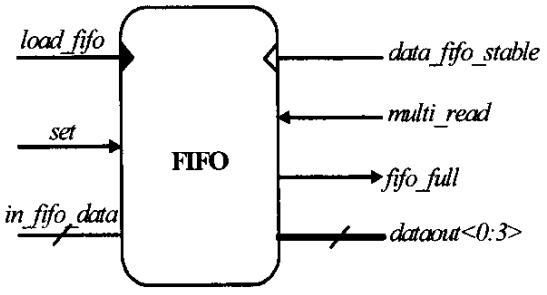

Fig. 2. Interface signals of the FIFO

Internal FIFO stages are implemented using GasP circuits, which allow data items to pass through the FIFO within a theoretically minimal operating cycle of 6 inverting CMOS delays per stage.

In order to communicate with the A-ADC using handshaking, a special stage, termed ADC/FIFO stage, is designed to interface two different handshaking protocols. Data on different stages can be loaded at the same time if necessary. Initially, all the GasP control units are ready to transfer data. When there is a data item available from the A-ADC, the ADC/FIFO stage is activated and data is latched. Consequently, the second stage, a GasP stage, is activated and the data item is passed to the second stage. Then the third stage is activated and so on. Meanwhile, if the microprocessor is ready to load data from the A-ADC (*data_fifo_stable* is set high by the microprocessor), a data item is loaded into the last register in the register file, which is reserved for the data sampling latched from the FIFO (the address is '11111'). In the case that the word-width of the register file is N times of the word-width of the A-ADC, the microprocessor can load several data items simultaneously into the register. (In our design, the bit-width of the microprocessor is 4 times that of the A-ADC). In this mode, the microprocessor sets the read mode signal *multi_read* high. The microprocessor loads the data only when the FIFO raises the *fifo_full* signal. Another usage of the multi-read mode is read data to the microprocessor in burst. Notice that usually the A-ADC is the bottleneck of performance, multi-read mode is useful to reduce the communication between the microprocessor and FIFO even the word-width of the A-ADC and microprocessor are the same.

The master clear set empties all the stages of the FIFO by setting all the Fifo_control lines. While set is high, new samples cannot be loaded.For simplicity, reset circuits are not shown in the figures

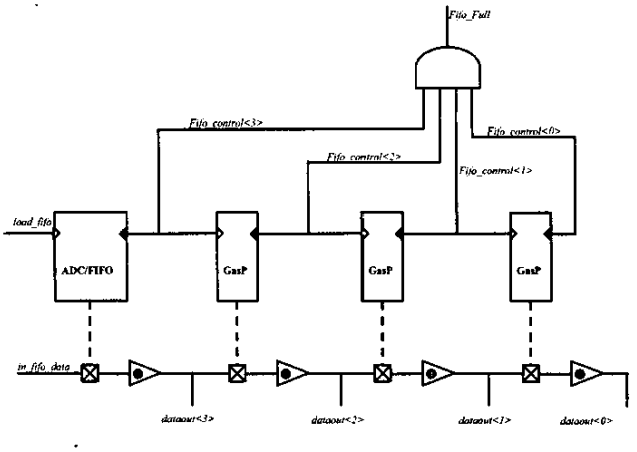

Fig. 3. Structure of FIFO

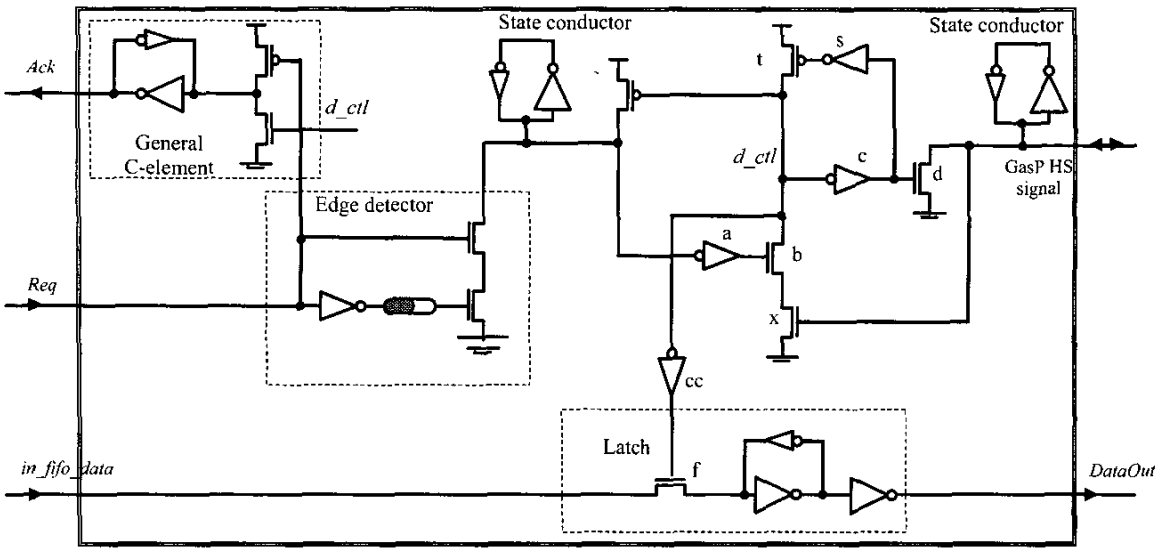

Fig. 4. Design of ADC/FIFO stage

IV. ADC/FIFO Interface

The detailed design of the ADC/FIFO stage is shown in Fig. 4. When data is ready at the input of the FIFO, the A-ADC sets *req* high. An edge detector in the ADC/FIFO stage catches the rising edge of the *req* and generates a falling edge on the handshake track. Since the transistor x is initially conducting, the internal node *d_ctl* is pulled down. The "self-reset" circuit of the GasP stage (inverter c, inverter s and PMOS t) will set *d_ctl* high again after three inverter delay, thus generating a pulse on the NMOS transistor f and latching *in_fifo_data* from the A-ADC.

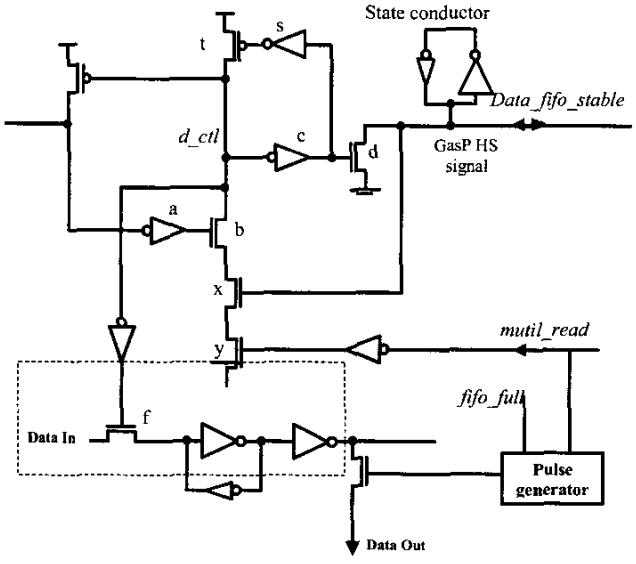

Fig. 5. Design of FIFO/Microprocessor interface (without set)

Meanwhile, the output *ack* is set high by signal *d_ctl*. A general C-element [7] is employed to generate an acknowledge signal in 4-phase handshaking protocol. The falling edge of the *ack* will be generated by the falling edge of the *req* to fulfill a cycle of 4-phase handshaking.

On the other side of the ADC/FIFO stage, single-track handshaking protocols are used, thus the protocol conversion is fulfilled by this stage.

V. FIFO / Microprocessor Interface

Since the microprocessor is implemented with GasP circuits, no protocol conversion is necessary. The microprocessor can load data from the ADC with low latency by register read operations.

Fig. 5 shows the interface between FIFO and microprocessor. When *multi_read* is low, a data item is loaded to the register file as soon as it reaches the last stage of the FIFO [Fig. 3]. When *multi_read* is low, the last stage of the FIFO is blocked (NMOS transistor y is off). Only when FIFO is full, a pulse is generated by the pulse generator and latches all the data items in the FIFO to the register. After that, FIFO is set to initial state (not shown in Fig. 5). Since in mutil-read mode, the microprocessor only communicates with the FIFO when FIFO is full, lower power consumption is achieved.

VI. Conclusion

In this paper, we have presented an interface circuit design to integrate an asynchronous microprocessor with its front-end devices. An irregular sampling A-ADC is used as the front-end device to illustrate our design. Our design

provides flexibility not only on the data transmission rate, but also on the power consumption of the circuit.

HSPICE simulations of the circuit implemented with CMOSP18 technology at 1.8V show that the throughput of GasP FIFO for one stage is 3.1 GDI/s with a latency of 234.6 ps. The rate of downloading 32-bit words from the FIFO is 8 ns, which allows for continuous storage of the samples.

The simulation also shows that the circuit operates robustly under supply voltages in the range of 1.0V to 2.6V and operating temperatures in the range of −35 to 100°C.

This experiment demonstrates the low-power advantage in asynchronous design, the feasibility of high rate continuous sampling in asynchronous implementation, and a robust interface between an A-ADC and an asynchronous microprocessor.

REFERENCES

[1] Sutherland, I.; Fairbanks, S. "GasP: a minimal FIFO control". In Proc. Int. Symp. on Advanced Research in Asynchronous Circuits and Systems, 2001, pp. 46-53.

[2] Rowson, J.A.; Sangiovanni-Vincentelli, A. "Interface-based design", In Proc. Design Automation Conference, June 9-13, 1997, pp.178 – 183

[3] Lao, X.; Saadallah, N., Zhu, T. "Low-power asynchronous RISC microprocessor", Paper submitted to the 2nd annual IEEE Northeast Workshop on Circuits and Systems (NEWCAS' 2004).

[4] Ebergen, J.; "Squaring the FIFO in GasP", Asynchronous Circuits and Systems, 2001. Seventh International Symposium on , 11-14 March pp. 194 -205

[5] Berkel, van K.; Bink, A. "Single-track handshake signaling with application to micropipelines and handshake circuits". In Proc., Int. Symp. on Advanced Research in Asynchronous Circuits and Systems, 1996, pp. 122-133.

[6] Allier, E.; Sicard, G.; Fesquet, L.; Renaudin, M. "A new class of asynchronous A/D converters based on time quantization". In Proc. Int. Symp. on Advanced Research in Asynchronous Circuits and Systems, 2003, pp. 196-205.

[7] Sutherland, I.E. "Micropipelines", Communications of the ACM, Volume 32, No.6. June 1989, pp.720-738.