A Fully Self-Timed Bit-Serial Pipeline Architecture for Embedded Systems

Achim Rettberg, Mauro Zanella, Christophe Bobda, Thomas Lehmann
University of Paderborn, Paderborn / Germany
Email: achim@c-lab.de, zanella@mlap.de, bobda@upb.de, Thomas.Lehmann@torkin.de

1 General Overview

Area minimization, low power and high performance are objectives to be reached in chip design. Bit-serial architecture offers a great advantage in comparison with bit-parallel architectures as regards area minimization. One field of application of such an architecture is e.g. signal processing in terms of digital filters or digital controllers. These algorithms may be realized in hardware by means of the proposed architecture; this requires only small chip area and an equally small number of input and outputs pins, thus reducing the size and the complexity of the printed circuit. The speed of the bit-serial processing is high enough for the application domain in question. E.g., in an electrical motor-current control there are the delays of input/output converters (e.g., A/Ds and D/As) and those resulting from the inertia of the motor.

In synchronous design, the performance of these architectures is affected by the long lines which are used to control the operators and the gated clocks. The architecture described in this contribution avoid a long control lines by a local distribution of the control circuitry on the operator level. To our knowledge, this is the second paper detailing the implementation of a fully interlocked synchronous architecture after that described in [1].

2 Self-Timed Bit-Serial

As a rule, mapping huge dataflow graphs to the target architecture requires a central control unit. This unit provides synchronization between the operators and controls data distribution and storage within the realized circuit. Usually, the control unit is realized by a finite-state machine. In combination with an increase in chip area it may not be possible to implement a central control unit. Therefore, the aim is to realize all the control signals locally. In the architecture presented, the central control unit is replaced by local control elements. These are synchronized and interlinked by handshake wires. Therefore the required control wires are local. As mentioned before, the handshake mechanism used in the architecture is similar to the one that is used in bit-serial asynchronous architectures (cf. [3]).

The central control unit may be conceived as a simplified counter that, at a given time, triggers defined actions in given time slots. This reflects the paradigm of synchronous design. Taking into account a so-called one-hot implementation, it is possible to map the counter to a defined shift register with a special marker that is "pushed" through the register. When the marker reaches a defined position within the shift register it initiates actions in the circuit. The main idea of this architecture is to map the counter (control unit) to a shift register in the data path. The modified operators in the data path recognize the marker of the counter. Therefore, the marker can pass through the operations unhindered and unmodified. It is attached to the data and controls the data processing in the data path. A data package contains the control marker and the processed data. Additionally, a gap may be included between the control marker and the data. Such a gap can be regarded as a separator. A valid data package is shown in Figure 1.

The bit order of the data requires LSB[1] first. To recognize the control marker in the data path, so-called scanners are needed (see Figure 1). These scanners identify the marker and activate the component control. The time factor is mapped to a position within the implemented data path; thus we know exactly the length of the control marker, the gap and the data and can therefore detect where the data are processed when a marker signal reaches a certain position in the data path. The distance between the control information (the marker) and the operations is predefined with the operations to be affected only by the control marker of the actually processed data. Consequently, the distance between control wires and the controlled elements is known (Figure 1).

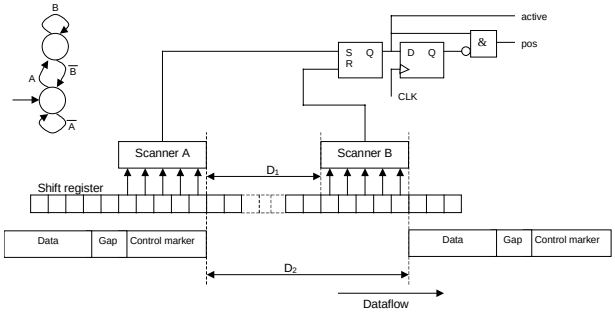

Figure 1. Two scanners with automaton

[1] LSB = least significant bit

1530-1591/03 $17.00 © 2003 IEEE

To avoid problems with identical bit patterns, we use two scanners (see Figure 1) that realize an insensitive automaton against those patterns. The first scanner activates the automaton and the second ends the activation phase. The automaton goes into idling state *(wait)*. An edge detection allows retaining the original scanner signal. Both edges can be used as a control signal depending on the required functionality.

It is necessary that their exist a minimum distance between the scanners, as well as between the data packages, to avoid data conflicts. All control signals of an operation with valid data are activated by a control marker of the corresponding data package, resulting in a independently data movement along the data path.

For this purpose, the fastest path in the dataflow graph is blocked by a so-called stall wire until all other necessary data packages have arrived at a specific synchronization point. This functionality is realized by a component named synchronizer (see section Figure 2). Due to the defined length of the data packages the length of the stall wire is locally limited.

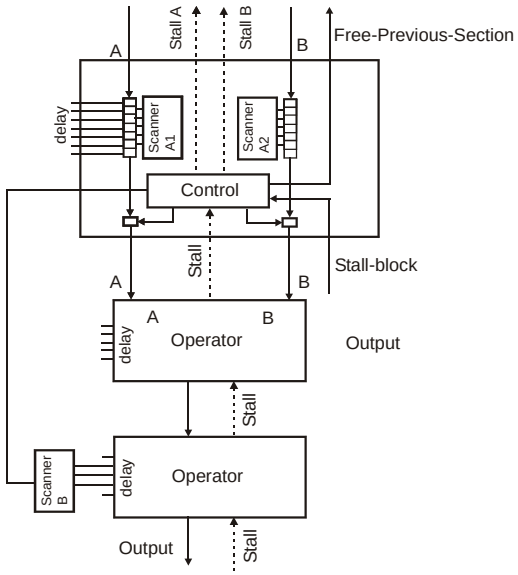

Figure 2. Synchronizer realization

The synchronizer is responsible to block the inputs that arrived earlier until all inputs are ready. The stall wire in Figure 2 realizes the blocking of the corresponding inputs. The valid data at the inputs are recognized by the control marker that is stored before the gap and the data in a data package. Here, the control marker can be interpreted as a synchronization marker.

Furthermore, we have introduced a *stall-block* signal in the architecture. "*Stall-block*" means that a complete block with scanners on the input and output sides of an operator is blocked. The counterpart of the *stall-block* is the *Free-Previous-Section* signal. In the case of the latter, the block can be used. Additionally, realization of the synchronizer requires the inputs to be synchronous and the block to be free (*Free-Previous-Section* is *true*). In this case the synchronizer reads the data packages and sets the *stall-block* signal. This leads to a communication and evaluation of scanner signals between the different synchronizers (see Figure 2).

3 Realized Example: PI controller

For test purposes we have realized a PI (Proportional and Integral) controller [2] on an FPGA. For this purpose it was necessary to transform the continuous integrator (s) into a discrete-time integrator (z). Applying the trapezoidal integration method for a given sampling period T, the PI compensator in a discrete form is given by:

$$y_t = u_t K_P + K_I \left(y_{t-1} + \frac{T}{2}(u_t + u_{t-1}) \right)$$

For this implementation we have used the target platform is the FPGA module of the RABBIT system [4]. The PI controller occupies 7 % of the total amount of FPGA slices. The sampling time of the controller is approximately 500 kHz. This may be slower than a parallel implementation, but much faster than required by such a system.

4 Conclusion and Outlook

The presented architecture has the peculiar feature of being self-timed and comprising a fully interlocked pipelining structure which aims at controlling the different computational paths of a system design.

One example is the automotive industry where performance, space, cost, size, and weight are of vital importance, the main feature of this architecture.

References

[1] H. M. Jacobson; P. N. Kudva; P. Brose; P. W. Cook; S. E. Schuster; E. G. Mercer; C. J. Myers. "*Synchronous Interlocked Pipelines*", 8th International Symposium on Asynchronous Circuits and Systems (ASYNC 02), Manchester, UK, April 2002.

[2] St. M. Shinners. "*Modern control system theory and design*", John Wiley & Sons, New York/Chichester, 2nd ed., 1998.

[3] A. Rettberg; A. Hennig; B. Kleinjohann. "*Re-Configurable Multiplier Units of the Asynchronous FLYSIG Architecture*", 10th NASA Symposium on VLSI Design, Albuquerque, NM, USA, March 2002.

[4] M. Zanella; M. Robrecht; Th. Lehmann; R. Gielow; A. de Freitas Francisco; A. Horst. "*RABBIT: A Modular Rapid Prototyping Platform for Distributed Mechatronic Systems*", SBCCI 2001 - XIV Symposium on Integrated Circuits and Systems Design, Brasília, Brazil.