EFFICIENT AND SAFE ASYNCHRONOUS WAVE-PIPELINE ARCHITECTURES FOR DATAPATH AND CONTROL UNIT APPLICATIONS

O. Hauck, M. Garg, and S. A. Huss

{hauck | garg | huss}@vlsi.informatik.tu-darmstadt.de
phone: +49 6151 16 3983 fax: +49 6151 16 4810
Integrated Circuits and Systems Lab Departments of CS and EE
Darmstadt University of Technology Alexanderstr. 10 64283 Darmstadt Germany

ABSTRACT

This paper presents a generalization of a previously proposed asynchronous wave-pipeline architecture. Four-phase and two-phase communication units supporting more than one wave in the logic are proposed. General feedback structures are then outlined. Simulations from a 16-bit add-and-shift ring demonstrate their feasibility. The same architecture is applicable for both datapath and control enabling the realization of complete high-throughput asynchronous systems.

1. INTRODUCTION

Clocking today's high performance synchronous chips is a complex task. Not only does the design of the clock nets represent a major part of the whole design, moreover, clocking may easily dominate the chip's power budget [1]. Furthermore, the higher the clock speed, the less the relative amount of the clock period that is usable for computation.

Asynchronous design may offer solutions to some of the problems found in synchronous systems. Abandoning the global clock in favor of local communication via a protocol offers three major potential advantages: first, if every component adheres to the same protocol, large systems can easily be built and modified in a plug-and-play fashion. Second, redundant transitions are avoided, because computation only takes place when data calls for it thus saving power. Finally, and this has turned out to be hard to realize, asynchronous systems could outperform synchronous systems, because fast data does not have to wait for a clock edge just to be synchronized with slow data.

Well-known problems with the asynchronous approach are the presence of hazards and the overhead in implementing the protocol. While the first two of the mentioned advantages have been successfully demonstrated [2], asynchronous chips with superior performance are the exception [3]. Because it is our belief that the asynchronous approach is of more interest to designers when it has to offer something at frequencies where synchronous clocking becomes really difficult, we emphasize high throughput systems.

This paper improves on [4] in that it introduces two-phase operation, more than one data wave active in the logic at any time, and general feedback.

2. ASYNCHRONOUS WAVE-PIPELINES

Traditional asynchronous circuits communicate via a handshake protocol with each other implying *request* and *acknowledge* signals, the first one triggering computation when data is valid, the latter one signalling that the computation has been completed and new data can be received. There are four-phase and two-phase variants of this protocol available.

Our basic observation is that an acknowledge signal is actually not necessary for asynchronous operation, even though it is often found in asynchronous designs. The reason for this is that the presence of this signal gives rise to the property of *elasticity* of operation, i. e. the ability to buffer. This aids in the aforementioned plug-and-play design of large systems, because one does not have to worry about whether the outputs are correctly saved. Whenever an elastic component cannot accept new data, blocking occurs and may cause a domino effect from the core of the chip up to the pad inputs. One reason why elastic components hinder performance is because it may take any time from a few ns to, say, a second until blocking is resolved; this in turn requires static memory which is slower than dynamic memory. If, on the other hand, we were always to know when results can be accepted by a following stage, we could get rid of the acknowledge signal. We believe that often this can be done at design time, or at least the use of elastic components can be minimized and restrained from the core where these would slow down operation.

0-7695-0104-4/99 $10.00 © 1999 IEEE

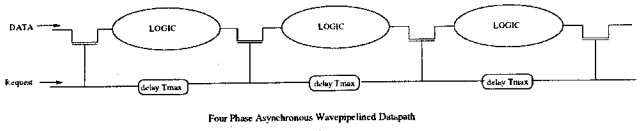

Figure 1: Four-phase asynchronous wave-pipeline.

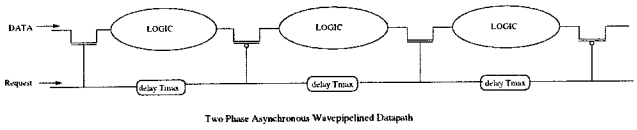

Figure 2: Two-phase asynchronous wave-pipeline.

Our approach is therefore to skip the acknowledge signal and just maintain the request signal, which indicates valid data and acts much like a local clock. Our basic building blocks are by default inelastic and we have to carefully consider the impact of this on the system level.

The second unique characteristic of our approach is the incorporation of *wave-pipelining* into an asynchronous context. Wave-pipelining [5] has been known for thirty years as an exotic design technique aimed at increasing the throughput of a pipeline without additional registers or latches. The basic idea is to have more than one data wave simultaneously active in combinational logic whereas in a conventional pipeline individual data waves are separated by register or latch stages. The input capacitance of logic gates provides for dynamic storage of the data. The central problem in a wave-pipeline is to make the data waves propagate coherently through the logic so that no fast wave catches up a preceding slow wave which would result in data loss. To achieve this, one has to balance the path delays that are influenced by the logic depth, actual and previous input data, logic style, process, temperature and voltage variations. The striking feature of a wave-pipeline is that the throughput is not determined by the worst-case delay of all stages in the pipeline but mainly by the *difference* Δ between the shortest and the longest path delay. Among the challenges to be met in a synchronous realization is the difficulty to sample the data at the output because a specific phase relationship has to be maintained between input and output clock. In contrast, sampling is easier in an asynchronous context where data and associated request signals arepropagated together.

2.1. Four-Phase Operation

The general structure of the proposed pipeline is shown in fig. 1. Logic stages are separated by switches that may consist of simple pass transistors or of transmission gates, depending on the logic style of the stages. The request line consists of a chain of delay elements matched to the worst-case delay of the corresponding logic stage. In order to control the corresponding switch, the request line is tapped between the delay elements.

The operation is as follows. Data waves accompanied by *pulses* on the request line enter the pipeline. After propagation through a logic stage the individual bits of the data word are skewed by a time value bounded by Δ. Because any further propagation of the data wave would increase the skew above Δ, a so-called *wave latch* consisting of a simple switch aligns the data bits by waiting for the slow bits. This occurs in the low phase of the request pulse and takes Δ time. When all bits have arrived the wave latch opens with the high phase of the request pulse and loads the data into the following stage. The width of the high phase is denoted by $\delta_{4-phase}$. Note that the frequency achievable on the request line is determined by $\Delta + \delta_{4-phase}$ and not by the worst-case delay of the delay elements. The delay elements in the request line are effectively wave-pipelined as well.

While it was shown in [4] that throughput rates in the Giga range are achievable, the handling of the pulsed request line may cause a problem when wave latches are used excessively. Wave-pipelining relaxes the requirements on the request line without decreasing the throughput at a cost of increased effort for balancing the logic stages.

Note that while the architecture and the operation of the pipeline is similar to a synchronous pipeline, it is quite actually asynchronous as the request is fed in at the input only as opposed to a clock driving all register stages in parallel.

2.2. Two-Phase Operation

Another possibility to relax the request line load is to abandon the pulsed operation in favor of a level-based scheme. Fig. 2 shows the two-phase pipeline. The only difference to the four-phase instance is the use of alternating positive and negative switches. However, the relationship between data and request is now more complex. Subsequent data words are now associated with *alternating* high and low levels on the request line. The corresponding level on the request line enters the pipeline first; the data follows at the *end* of this level. After propagating through the logic, the data bits are skewed but still lie "inside" the level. When the level is low,

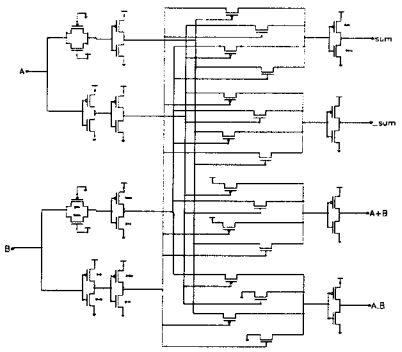

Figure 3: Conditional cell in double pass transistor logic.

then data flows through the open P-type switches. The data is not subject to setup delay, because the switch is already open when data arrives. The same is true for high levels and N-type switches. When data with low-level request arrives at a positive wave latch — similarly, data with high-level request arrives at a negative wave latch —, the skew is maximal and the wave gets aligned while passing through the switch (wave latch). Due to the propagation delay $\delta_{2-phase}$ of the switch, we have then to delay the request signal by $\delta_{2-phase}$ to make sure that the data is again "inside" its level.

In analogy to the four-phase case, the throughput is mainly determined by $\Delta + \delta_{2-phase}$, where usually $\delta_{2-phase} > \delta_{4-phase}$ due to the asymmetric behaviour of the two-phase scheme. When compared to the four-phase protocol, the power dissipation of the request line has been halfed.

3. ASYNCHRONOUS WAVE-PIPELINES WITH FEEDBACK

Feedback is essential for a complete design methodology either for iterative datapaths, e. g. SRT ring division, or finite state machines. We illustrate four-phase and two-phase feedback operation with the example of a ring that sums up its input and shifts right before each addition. This is exactly the kernel of the distributed arithmetic algorithm for computing scalar products. The adder at hand is a 16-bit conditional sum adder balanced for $\Delta \approx 400$ps. Fig. 3 shows the conditional cell implemented in double pass transistor logic. The following simulations were performed on schematics for a 0.6μm CMOS technology.

3.1. Four-Phase Operation

Fig. 4 shows the four-phase feedback architecture. Note how the feedback is synchronized in the wave latch.

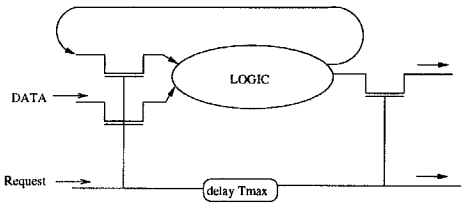

Figure 4: Four-phase setup with feedback.

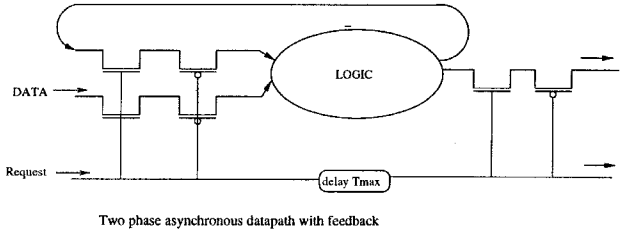

Figure 6: Two-phase setup with feedback.

Fig. 5 shows the request pulse together with the skewed data. The ring consumes data at 1.0GHz and has five waves active.

3.2. Two-Phase Operation

In analogy to fig. 4 and 5, we visualize in fig. 6 and 7, respectively, the two-phase setup and skewed data wave. There are four active waves at a throughput of 800MHz. We claim that with a slight modification the described feedback structures can be made work at varying rates even with non-integer number of waves.

3.3. Implications for FSMs

The example of the preceding section was a datapath application, but its structure is that of a FSM. Therefore the circuits in fig. 4 and 6 can be used as controllers with the logic computing next state and output values. In the following the controller operation of a structure

Property	Two-phase	Four-phase
Throughput	$1/(\Delta + \delta_{2-phase})$	$1/(\Delta + \delta_{4-phase})$
Protocol	data not context-free uses logic levels levels more efficient	data context-free uses narrow pulses pulses inefficient
Logic	requires small Δ	does not require small Δ

Table 1: Comparison of two-phase and four-phase architectures.

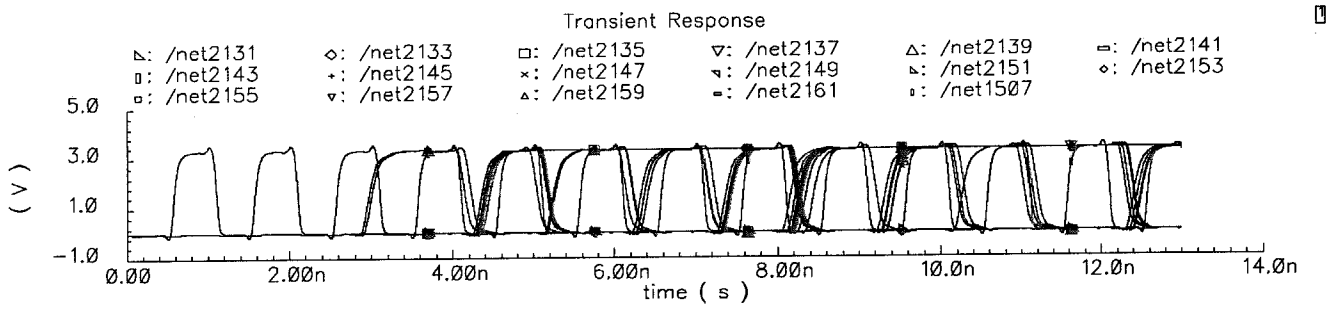

Figure 5: Skewed data wave just before the four-phase wave latch in fig. 4.

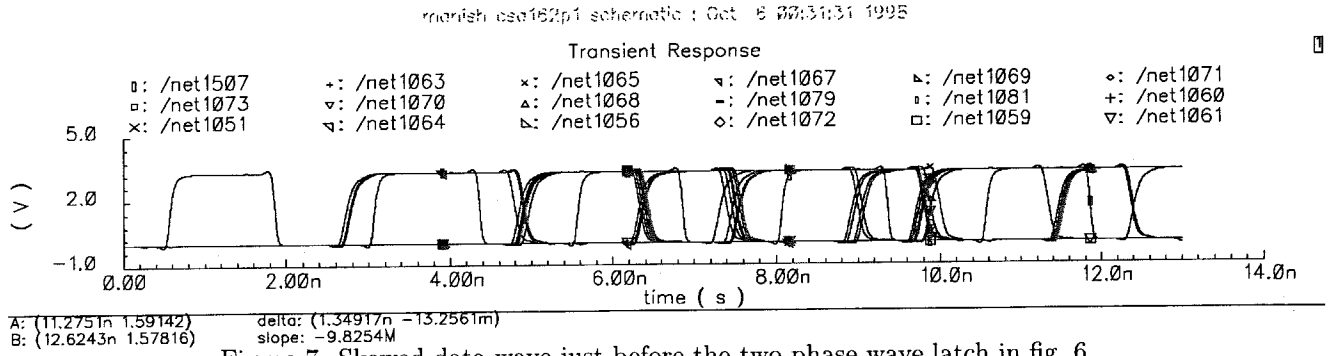

Figure 7: Skewed data wave just before the two-phase wave latch in fig. 6.

as in fig. 4 or fig. 6 will be detailed. We restrict ourselves here to the case of one data value present in the feedback, as it is usually the case with controllers, both synchronous and asynchronous. However, as the previously shown multiply-and-accumulate unit demonstrates, it is possible to use wave-pipelining in feedback configuration and thereby constructing *wave-pipelined controllers*. The practical value of these has to be investigated in further studies.

Four-phase controllers as in fig. 4 and two-phase controllers as in fig. 6 work similarly, so, for simplicity, consider fig. 4 only. The operation starts with power-on so the first issue is initialization. The initialization of the feedback path can be performed with any scheme used in synchronous circuits, e. g. switched pull-down or weak permanently-on pull-down. The second issue is state storage. As long as no request shows up at the primary input, the computed next state is dynamically stored at the input of the wave latch. If it cannot be guaranteed that there is frequent activity on the primary input then the feedback path has to be made static by weak feedback inverters. It has to be emphasized that this type of controllers inherently avoids all sorts of hazards because the feedback flow is interrupted by the wave latch. Yet this structure is asynchronous: there is no need for a global clock, there are no clocked storage elements, and it idles under absence of external inputs. The simple scheme for synchronization of primary inputs and feedback path imposes one limitation on this type of machines. As the request line is shared between primary inputs and feedback and powered by the primary inputs, the logic can only evaluate — computing outputs and next state — under primary input activity. The consequence is that state sequences independent of primary inputs, i. e. pattern generation, cannot be done. But this can be achieved by a machine with no primary inputs at all, that, once initialized, produces the desired pattern sequences.

4. REFERENCES

[1] D. W. Dobberpuhl, "Circuits and Technology for Digital's StrongARM^{tm} and ALPHA Microprocessors," *Proceedings 17th Conference on Advanced Research in VLSI*, pp. 2–11, 1997.

[2] N. C. Paver, P. Day, C. Farnsworth, D. L. Jackson, W. A. Lien, and J. Liu, "A Low-Power, Low Noise, Configurable Self-Timed DSP," *Proceedings Fourth International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 32–42, 1998.

[3] T. E. Williams, "A Zero-Overhead Self-Timed 160-ns 54-b CMOS Divider," *IEEE Journal of Solid-State Circuits*, vol. 26, no. 11, pp. 1651–1661, 1991.

[4] O. Hauck and S. A. Huss, "Asynchronous Wave Pipelines for High Throughput Datapaths," *Proceedings 5th IEEE International Conference on Electronics, Circuits and Systems*, pp. 283–286, 1998.

[5] L. W. Cotten, "Maximum-rate pipeline systems," *1969 AFIPS Proc. Spring Joint Computer Conf.*, vol. 34, Montvale, NJ: AFIPS Press, pp. 581–586, May 1969.