Synchronous clocked and self-timed pipeline configurations

Christoph Heer
SIEMENS AG
Corporate Research and Development
D-81730 Munich, Germany
e-mail: heer@mess14.zfe.siemens.de

Abstract

Self-timed pipelines and data path architectures are often quite similar to their synchronous clocked counterparts. In this contribution it will be shown how the delays of different building blocks of the data path like control, storage and functional logic units contribute to the global data throughput rate. This analysis will give a new insight in different configurations of self-timed pipelines and faciliate their understanding by the comparison with the synchronous clocked architectures.

1. Introduction

The most significant difference between synchronous clocked and self-timed circuits is given by the different dependence between data and control signals. In synchronous clocked systems each operation of an arithmetic or logic unit has to be finished within a time slot given by the overall clock signals. Data signals have to be stable at latching time. In self-timed circuits latching will be caused by the data signals itself. The end of the operation of an arithmetic or logic unit is indicated within the output data and will initiate the data storage. While in synchronous circuits the data signals are required to be stable at the edge of the control signals in self-timed circuits the data signals will cause the control signals to switch.

But despite these fundamental difference there are quite similar architectures as far as data paths and pipelines are concerned. In the following, three different granularities of synchronous clocked and self-timed pipelines are sketched. The contributions of the control and logic delays to the minimum achievable clock period of the synchronous circuit will be compared with their contributions to the cycle time in self-timed circuits. The minimum cycle time is given by the minimum delay between to consecutive requests for operation of one unit.

2. Synchronous clocked data paths

The most common clocking for synchronous clocked data paths is the complementary two-phase clocking scheme (figure 2). Two latches for a master-slave register are used to store the output data of a previous and the input data of the next functional unit (figure 1). After the rising edge of the slave signal input data is applied to the function blocks. The operation of the units has to be finished until the master signal is falling and the output latches are switched into the nontranparent mode.

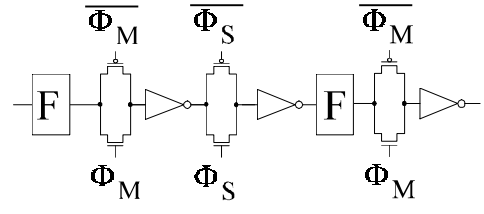

Figure 1: synchronous clocked master-slave register

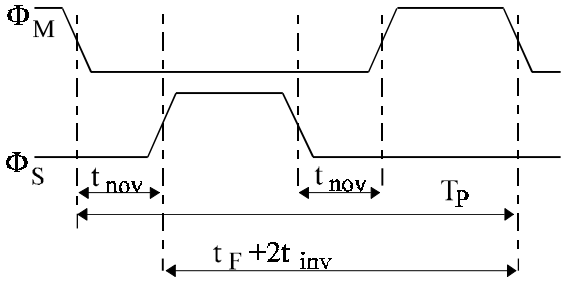

Figure 2: nonoverlapping two-phase clocking scheme

The clock period will give a limit for the computation time of the functional blocks. Taking nonoverlap time and delay of the latches into account the following equation is derived:

$$T_P \geq t_{Fmax} + t_{nov} + 2 \cdot t_{inv}$$

The computation time plus two inverter delays and the nonoverlap time has to be smaller than the clock period.

A modification of the two-phase clocking scheme is often used to reduce the circuitry for data storage. Placing the functional blocks between the latches (figure 3) the inverters for signal amplification are no longer required.

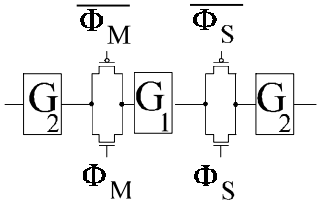

Figure 3a: synchronous clocked pipeline with transmission gate latches

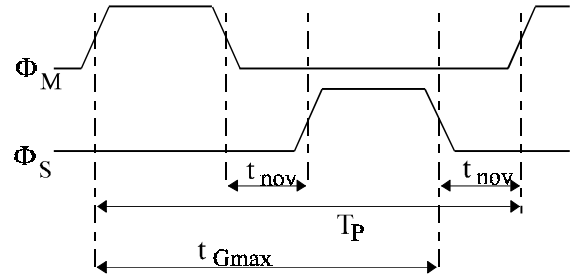

Figure 3b: timing for synchronous clocked pipeline with transmission gate latches

But while the maximum delay allowed for a single functional block remains unchanged, the minimum clock period is now given by the delay of two neighbored blocks: $T_P \geq 2t_{Gi}$

The overlapping three-phase clocking scheme has been introduced in [GDo85] because of the difficult distribution of nonoverlapping clock signals for large clock systems (figure 4).

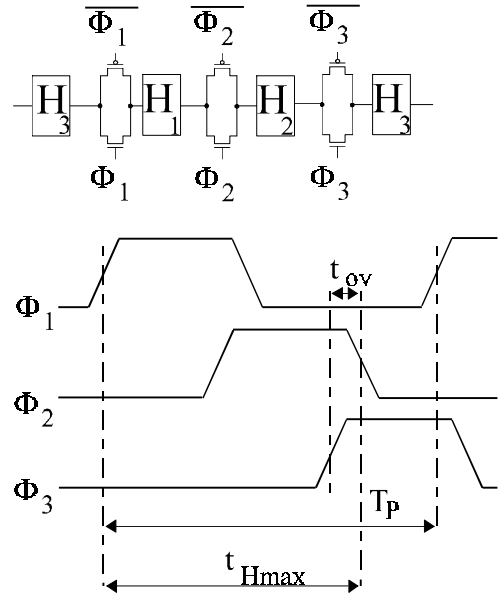

Figure 4: synchronous clocked pipeline with overlapping three-phase clocking scheme

Again each functional block has to evaluate between the rising clock edge of its input latch signal and the falling edge of its output clock signal. The operation of functional block H_1 can be started when phi_1 is set and has to be finished until phi_2 is reset to zero. The maximum delay for each block is therefore two third of the clock period plus one overlap time. But the sum of delays of three neighboring blocks has always to be smaller than the clock period. The minimum clock period for the overlapping three-phase clock is therefore given by three computation times: $T_P \geq 3t_{Hi}$

Three different clocking schemes and pipeline granularities for synchronous clocked data path architectures have been shown. The minimum achievable clock period is in a first order aproximation given by one, two or three times the delay of the functional units. For data storage a register or at least one latch is required per output signal of a functional unit.

3. Self-timed data paths

The control structures of self-timed circuits are based on the muller-c-element. The output of the muller-c-element is set to the logic input values if they are identical. Otherwise the previous output state is stored. This element and an inverter can be used to build a handshake-circuit for communication following a request-acknowledge-protocol (figure 5).

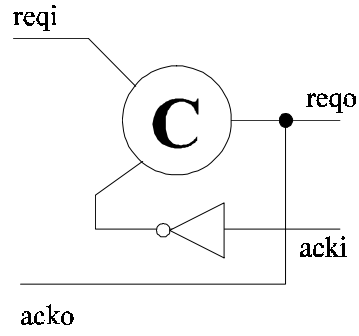

Figure 5: handshake circuit (muller-c-element and inverter)

The following data path architectures are based on the four-phase or return-to-zero protocol. Therefore, also the self-timed functional blocks will follow the four-phase protocol with a start and a ready signal to be set and reset. Operation and reset or precharge of the functional blocks will alternate. We use the differential cascode voltage switch logic [HDG84], which has been shown to be the most efficient implementation of self-timed function blocks [Nie94].

The first configuration of a self-timed pipeline is derived from the two-phase clocking scheme with master-slave registers as shown in figure 1. Each latch of the register is now controled by one handshake-circuit. The circles at the input of the muller-c-element denote the inverters. The self-timed configuration of this pipeline has first been reported by Meng and is called the fullhandshake-configuration [Men88]. The timing diagram of figure 6 is given for explanation of the function.

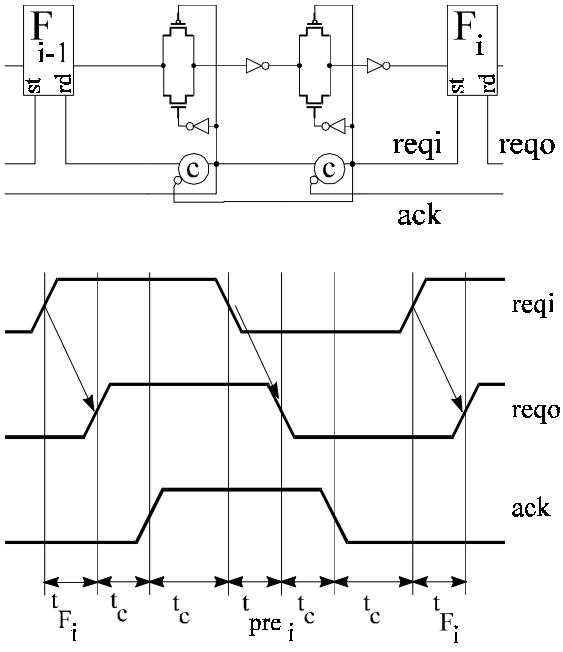

Figure 6: pipelinestage with fullhandshake-configuration amd timimg diagram

When the request signal is set, the operation of F_i is started and the input latch is switched into the nontransparent mode to keep the data stable. At the end of operation the **req**uest **o**ut signal will cause the storage of the output data and the **ack**nowledge signal to be set. When the request signal is set to zero the functional unit F_i is precharged, the output latch is switched back to the tranparent mode and the acknowledge signal is reset. The minimum cycle time between two consecutive requests for one functional block is given by the computation and precharge time of one block plus four handshake transitions:

$$T_{Cycle_{Fullhandshake}} = t_{F_i} + t_{Pre_i} + 4t_C$$

Placing a functional logic block after each latch like in the synchronous case in figure 3 the throughput rate of the self-timed pipeline is reduced too, but also the circuitry for data storage (figure 7).

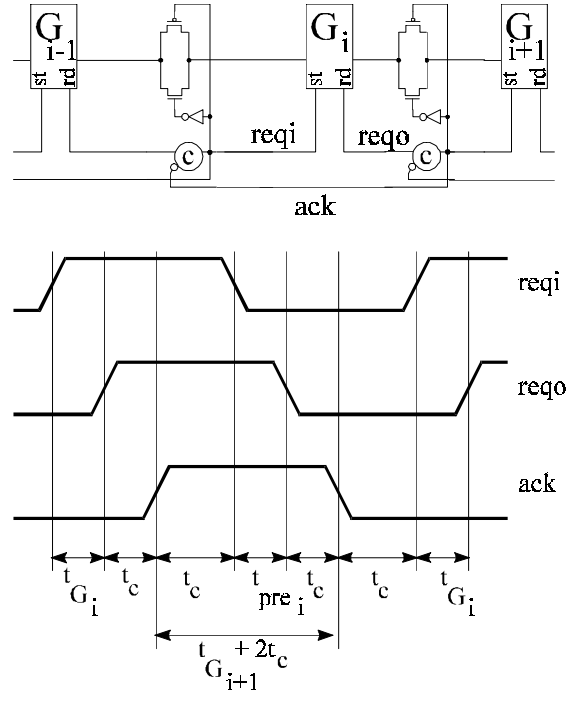

Figure 7: two pipelinestages with halfhandshake-configuration

Assuming the computation time to be larger than the precharge time of a functional block the time critical cycle is now given by two computations in neighboring blocks and four handshake transitions. This configuration is called halfhandshake configuration because each functional block is only active for at most half the cycle time.

$$T_{Cycle_{Halfhandshake}} = 2t_G \quad + 4t_C$$

A slight modification of the pipeline will reduce any latch between the neighboring functional blocks using the inherent storage capability of the DCVS-logic (figure 8). When the computation has been finished the input signals of the DCVS-logic can be

reset to zero without violating the previous computed output values.

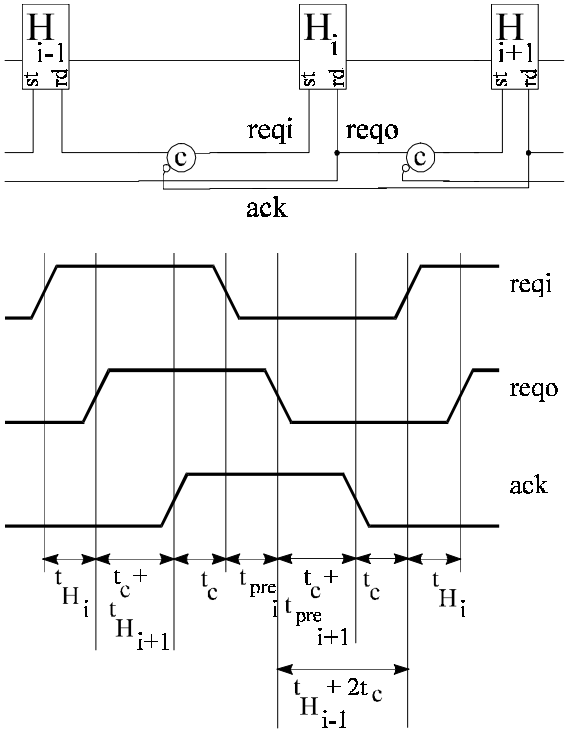

Figure 8: two pipelinestages with thirdhandshake-configuration

Therefore, instead of the start signal of a functional block we use the ready signal to acknowledge the end of operation and the storage of data. This will of course introduce additional delay into the time critical cycle. The time critical cycle of this thirdhandshake configuration is now given by three computations and one precharge phase of neighbored functional blocks and four handshake-transitions (figure 8):

$$T_{Cycle_{Thirdhandshake}} = 3t_H + t_{Pre} + 4t_C$$

Three different handshake configurations for self-timed pipelines have been shown. The minimum achievable cycle time is given by one, two or three times the delay of the functional units assuming the delay of the handshakes and latches to be negligible. For data storage a register, a latch or at least only the inherent storage capability of the DCVS-logic blocks is required per output signal of a functional unit. Therefore the self-timed approach allows for the thirdhandshake configuration a further reduction of storage circuitry compared to the equivalent synchronous clocked pipeline with overlapping clock signals.

4. Self-timed handshake configurations

One-dimensional and two-dimensional self-timed pipelines and arrays have been investigated for very fine grained granularities. One fulladder and a NAND/NOR-gate have been choosen as functional block, well known from pipelined multiplier architectures. For each handshake configuration a pipeline and an array have been built and compared with respect to circuitry and data throughput rate. The reciprocal of the product of transistor count and the minimum cycle time is also given for comparison (table 1 and 2).

	fullhand-shake	halfhand-shake	thirdhand-shake
cycle time τ [ns]	10.1	12.4	15.7
transistor count n	97	66	50
efficiency $1/\tau n$ [normalized]	1.00	1.20	1.25

Table 1: cycle time, transistor count and efficiency for a one-dimensional pipeline

	fullhand-shake	halfhand-shake	thirdhand-shake
cycle time τ [ns]	11.2	15.7	19.8
transistor count n	119	80	64
efficiency $1/\tau n$ [normalized]	1.00	1.06	1.05

Table 2: cycle time, transistor count and efficiency for a two-dimensional array

5. Conclusion

Three different handshake configurations of self-timed pipelines and arrays have been shown in contrast to similar synchronous clocked architectures. Self-timed architectures can be seen as data paths with new control structures. Instead of the overall clock system now data driven local control circuits will cause the storage of the data itself. But the analysis of the minimum cycle time leads to similar contributions of the delays of logic and control units as for the equivalent synchronous clocked pipelines.

Literature

[HGD84] L. G. Heller, W. R. Griffin, J. W. Davis; Cascode voltage switch logic: a differential CMOS logic family; ISSCC Digest of technical papers 1984; pp. 16-17

[Nie94] C. Nielsen; Evaluation of function block designs; Technical report, ID-TR:1994-135, ISSN 0902-2821, DTH Copenhagen; January 1994

[Men88] T. Meng; Asynchronous design for digital signal processing architectures; PhD-thesis; EECS UC Berkeley, California; 1988

[GDo85] L. A. Glasser, D. W. Dobberpuhl; The design and analysis of VLSI circuits; Addison-Wesley 1985; pp. 350-354