A HIGH PERFORMANCE LOW POWER MESOCHRONOUS PIPELINE ARCHITECTURE FOR COMPUTER SYSTEMS

By

SURYANARAYANA BHIMESHWARA TATAPUDI

A dissertation submitted in partial fulfillment of
the requirements for the degree of

DOCTOR OF PHILOSOPHY

WASHINGTON STATE UNIVERSITY
School of Electrical Engineering and Computer Science

MAY 2006

To the Faculty of Washington State University:

The members of the Committee appointed to examine the dissertation of SURYANARAYANA B. TATAPUDI find it satisfactory and recommend that it be accepted.

_____
Chair

_____

_____

ACKNOWLEDGEMENT

I would like to thank my advisor Dr. José Delgado-Frias for his valuable support and guidance in mentoring me during the course of my education. It has been a wonderful learning experience working with him. This dissertation would not have been possible without his help. I would like to thank Dr. Jabulani Nyathi for his help in research and it has been a fun experience working with him. I also would like to thank Dr. Valeriu Beiu and Dr. Partha Pande for being on my committee and their guidance during my study at WSU. I wish to thank my parents and brother for providing constant encouragement and support. Finally, I would like to thank the School of Electrical Engineering and Computer Science for awarding me a graduate teaching assistantship position. Without this support, I wouldn't have had the opportunity to write this dissertation.

A HIGH PERFORMANCE LOW POWER MESOCHRONOUS PIPELINE ARCHITECTURE FOR COMPUTER SYSTEMS

Abstract

by Suryanarayana Bhimeshwara Tatapudi, Ph.D.
Washington State University
May 2006

Chair: José G. Delgado-Frias

In a conventional pipeline scheme each pipeline stage operates on only one data set at a time. The clock period in conventional pipeline scheme is proportional to the maximum pipeline stage delay. We propose a mesochronous pipeline scheme, where pipeline stages operate on multiple data sets simultaneously. In this scheme the amount of logic in a stage is more and number of stages is less compared to a conventional pipeline. The clock period in this scheme is proportional to the maximum pipeline stage delay difference, which means higher clock speeds are possible and number of pipeline stages is significantly less. In mesochronous pipeline scheme, clock distribution network is simple and load on it is less. A detailed analysis of the clock period constraints is provided to show the performance gain and *Speedup* of mesochronous pipelining over other pipelining schemes. In mesochronous pipeline scheme, overall current drawn is less, resulting in significant power savings and also less *IR* drop on power lines. Also, the variation in supply current (*di/dt*) drawn by clock network is significantly less in mesochronous scheme, thus power supply noise is less. An 8×8-bit multiplier using carry-save adder technique has been simulated in conventional and mesochronous pipeline approach using TSMC 180nm (drawn length 200nm). The mesochronous

pipelined multiplier is able to operate on a clock period of 350ps (2.86GHz). This is a *Speedup* of 1.7 over conventional pipeline scheme and requires fewer pipeline stages and pipeline registers. The over-all power dissipation in mesochronous pipeline multiplier is less than 50% of the power dissipation in conventional pipeline multiplier. In the conventional implementation, power dissipation in clock network and pipeline registers is close to 80% of total power dissipation, while in the mesochronous implementation logic is dissipating more power. Also, the variation in current drawn by clock network in mesochronous scheme is less, causing less power supply noise.

Table of Contents

Page

List of Tables

Page

List of Figures

Page

Chapter 1

Introduction

Pipelining is a technique used to design high performance computer systems. Pipelining partitions a single large combinational logic block into small logic blocks called pipeline stages, separated by pipeline registers (latches, flip-flops). Fig. 1.1 shows a pipelined system with *N* stages. Pipelining is used to exploit the parallelism among various operations. The result is a reduction in average execution time and a significant speedup in a system's operation.

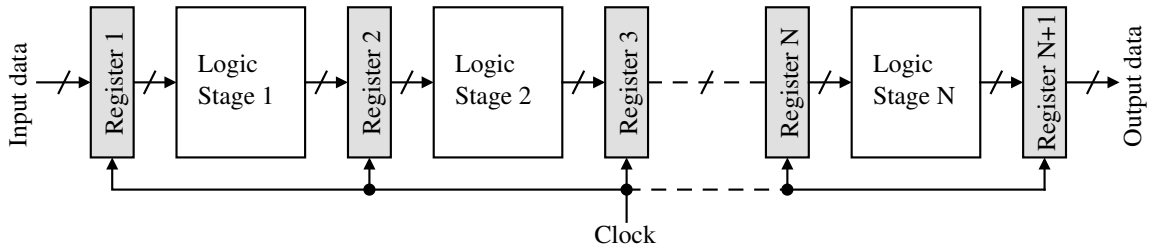


Fig. 1.1. *N* stage pipelined system.

1.1. Conventional pipeline scheme

In a Conventional Pipeline (CPP) system, pipeline stages operate on different data sets simultaneously and each stage on only one data set at any given time. Pipeline registers synchronize data movement from one stage to next with reference clock edge (typically the leading edge). New data is admitted into a stage only after data in that stage has been

cleared and latched by the register following it. In a pipelined system, pipeline stage with the longest computation time dictates clock-cycle time for the entire system. In designing a pipelined system the goal is to balance delays of all pipeline stages. However it is not always possible to perfectly balance the stages and there is always a critical stage with the longest computation time. Since all data synchronization in a pipelined system is based on clock signal, clock uncertainties (skew, jitter) must be controlled for proper functioning of the system. This is especially important as clock periods shrink further. Fig. 1.2 shows a graphical representation of a combined temporal and spatial variation for a generic pipeline stage *i*. Time and logic depth are represented in the horizontal and vertical axes, respectively. The shaded region in Fig. 1.2 is called computation cone and represents when computation is performed in this stage. The computational cones have been made linear to allow for a simpler analysis.

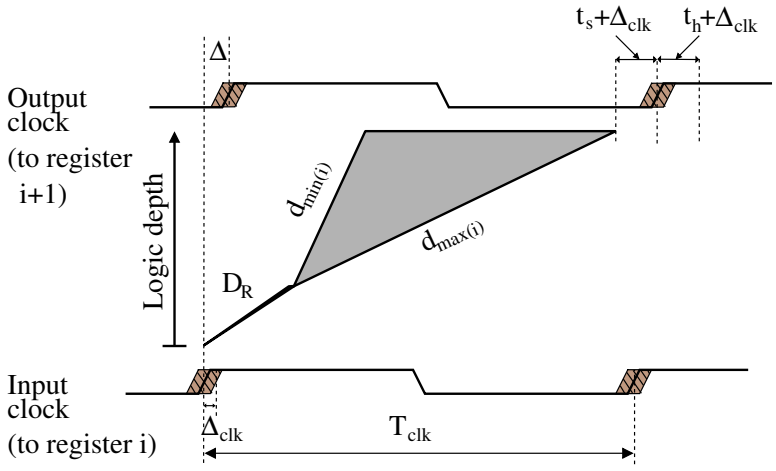


Fig. 1.2. Temporal/Spatial diagram of a pipeline stage *i*.

The variables used in Fig. 1.2 are defined as follows.

T_{clk}	Clock period
Δ	Constructive clock skew
Δ_{clk}	Unconstructive clock skew or clock uncertainties

D_R	Clock-to-output delay of the pipeline register
t_s, t_h	Pipeline register setup and hold times
$d_{min(i)}$	Minimum propagation delay through a stage *i* of a multi-stage system
$d_{max(i)}$	Maximum propagation delay through a stage *i* of a multi-stage system

Fig. 1.2 shows that delays in a pipeline are not only from pipeline stages (d_{min} and d_{max}) but also from pipeline registers (D_R, t_s and t_h). This is the overhead involved in pipelining a digital system. The delay of critical path includes D_R (clock-to-output delay of register), d_{max} (maximum stage propagation delay) and t_s (register setup time).

Temporal and spatial diagram of a three stage pipelined system is shown in Fig. 1.3. It is assumed that second stage in Fig. 1.3 is the critical stage in the system and has the maximum propagation delay (d_{max}).

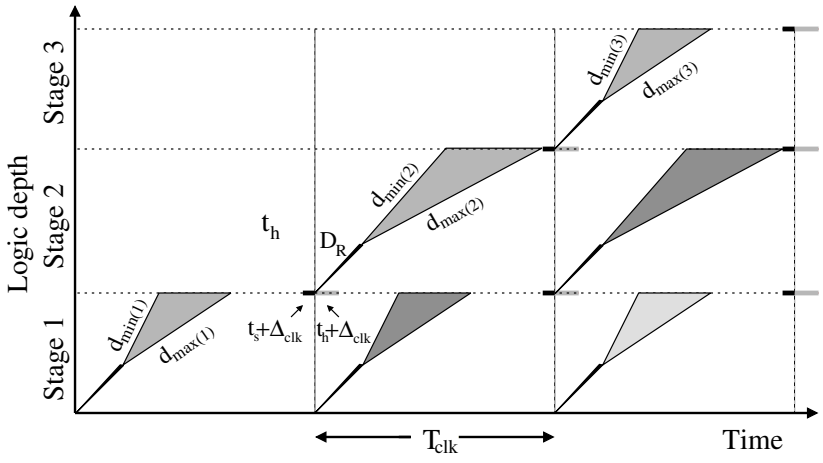


Fig. 1.3. Temporal/spatial diagram of a three stage CPP system.

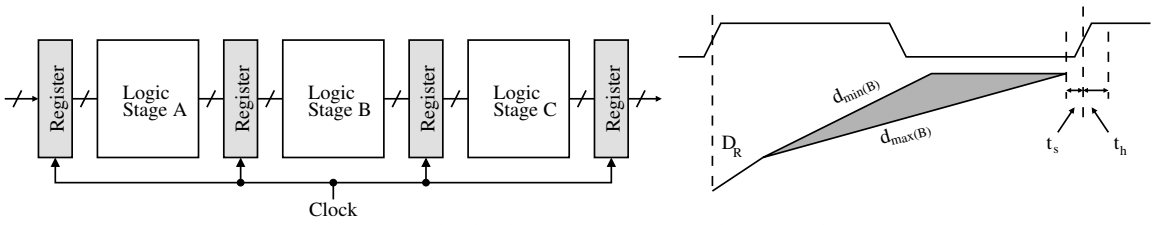
Equation (1.1) defines the clock period for a conventional pipeline system, where D_{max} is the largest of maximum propagation delay (d_{max}) values of all stages in the pipeline, $D_{max} = max(d_{max(i)})$. For example in Fig. 1.3, $D_{max}=d_{max(2)}$. The registers are also an overhead on the clock cycle time.

$$T_{clk\_cpp} \geq D_{\max} + D_R + t_s + \Delta_{clk} \tag{1.1}$$

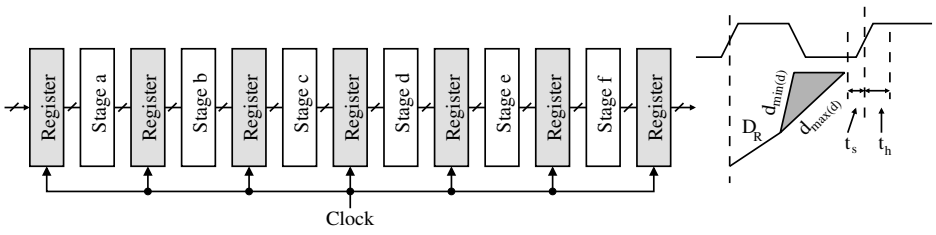
For (1.1) to be valid, the following condition must be satisfied. Here $D_{min}=min(d_{min(i)})$. The condition in (1.2) ensures that new data does not appear at input of a register before its hold time is up.

$$D_{\min} + D_R \geq t_h + \Delta_{clk} \tag{1.2}$$

From (1.1) it is clear that small clock periods are possible by decreasing delays: D_{max}, D_R, t_s and/or Δ_{clk}. Scaling can help decrease these delays and achieve smaller clock periods i.e. higher clock frequencies. However, in a given technology, to shrink the clock period further, the only delay which can be reduced is D_{max}. It is extremely difficult to further decrease register delays (D_R and t_s) and Δ_{clk} in the same technology. By partitioning each pipeline stage into more stages as shown in Fig. 1.4(b), stage delays can be reduced, in turn reducing D_{max} and T_{clk_cpp}. The result of such a partition is super-pipelines. In Fig. 1.4(a), it is assumed that stage *B* has the maximum propagation delay, while in Fig. 1.4(b) it is stage *d*.



(a) Pipelined system and computation cone of stage B



(b) Super–pipelined system and computation cone of stage d

Fig. 1.4. Temporal/spatial diagram of a three stage pipelined system.

From Fig. 1.4, it can be observed that the clock period can be reduced by means of super-pipelining [1]. However, this approach faces limitations imposed by the pipeline register delays (namely, D_R and t_s) and the maximum logic propagation delay (d_{max}). By partitioning the pipeline stages, stage propagation delay may become comparable to the register delays. As shown in Fig. 1.4(b) the register delays are a significant portion of the clock period. If this approach is used to reduce the clock period, the following issues arise: 1) each stage needs to be made ultra-thin to reduce d_{max}; 2) pipeline register becomes the dominant factor in the computation at each stage; 3) the number of pipeline registers is increased, in the example the number of register sets goes from four to seven; 4) clock distribution network becomes more complex with additional pipeline registers; 5) higher power requirements as the number of pipeline registers, clock frequency, and clock distribution network complexity increase; 6) tighter control on the clock skew will be required.

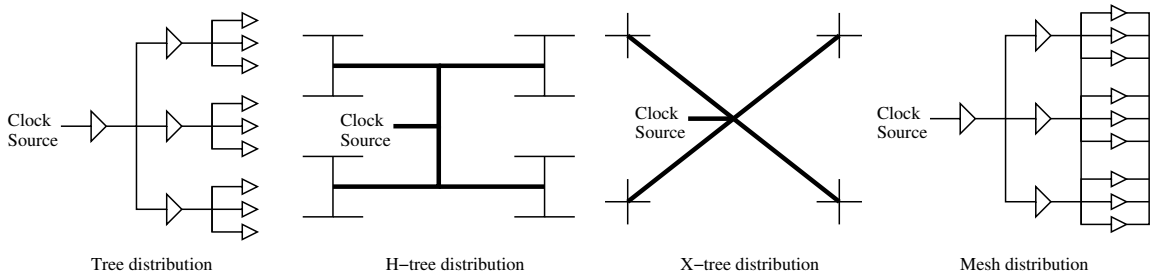


Fig. 1.5. Structures of common clock distribution networks.

The synchronization of data between various pipeline stages is very important for proper function of a CPP system. A globally distributed clock signal is used to synchronize all switching events and data movement in a CPP system. Today's high frequency clocks have to be generated on chip and distributed throughout the chip. In any digital system, of all data and control signals, clock signal is the one with the largest fan-

out, and fastest switching rate. The clock distribution network must be designed properly so that the clock signal triggers all pipeline register stages simultaneously and the critical timing requirements are satisfied. The clock signal must arrive at every registers in a CPP system and at precisely the same time. The most general approach to clock distribution is using buffered trees, H-trees, X-trees, and mesh network. The structures of these distribution schemes are illustrated in Fig. 1.5.

Due to variations in process parameters, shrinking feature sizes, and environmental variations, clock uncertainties like uncontrolled transmission line effects, clock skew and clock jitter [2], [3] are increasing. Large portion of clock period is being spent to counter these uncertainties. Thus the useful portion of clock period available for computation is decreasing. With shrinking feature sizes, interconnects are becoming thin, long, and their resistance is increasing. With high speed signals distributed on thin long wires, the inductive component of wire parasitic is gaining significance [4]. Also, by using ultra-thin super-pipelines shown in Fig. 1.4(b) to achieve higher operational (clock) frequencies, the load on clock network is increasing and it is becoming extremely difficult to distribute a clean giga-hertz frequency clock signal [5], [6]. With increase in size of clock network its power consumption also has increased to around 50% of the total chip power consumption [7].

1.2. Wave pipeline scheme

Wave pipelining (WPP) [8], [9] is one of the design methods that can be used in implementing computer systems. This pipeline scheme significantly reduces clock load, clock distribution area, power consumption and latency, compared to a CPP system.

In the WPP design method, pipelines are implemented without using intermediate pipeline registers. In this scheme no pipeline registers are used between logic stages. The entire system is treated as a single logic stage and new data sets are applied to the inputs of the logic stage before the outputs of previous data set are available. In this scheme, logic gates sever as virtual storage elements and multiple data sets (or waves) simultaneously propagate through different stages of logic without synchronization. This approach results in multiple data sets admitted during different clock periods being in the system at the same time and at various stages of computation. The wave pipeline approach results in maximum utilization of logic and eliminates the need for intermediate pipeline registers. The schematic of this scheme is shown in Fig. 1.6.

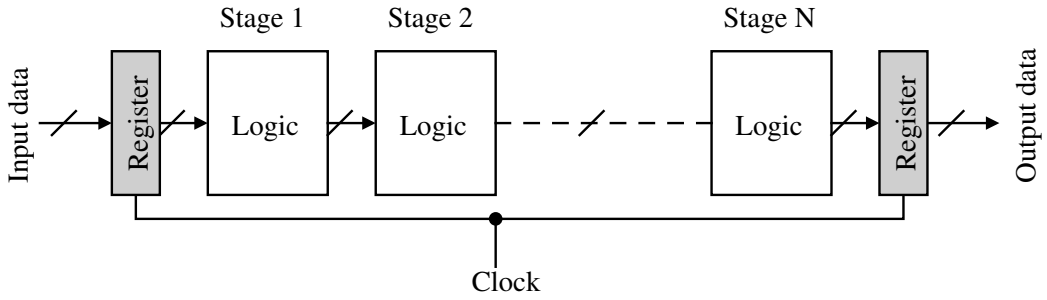


Fig. 1.6. Wave pipeline system.

The temporal and spatial diagram of WPP system is shown in Fig. 1.7. This diagram can be used to derive the equation for clock period for a WPP system. A detailed derivation of clock period is presented in [8], [9]. In Fig. 1.7, D_{MAX} and D_{MIN} are the maximum and minimum propagation delays of the entire system.

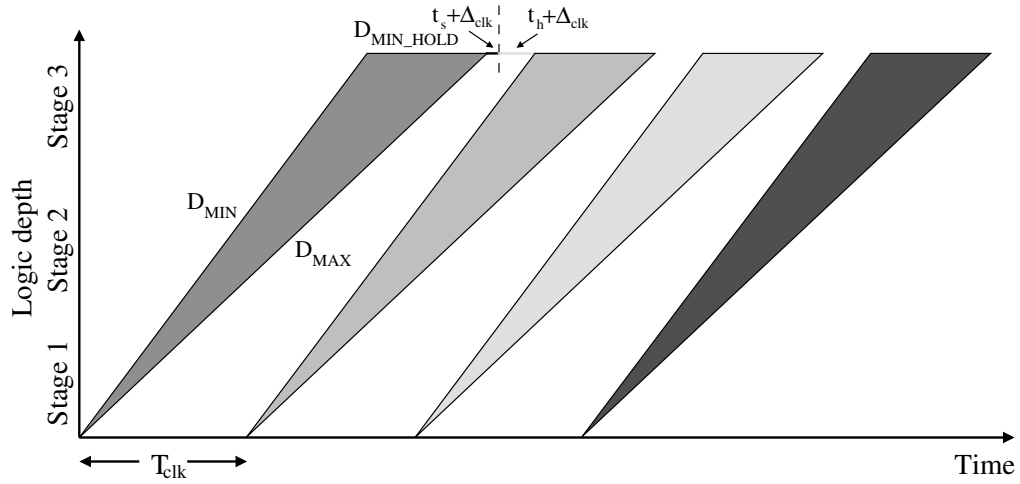


Fig. 1.7. Temporal/spatial diagram of a three stage WPP system.

Following the direction of arrows in Fig. 1.8, the equation for clock period in WPP can be written as follows

$$T_{clk\_wpp} + D_{MIN} - t_h - \Delta_{clk} - t_s - \Delta_{clk} - D_{MAX} \geq 0$$

$$T_{clk\_wpp} \geq (D_{MAX} - D_{MIN}) + t_s + t_h + 2\Delta_{clk} \qquad (1.3)$$

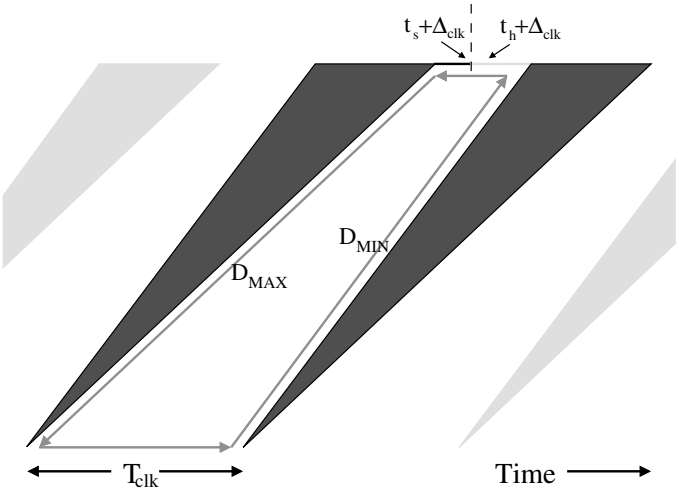


Fig. 1.8. Temporal/spatial diagram of a three stage WPP system.

The clock period in this pipeline scheme is determined by the difference between the maximum and minimum computation times of the entire system and safe time required

before a new data wave is admitted into the system. From (1.3) it is clear that a smaller delay difference would result in a higher clock frequency. The difference between D_{MAX} and D_{MIN} can be a large value since it takes into account all the intermediate stages. The delay difference can be minimized by delay balancing using buffers [8], [9].

In WPP scheme, the clock signal is distributed to the input and output registers only. The input register determines the rate at which data sets are admitted into the system, while the output register synchronizes the data sets at the end of computation. This is a simple clock distribution scheme.

However in WPP scheme, due to the absence of intermediate pipeline registers, it is extremely difficult to capture the state of intermediate nodes for test and debug purposes. Since the entire system is considered as a single wave pipelined stage, significant care must also be taken in designing the logic blocks and addition logic is required to keep the system delay difference small for maximum performance.

1.3. Micropipeline scheme

Micropipelines (μPP) is another pipelining technique that was introduced by I. Sutherland [10]. This scheme is an asynchronous pipeline scheme and it uses a two phase handshake signal for synchronization, instead of clock signal. The schematic of this scheme is shown in Fig. 1.9. A set of data at the inputs requests an operation *R(in)* and event on the *A(in)* line acknowledges the data word. Once data is acknowledged, this is passed to logic stage to perform the operation. At the end of the computation a request signal is generated through a delay circuit.

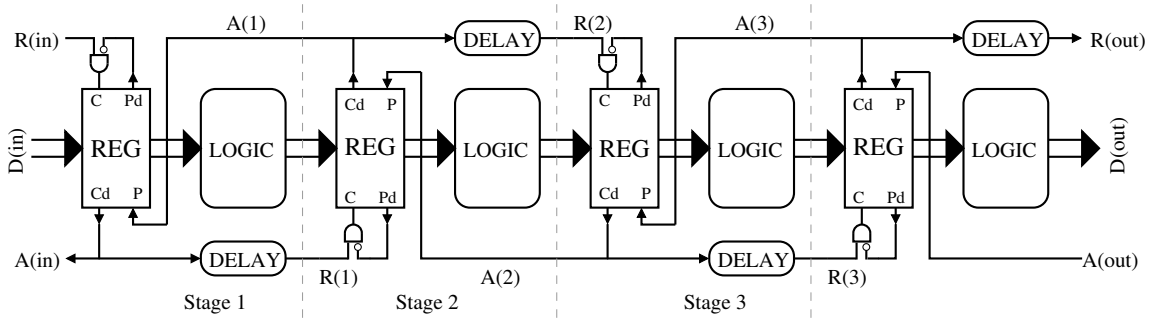


Fig. 1.9. Micropipeline system.

The temporal and spatial diagram of μPP system is shown in Fig. 1.10. It is assumed that second stage in Fig. 1.9 has the maximum propagation delay. Equation (1.4) defines the clock period for a micropipeline system, where D_{max} is the largest of maximum propagation delay (d_{max}) of all stages in the pipeline, $D_{max} = max(d_{max(i)})$. For example in Fig. 1.10, $D_{max}=d_{max(2)}$. The new term d_{Ack_max} is the time to produce and send back an acknowledge signal in the stage with the longest delay (given by D_{max}).

$$T_{clk\_upp} \geq D_{\max} + D_R + t_s + T_{Ack\_\max} \qquad (1.4)$$

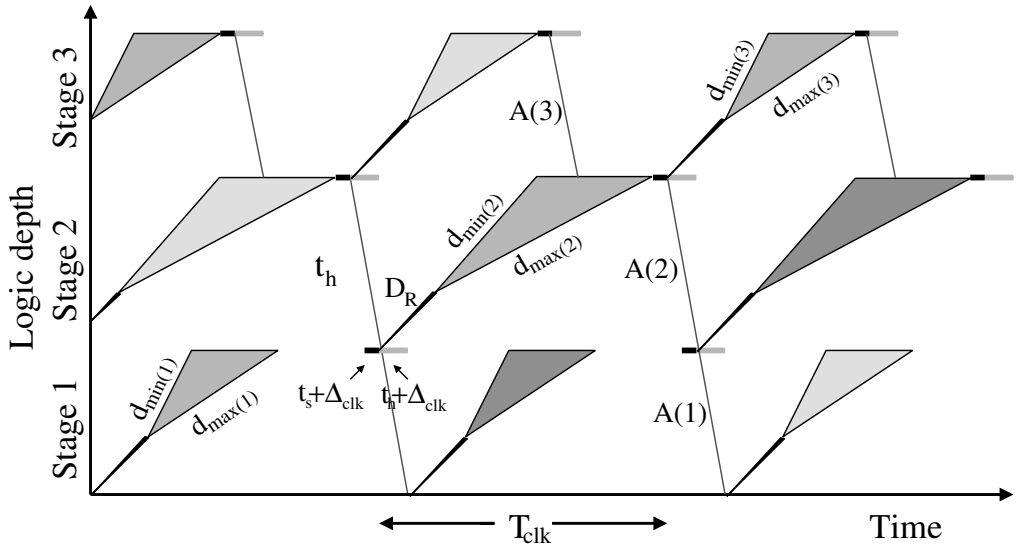


Fig. 1.10. Temporal/spatial diagram of a three stage μPP system.

The μPP scheme is an asynchronous pipeline scheme and does not require a globally distributed clock signal. The necessary data synchronization is achieved using a pair of request and acknowledge signals. These request and acknowledge signals perform handshaking between stages before data transmission. This means that always worst case path delays must be considered in designing the system. The handshaking protocol introduces addition delay and is an overhead on system performance. In a conventional pipeline implementation it is possible to design a system by considering the average path delays instead of worst case delays [11]. By careful design of a CPP system and its global clock distribution, better performance can be derived compared to the μPP scheme [11].

1.4. Need for novel pipeline architecture

In order to achieve significant performance gains compared to conventional pipeline implementation, pipeline architecture has to be modified to eliminate large pipelines and complex clock distribution mechanism. Architectures like wave pipelining [7], [8], micropipelines [10] and package wiring [2] have been proposed, but the performance gain is not significant. An asynchronous pipelining scheme like micropipelines may be appealing since it does not require a clock signal. However, it is complex compared to synchronous schemes and the performance improvement is higher in alternate synchronous schemes [2], [11]. In order to improve the performance of pipelined systems, and gain significant power savings we propose a novel pipeline scheme called mesochronous pipelining.

1.5. Summary

In this section we shall present a summary of important points from this chapter.

- *Conventional pipeline (CPP) scheme*: CPP scheme is often used in implementing high performance digital systems. Clock period in CPP scheme is proportional maximum propagation delay of the critical stage. $T_{clk\_cpp} \geq D_{\max} + D_R + t_s + \Delta_{clk}$. For proper functioning of a CPP system, a globally distributed clock signal is used, which has to be distributed throughout the system to trigger all pipeline registers simultaneously.
- *Super-pipelining*: Performance of a CPP can be increased by further partitioning the logic stages into smaller logic blocks. The result is large pipelines with large number of pipeline stages and pipeline registers. This complicates clock distribution and there is significant increase in power consumption.
- *Wave pipeline (WPP) scheme*: In WPP scheme, entire system is treated as a single logic block and system is clocked such that multiple data sets are simultaneously present in the system at various stages of computation. Clock period in WPP scheme is proportional to the difference between maximum and minimum propagation delay of the entire system. $T_{clk\_wpp} \geq (D_{MAX} - D_{MIN}) + t_s + t_h + 2\Delta_{clk}$. This scheme does not use any intermediate registers to synchronize data.
- *Micropipeline (μPP) scheme*: This is an asynchronous pipeline scheme, where a pair of handshake signals is used for data movement and synchronization between stages. So it does not require a globally distributed clock signal. Clock period of this system can be determined using $T_{clk\_upp} \geq D_{\max} + D_R + t_s + T_{Ack\_\max}$.

- *Need for novel architecture*: Novel architectures are required in future to design high performance low power digital systems. We propose the Mesochronous pipeline scheme.

1.6. Organization of this dissertation

The organization of this dissertation is as follows. In Chapter 2 we have a detailed discuss of the proposed mesochronous pipeline architecture and its clock distribution network. In Chapter 3 we compare performance of the proposed scheme with conventional pipeline scheme. In Chapter 4 we discuss some methods to tackle delay variations that could arise due to process and environmental variations. A Carry-Save Adder (CSA) multiplier has been implemented in conventional and mesochronous pipeline architectures, as a design example. A detailed discussion of their implementation and performance is presented in Chapter 5. In Chapter 6, we discuss the power consumption of the multiplier in conventional and mesochronous pipeline schemes. In Chapter 7, we discuss the implementation of a 4-bit mesochronous pipeline multiplier in AMI 0.5μm technology and the results obtained from the fabricated chip. In Chapter 8 some concluding remarks, contributions of this research and future research problems in mesochronous pipeline scheme are presented.

Chapter 2

Mesochronous Pipeline Scheme

The proposed Mesochronous Pipeline (MPP) scheme modifies conventional pipeline scheme to achieve higher performance and significant power savings. The term mesochronous has been used in the communications field; it has been defined as: the relationship between two signals such that their corresponding significant instances occur at the same rate. In this chapter the mesochronous pipeline architecture is discussed in detail and the design of the clock distribution network is also described.

2.1. Mesochronous pipeline scheme

In the Mesochronous pipeline (MPP) scheme a digital system is partitioned into pipeline stages like in the conventional pipeline (CPP) scheme. However it is clocked such that a pipeline stage is operating on more that one data set simultaneously. At any given time, multiple data sets can be present in a stage and these data sets are separated based on physical properties of internal nodes. This eliminates the need for some pipeline registers. This concept has some similarities to the wave pipeline scheme (WPP) [8], [9]. The number of registers that can be eliminated depends on how many simultaneous data sets can be sustained in a stage without synchronization. Effectively, MPP implementation of a digital system consists of more logic in pipeline stages and fewer pipeline stages compared to a CPP implementation. The schematic of this scheme is

shown in Fig. 2.1. Unlike the CPP scheme, clock signal in MPP scheme travels along with data and it is possible that different pipeline registers are triggered at different times. In the CPP approach it is absolutely necessary for all the pipeline registers to be triggered simultaneously. In MPP scheme, clock signal path includes delay elements (Δ_{Si}) which emulate the delay experienced by data in pipeline stages. In this pipelining scheme 1)higher clock frequencies are possible, 2) complexity of clock distribution is greatly reduced 3)influence of clock uncertainties is mitigated and 4) there are significant power savings. This architecture can be used in design of any high performance pipelined system.

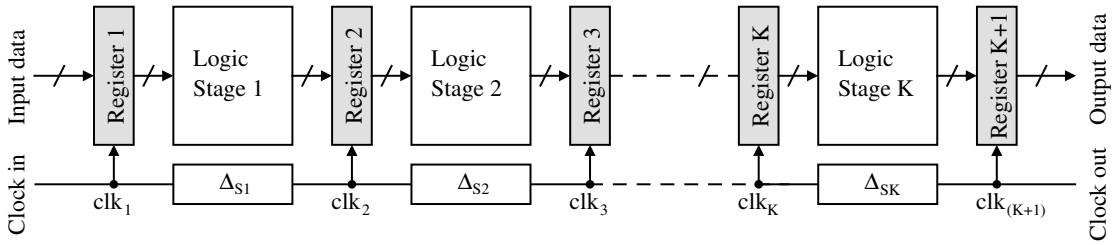


Fig. 2.1. Mesochronous pipeline scheme.

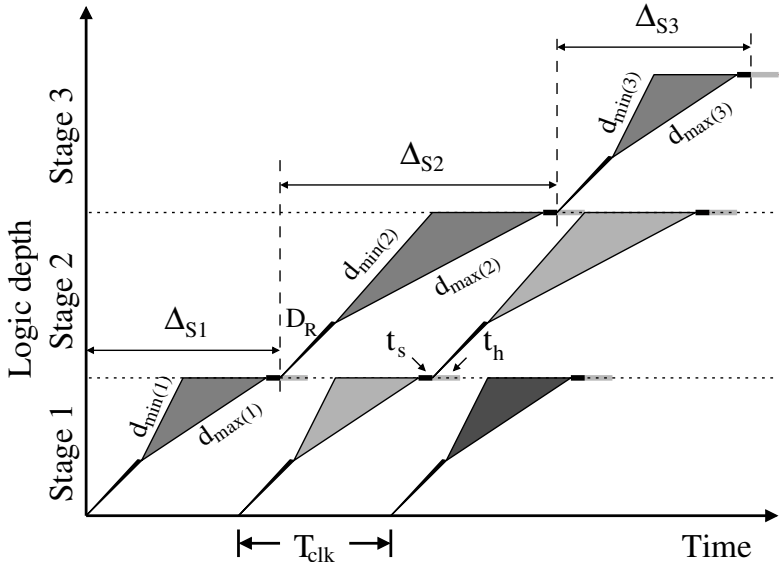


Fig. 2.2. Temporal/spatial diagram of proposed MPP system.

Temporal and spatial variation of the proposed MPP scheme is shown in Fig. 2.2 for a three stage system. In Fig. 2.2 it is assumed that stage 2 has the maximum delay difference. We shall refer to the difference between maximum and minimum propagation delays ($d_{max(i)} - d_{min(i)}$) of a stage i as the delay difference of that stage. The delay difference of any stage gives the amount of time the values generated at d_{min} have to be held, till the computation is complete in that stage.

The temporal and spatial diagram of a MPP system, shown in Fig. 2.3 can be used to derive the equation for clock period in this system.

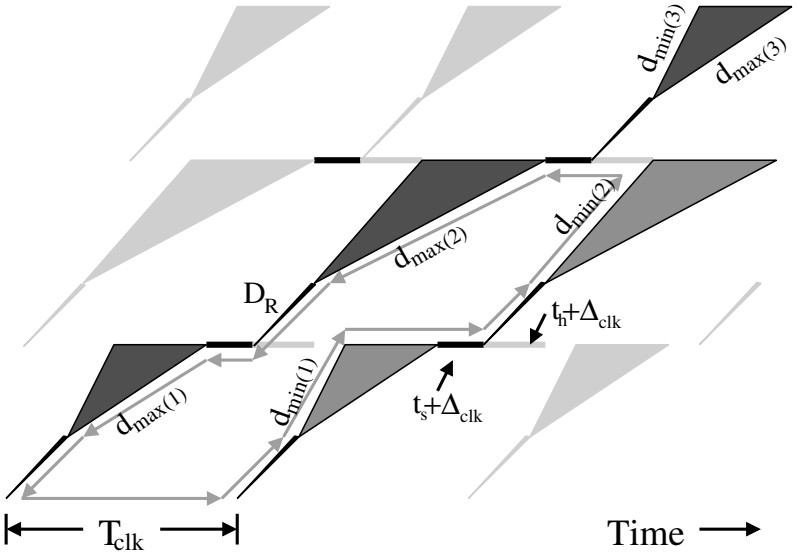


Fig. 2.3. Temporal/spatial diagram of a three stage MPP system.

Following the direction of arrows in Fig. 2.3, the equation for clock period in MPP can be written as follows. Here $d_{hold(i)}$ ($=d_{max(i)} - d_{min(i)}$) refers to the delay difference of a stage i.

$$T_{clk\_mpp} + D_R + d_{\min(1)} + d_{hold(1)} + D_R + d_{\min(2)} - d_{\max(2)} - D_R - d_{\max(1)} - D_R - t_s - \Delta_{clk} - t_h - \Delta_{clk} \geq 0$$

A general expression for deriving the clock period for MPP system can be written as

$$T_{clk\_mpp} \geq \sum_{i=0}^{j} d_{\max(i)} - \left(\sum_{i=0}^{j} d_{\min(i)} + \sum_{i=0}^{j-1} d_{hold(i)} \right) + t_s + t_h + 2\Delta_{clk} \qquad (2.1)$$

In (2.1), j is the stage with the maximum delay difference (in Fig. 2.2 stage j is stage 2). Equation (2.1) can be simplified as

$$T_{clk\_mpp} \geq \sum_{i=0}^{j} d_{\max(i)} - \left(\sum_{i=0}^{j} d_{\min(i)} + \sum_{i=0}^{j-1} \left(d_{\max(i)} - d_{\min(i)} \right) \right) + t_s + t_h + 2\Delta_{clk}$$

Eliminating the redundant terms in the above equation, we have the clock period equation for a MPP system as

$$T_{clk\_mpp} \geq d_{\max(j)} - d_{\min(j)} + t_s + t_h + 2\Delta_{clk} \qquad (2.2)$$

The clock period in MPP scheme is determined by the stage with the largest delay difference and safe time required before a new data set can be admitted into this stage. From (2.2) it is easy to see that for any stage i, $d_{max(i)} \geq T_{clk\_mpp}$ is always true. This means that new data is admitted into a stage before computation on previously admitted data set is complete. Depending on the $d_{max(i)}$ value of a stage and T_{clk_mpp}, at any given time two or more data sets can be present in a stage. From (2.2) it is clear that a smaller delay difference would result in a higher clock frequency. The delay difference can be minimized by delay balancing using buffers [8], [9].

2.2. Internal node constraints

Equation (2.2) indicates that the clock period is determined by the register setup and hold times when the input to output logic paths are equalized i.e. when $d_{max(j)} = d_{min(j)}$. It should be understood that factors like signal rise/fall time, capacitive loading, and circuit technology also influence the clock speeds. The limitations resulting from physical properties of internal nodes must also be considered to prevent any two adjacent data sets

from colliding. The fundamental circuit limitations determine the safe time to separate any two adjacent data sets. Consider the example shown in Fig. 2.4, the clock period is determined by the delay difference and register overhead, but the internal node variation is large causing adjacent data sets to collide.

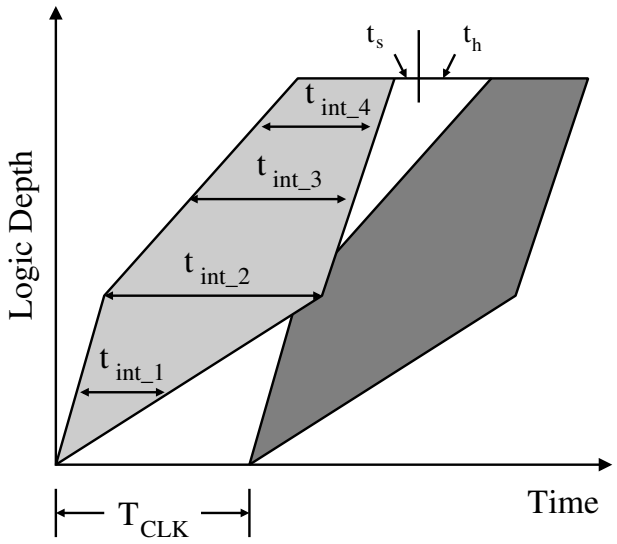


Fig. 2.4. Data sets collision.

A more general representation of minimum clock period of the MPP system is

$$T_{clk\_mpp} \geq \max(T_{\text{int}} \ , d_{\max(j)} - d_{\min(j)} + t_s + t_h + 2\Delta_{clk}) \qquad (2.3)$$

where T_{int} is the maximum value of all the internal node constraints

$$T_{\text{int}} = \max(t_{\text{int\_1}}, t_{\text{int\_2}}, t_{\text{int\_3}}, t_{\text{int\_4}}, \ldots\ldots\ldots\ldots) \qquad (2.4)$$

The internal node constraints can be eliminated by designing pipeline stages such that a stage's delay difference is greater than the delay difference at any internal node in that stage or in other words the delay difference should monotonically increase from input to output of a stage [8], [9] as shown in Fig. 2.5.

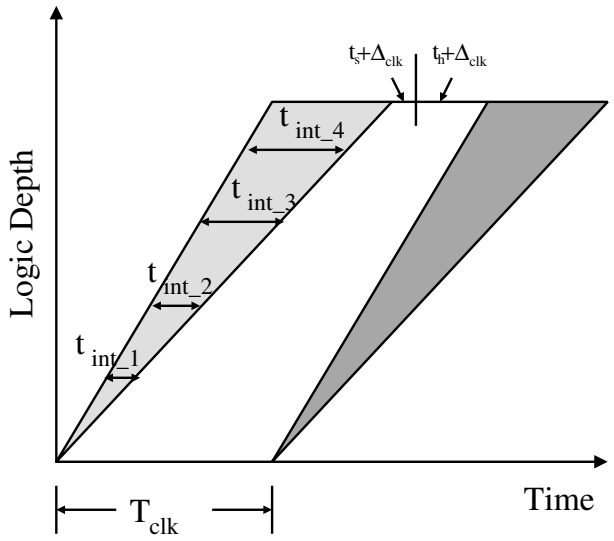


Fig. 2.5. Monotonically increasing delay difference.

Assuming that stages are designed to have monotonically increasing delay difference, we shall use (2.2) ($T_{clk\_mpp} \geq d_{\max(j)} - d_{\min(j)} + t_s + t_h + 2\Delta_{clk}$) to determine the clock period for rest of the discussion.

In MPP sceme, as the delay difference ($d_{max(j)} - d_{min(j)}$) approaches the timing requirements of the registers (setup time, hold time), the registers start to dictate the achievable performance gains. Until this point, focus was on the delay difference and its influence on the clock period, but the pipeline register could well be the dictating factor. Re-writing (2.2) as follows, the limit on delay difference of combinational logic is established.

$$d_{\max(j)} - d_{\min(j)} \leq T_{clk\_mpp} - \left(t_s + t_h + 2\Delta_{clk}\right) \qquad (2.5)$$

So the combinational logic between any two adjacent registers can be varied as long as the above condition is valid. This discussion emphasizes that it is important to design fast registers to derive improved performance. Unlike CPP scheme where a significant

portion of clock period is the register delay, MPP scheme is immune to this delay as computation takes place over multiple clock cycles.

In the MPP scheme, the clock signal travels with data (Fig. 2.1). Delays are included in the clock signal path so that clock experiences the delay similar to data sets in pipeline stages. In the next section we present some aspects of the clock path.

2.3. Designing the clock signal path delay elements

Consider the example of a stage shown in Fig. 2.6. The clock edge at ***A*** samples a data set from the previous stage. After traveling through the register and the stage *i*, the data set arrives at the next register before time ***E***. The next register must latch this data for the next stage (*i+1*) at time ***E***. The clock edge at ***A*** must be delayed for time period ***AE*** which can be represented as

$$T_{AE} = d_{\max(i)} + D_R + t_s + \Delta_{clk} \qquad (2.6)$$

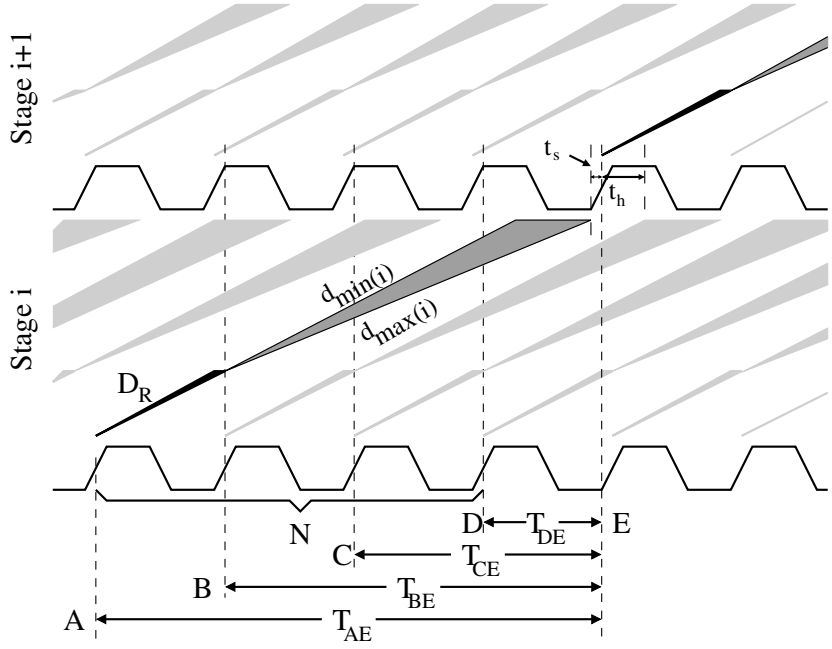


Fig. 2.6. Clock period and delay element.

By delaying the clock edge at ***A*** till ***E***, this clock edge triggers the register *i*, inputs the data set into stage *i*. Then it travels with the data set till time ***E***. By the time this clock edge arrives at ***E***, computation is complete on the data set. So the same clock edge triggers the register *i+1* to move the data into stage *i+1*. In this implementation, just as there are multiple data sets simultaneously present in a stage, there multiple clock edges present in the delay element Δ_{Si}.

The delay value shown in (2.6) must be present in the clock signal path to ensure that delays experienced by logic and clock satisfy the relation: clock delay ≥ logic delay. This value of delay required in clock signal path is large. Instead of using such a delay element (Δ_{Si} in Fig. 2.1) we can take advantage of the periodic nature of the clock signal. As shown in Fig. 5, the delay ***AE*** can be expressed as a smaller delay ($\delta_{(i)}$) plus an integer multiple ($N_{(i)}$) of clock period.

$$T_{AE} = \Delta_{Si} = d_{\max(i)} + D_R + t_s + \Delta_{clk} = N_{(i)} T_{clk\_mpp} + \delta_{(i)} \qquad (2.6)$$

From the example in Fig. 2.6, possible combinations of $N_{(i)}$ and $\delta_{(i)}$ are shown in Table 2.I. By choosing a higher value of $N_{(i)}$ in designing the clock signal path, the delay values can be reduced. This technique helps further reduce power consumption of clock network in the MPP scheme.

TABLE 2.I. COMBINATIONS OF $N_{(i)}$ AND $\delta_{(i)}$

$N_{(i)}$	$\delta_{(i)}$
3	DE
2	CE
1	BE
0	AE

2.4. Summary

The following is a summary of important points from this chapter.

- *Mesochronous pipeline (MPP) scheme*: In the MPP scheme, digital system is partitioned into large pipeline stages. The system is clocked such that multiple data sets, at different stages of computation are simultaneous present in every stage of the system. In the proposed scheme clock period is determine by the stage with the largest difference between its minimum (d_{min}) and maximum (d_{max}) propagation delays. Let Stage j be the stage with largest difference between its minimum and maximum propagation delays (delay difference), then clock period is defined by

$$T_{clk\_mpp} \geq d_{\max(j)} - d_{\min(j)} + t_s + t_h + 2\Delta_{clk}$$

- *Small number of pipeline registers and pipeline stages*: MPP scheme requires fewer pipeline stages and pipeline registers to obtain similar or better performance than a conventional pipeline scheme.
- *Clock distribution network*: Clock signal in MPP scheme travels along with data. So the clock path is parallel to the data path. Delay elements (Δ_{Si}) are included in the clock signal path so that clock signal experiences the same delay as the data set in the data path. This is a simpler clock distribution compared to the conventional pipeline scheme. Since there are fewer pipeline registers in MPP scheme, the load on clock network is also less.
- *Clock signal path delay elements*: The delay elements (Δ_{Si}) included in clock signal path can be simplified by taking advantage of periodic nature of clock signal. $\Delta_{Si} = N_{(i)} T_{clk\_mpp} + \delta_{(i)}$.

Chapter 3

Mesochronous Pipeline Performance Comparison

In this chapter we present a comparison of performance from the proposed pipeline architecture with the conventional and other pipeline architectures introduced in Chapter 1. This performance comparison is in terms of clock period. Mesochronous pipeline (MPP) scheme can operate with a smaller clock period, i.e. at a higher clock frequency compared to the conventional pipeline (CPP), wave pipeline (WPP), and micropipeline (μPP) schemes. The clock period for MPP scheme is proportional to maximum (stage) delay difference instead of maximum (stage) delay, as derived in Chapter 2. Since the CPP scheme is the most widely used pipeline architecture to implement computer systems, we shall compare the speedup of our proposed MPP scheme with it.

3.1. Comparison of clock cycle time

Clock period for conventional pipeline, wave pipeline, micropipeline and the proposed mesochronous pipeline schemes is shown in Table 3.I.

TABLE 3.I. COMPARISON OF CLOCK CYCLE TIME (T_{CLK})

Pipeline Scheme	**T_{clk}**
Conventional (CPP)	$D_{\max} + D_R + t_s + \Delta_{clk}$
Wave (WPP)	$(D_{MAX} - D_{MIN}) + t_s + t_h + 2\Delta_{clk}$
Micropipeline (μPP)	$D_{\max} + D_R + t_s + T_{Ack\_\max}$
Mesochronous (MPP)	$d_{\max(j)} - d_{\min(j)} + t_s + t_h + 2\Delta_{clk}$

In general, for a system implementation in any of the four pipelining schemes the following three inequalities hold

$$d_{max(j)} \leq D_{MAX}$$

$$d_{min(j)} \leq D_{MIN}$$

$$(d_{max(j)} - d_{min(j)}) \leq D_{max}$$

It is not difficult to show that for any system the following expression is valid.

$$(D_{\max} + D_R) \quad \geq \quad (D_{MAX} - D_{MIN} + t_h + \Delta_{clk}) \quad \geq \quad (d_{\max(j)} - d_{\min(j)} + t_h + \Delta_{clk})$$

This implies that

$$T_{clk\_mpp} \leq T_{clk\_wpp} \leq T_{clk\_cpp}.$$

This in turn validates our claim that MPP scheme delivers an improved performance compared to CPP, MPP and μPP schemes.

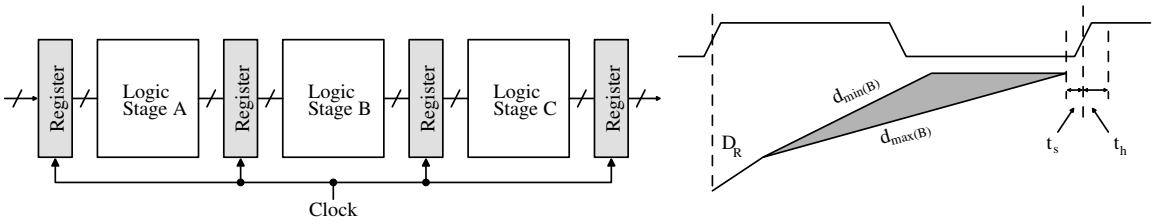
In WPP scheme, data propagation from one stage to the next is a function of delays through the stages and synchronization with the global clock occurs at the output register. This design approach leads to cumulative system delays, since the delays through the stages are added. The stage clocks are determined from data dependencies and delays. The global clock rate is higher in MPP scheme and this is shown in the equations derived above. Delay minimization per stage would allow for ease of testing in MPP scheme compared to WPP where the system delays are lumped; i.e. the minimum and maximum

delays considered are for the entire system instead of a stage by stage delay minimization.

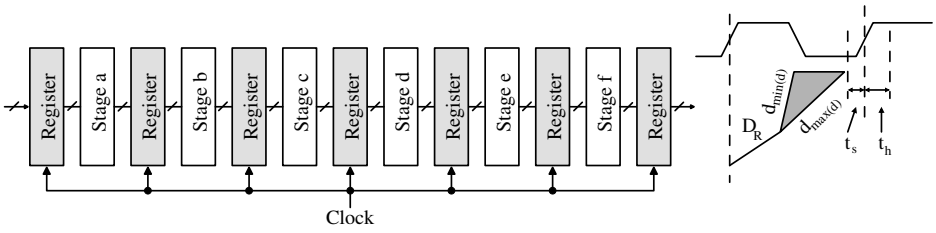
3.2. Conventional and Mesochronous pipeline performance comparison

To compare the performance gain from mesochronous pipeline scheme, we define a *Speedup* [1] metric as follows

$$Speedup = \frac{T_{clk\_cpp}}{T_{clk\_mpp}} = \frac{D_{\max} + D_R + t_s + \Delta_{clk}}{d_{\max(j)} - d_{\min(j)} + t_s + t_h + 2\Delta_{clk}} \qquad (3.1)$$



(a) Pipelined system and computation cone of stage B



(b) Super−pipelined system and computation cone of stage d

Fig. 3.1. Temporal/spatial diagram of a three stage CPP system.

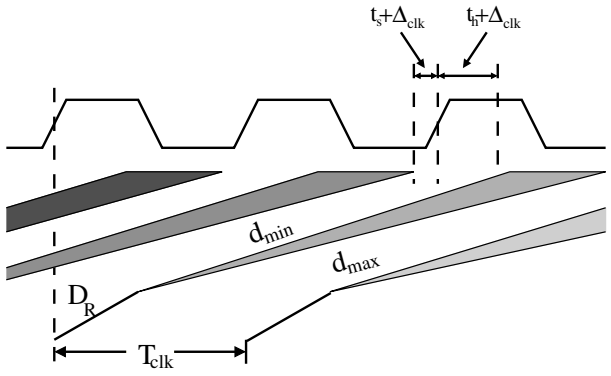


Fig. 3.2. Computation cones of critical stage in MPP system.

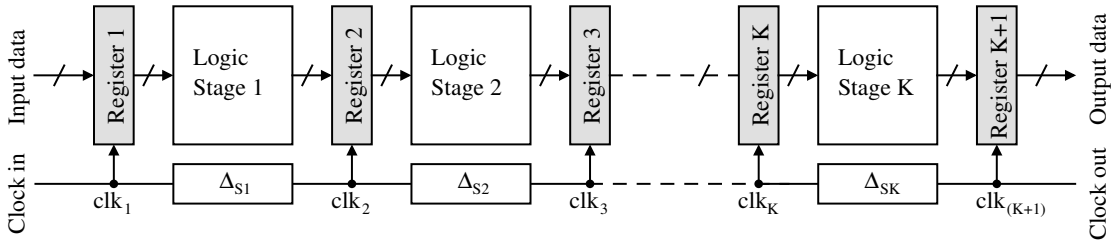


Fig. 3.3. Mesochronous pipeline scheme.

We study performance gain with the *Speedup* metric using Fig. 3.1 and Fig. 3.2 as reference. In Fig. 3.1(a) a three-stage CPP system and computation cone of the stage with maximum propagation delay (d_{max}) are shown. A similar MPP system is shown in Fig. 3.3 and the computation cones of the stage with maximum delay difference ($d_{max} - d_{min}$) are shown in Fig. 3.2. In Fig. 3.1(a) it is assumed that stage *B* has the maximum propagation delay and in Fig. 3.3 stage 2 has the maximum delay difference.

Comparing Fig. 3.1(a) and Fig. 3.2 it can be observed that D_{max} is far greater than $d_{max(j)} - d_{min(j)}$ and register delays (D_R, t_s and t_h), so the *speedup* in this case is

$$Speedup = \frac{D_{\max} + D_R + t_s + \Delta_{clk}}{d_{\max(j)} - d_{\min(j)} + t_s + t_h + 2\Delta_{clk}} \approx \frac{D_{\max}}{d_{\max(j)} - d_{\min(j)} + t_s + t_h + 2\Delta_{clk}} > 1 \qquad (3.2)$$

Equation (3.2) shows that better performance can be obtained by using MPP scheme and can be further improved by reducing the delay differences ($d_{maxIj)} - d_{min(j)}$).

Using the same technology, the performance of MPP scheme can be achieved in conventional scheme by partitioning the pipeline stages further as shown in Fig. 3.1(b). In Fig. 3.1(b) it is assumed that stage *d* has the maximum propagation delay. If D_{max} is approximately equal to $d_{max(j)} - d_{min(j)}$, *Speedup* is close to 1 (without loss of generality it can be assumed that $D_R+t_s+\Delta_{clk} \approx t_s+t_h+2\Delta_{clk}$).

$$Speedup = \frac{D_{\max} + D_R + t_s + \Delta_{clk}}{d_{\max(j)} - d_{\min(j)} + t_s + t_h + 2\Delta_{clk}} \approx 1 \qquad (3.3)$$

To achieve the same performance (i.e. achieve $D_{max} \approx d_{max(i)} - d_{min(j)}$), a large number of stages (in turn more registers) will be required in conventional pipeline implementation compared to MPP scheme.

It should be noted that using thin pipeline stages (i.e. reducing d_{max}) in conventional scheme, will make register delays the main delay component in each stage. On the other hand in MPP the objective is to decrease the delay difference.

The proposed MPP scheme has been shown to be superior to CPP scheme. Mesochronous pipeline scheme achieves better performance that conventional pipeline scheme, with a small number of pipeline registers.

3.3. Summary

The following is a summary of important points from this chapter.

- *Higher clock frequencies*: In the mesochronous pipeline scheme, clock period is determined by the critical stage delay difference. In conventional and micropipeline schemes, clock period is determined by the critical stage delay. In wave pipeline

scheme, system delay difference determines the system clock period. It is easy to show that critical stage delay difference in MPP scheme is less than the delays that determine the clock period in CPP, WPP and μPP schemes. So, MPP scheme has a smaller clock period i.e. it can operate at higher clock frequencies.

- *Speed-up*: MPP has a considerable *Speed-up* on CPP. Also, performance of a CPP system can be achieved in MPP system using fewer pipeline stages and pipeline registers.

Chapter 4

Tackling Clock and Delay Variations

In Chapter 2, the mesochronous pipeline (MPP) scheme's clock signal path design has been discussed. It was shown that periodic nature of clock signal can be used in designing the delay elements Δ_{Si}, in the clock path as shown in (4.1). This helps in reducing the size of delay elements and saves significant amount of area and power in clock distribution.

$$\Delta_{Si} = N_{(i)} T_{clk\_mpp} + \delta_{(i)} \qquad (4.1)$$

In this chapter, some of the implications of using small delay elements in clock signal path are discussed. The emphasis is on how the value of $N_{(j)}$ of the critical stage j affects the clock period of the MPP system.

Also, in this chapter, we shall discuss the issues resulting from variation in delay values and how they can be handled. Process and environmental variations can cause variations in the stage delay values. These variations could jeopardize the functioning of the system. It is possible to adjust the clock signal path delays and clock frequency to restore the system to a working condition.

4.1. Clock variation tolerance

For a value of $N_{(i)}$ greater than one, data set no longer travels with its clock edge in a given stage and the following inequalities must be satisfied to prevent two adjacent data sets from colliding.

$$\begin{aligned} d_{\min(i)} + D_R - t_h - \Delta_{clk} &\geq (N_{(i)} - 1)T_{clk\_mpp} + \delta_{(i)} \\ d_{\max(i)} + D_R + t_s + \Delta_{clk} &\leq N_{(i)}T_{clk\_mpp} + \delta_{(i)} \end{aligned} \qquad \forall\ N_{(i)} > 1 \qquad (4.2)$$

These conditions introduce a bound on the clock period. The minimum value clock period can take is

$$\max\left(\left(d_{\max(j)} - d_{\min(j)} + t_s + t_h + 2\Delta_{clk}\right) \ , \ \max\left(\frac{d_{\max(i)} + D_R + t_s + \Delta_{clk} - \delta_{(i)}}{N_{(i)}} \right) \right) \qquad (4.3)$$

The maximum value clock period can take is

$$\min\left(\frac{d_{\min(i)} + D_R - t_h - \Delta_{clk} - \delta_{(i)}}{N_{(i)} - 1} \right) \qquad (4.4)$$

Here *j* is the stage with the maximum delay difference and *i* is the set of all the stages.

When $\delta_{(i)} = d_{max(i)}$ ($N_{(i)} = 0$) or $d_{min(i)}$ ($N_{(i)} = 1$), the upper bound on clock period approaches infinity and the lower bound approaches the value given by (2.2) $T_{clk\_mpp} \geq d_{\max(j)} - d_{\min(j)} + t_s + t_h + 2\Delta_{clk}$.

This means that when a delay element is used to derive the entire delay on the clock signal path, clock edge travels with data set and the system can run at any clock period.

For a value of $N_{(i)}$ greater than one, as $N_{(i)}$ increases, value of $\delta_{(i)}$ decreases rapidly and the clock period bounds can be written as

$$\max\left(\frac{d_{\max(i)} + D_R + t_s + \Delta_{clk} - \delta_{(i)}}{N_{(i)}} \right) \leq T_{clk\_mpp} \leq \min\left(\frac{d_{\min(i)} + D_R - t_h - \Delta_{clk} - \delta_{(i)}}{N_{(i)} - 1} \right) \qquad (4.5)$$

So, the range of clock periods the system can operate decreases rapidly in this case. Due to these limitations, it is recommended to design using small $N_{(i)}$ values. It should be pointed out that if it is required to run the system at its maximum frequency, the limiting factor would be the register delays as shown in the multiplier example (Chapter 5). This in turn imposes a limitation on the number of data sets that can be computed within the stage to a few. Thus maximum $N_{(i)}$ tends to be small.

4.2. Tackling delay variation

The cases which could necessitate change in clock period are when $d_{min(j)}$ and/or $d_{max(j)}$ of the critical stage j change. This would cause the failure of setup and/or hold time requirements and ultimately system failure. Variation in the stage delays would change the bounds on clock period as given by (4.5). So the clock period must be adjusted so that it falls in the range. In this case the delay units in clock signal path must also be adjusted so that clock edge arrives at the register at the required time. This must be done for every stage. These only arise if the value of parameter $N_{(i)}$ is greater than one. For example consider this simple system with the temporal and spatial diagram of critical stage j shown in Fig. 4.1.

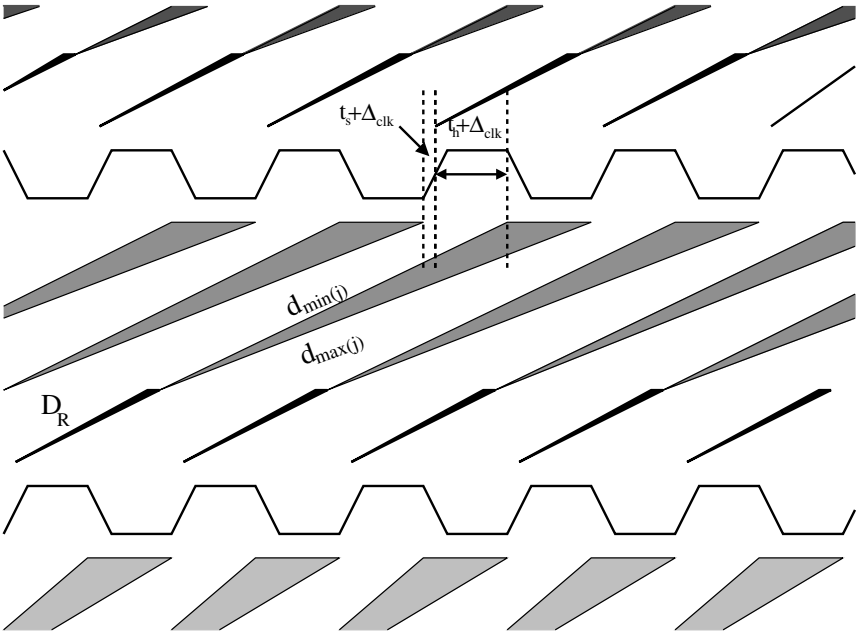


Fig. 4.1. Sample stage computation cones in a MPP system.

In Fig. 4.2(a), an example of variation in $d_{min(j)}$ value is shown, which causes the violation of hold time in stage *j*. Similarly an increase in $d_{max(j)}$ would violate the setup time requirement. In such cases the clock period must be increased as shown in Fig. 4.2(b). The increase shown in this example is more than the required amount and was chosen for clarity.

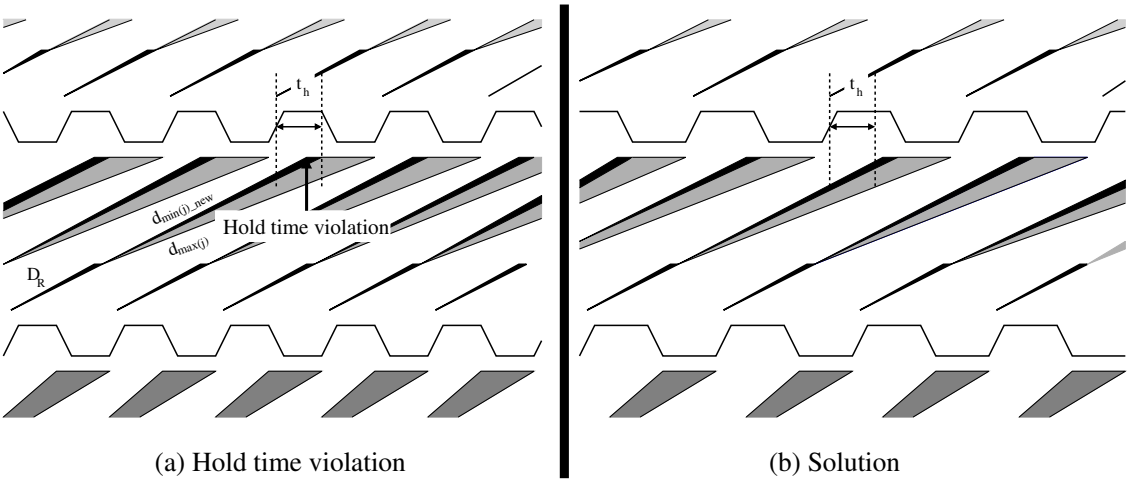


Fig. 4.2. Variation in d_{min} value.

We know that the following equations must be true for any stage *i*.

$$\begin{aligned} & d_{\max(i)\_original} + D_R + t_s + \Delta_{clk} = N_{(i)\_original} T_{clk\_mpp(original)} + \delta_{(i)\_original} \\ & d_{\max(i)\_new} + D_R + t_s + \Delta_{clk} = N_{(i)\_new} T_{clk\_mpp(new)} + \delta_{(i)\_new} \\ & T_{clk\_mpp(original)} \geq d_{\max(j)\_original} - d_{\min(j)\_original} + t_s + t_h + 2\Delta_{clk} \\ & T_{clk\_mpp(new)} \geq d_{\max(j)\_new} - d_{\min(j)\_new} + t_s + t_h + 2\Delta_{clk} \end{aligned} \tag{4.6}$$

The delay element must be adjusted according to (4.7) for proper functioning of the system.

$$\begin{aligned} \delta_{(i)\_new} = \delta_{(i)\_original} & + \left(N_{(i)\_original} T_{clk\_mpp(original)} - N_{(i)\_new} T_{clk\_mpp(new)} \right) \\ & + \left(d_{\max(i)\_new} - d_{\max(i)\_original} \right) \end{aligned} \tag{4.7}$$

Digitally variable delay elements can be used instead of static delay elements, in the clock signal path to tackle variations. Fig. 4.3 shows the schematic of a starved inverter used as a digitally variable delay element. In Fig. 4.3, the inputs C1, C2, C3 are used to program the delay element to provide different delay values.

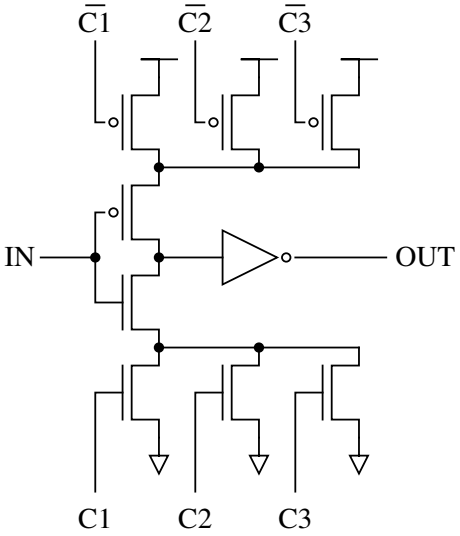


Fig. 4.3. Digitally variable delay element.

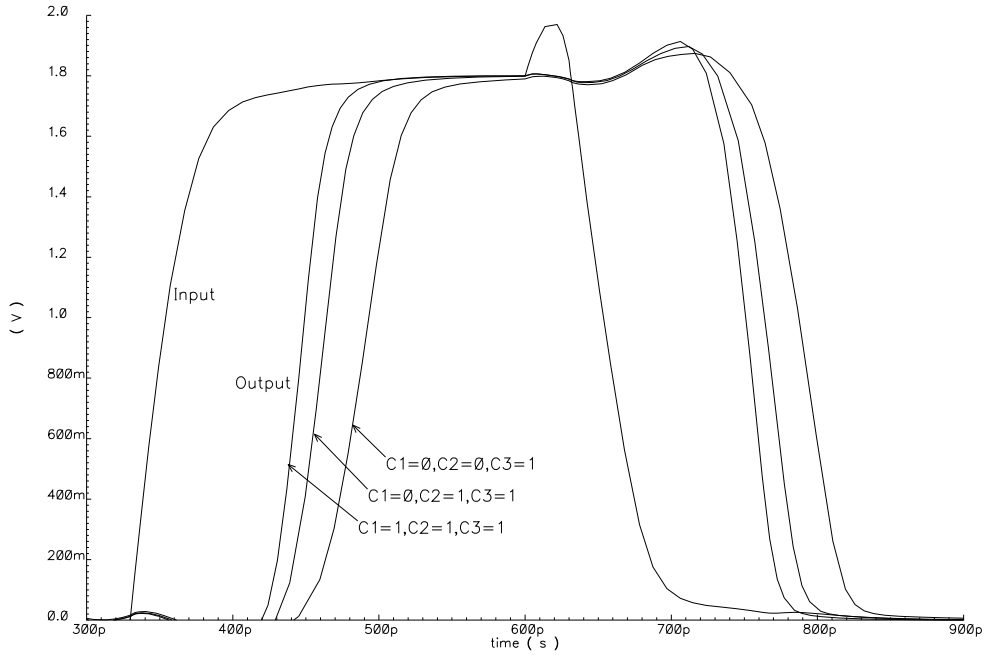


Fig. 4.4. Digitally variable delay element simulation.

A sample digitally variable element shown Fig. 4.3 has been simulated. The simulation results are shown in Fig. 4.4. In Fig. 4.4 it can be seen that by controlling the inputs C1, C2, and C3, delay value can be varied. The delay values for various combinations of C1, C2, and C3 are shown in Table 4.I. Sizing of transistors and using more control inputs, higher delay variation can be achieved from the delay element. Complex variable delay elements like: thyristor based delay elements [12], and programmable delay elements [13] can also be used to achieve higher delay variation.

TABLE 4.I . DELAY VARIATION IN DIGITALLY VARIABLE DELAY ELEMENT

Control inputs			**Delay(ps)**
C1	C2	C3	
0	0	1	139.94
0	1	1	110.81
1	1	1	96.03

4.3. Summary

The following is a summary of important points from this chapter.

- *Clock signal path*: In MPP scheme, clock signal travels parallel to data path. Clock path has delay elements (Δ_{Si}), so that clock travels with data. Clock signal path can be simplified by taking advantage of periodic nature of clock as $\Delta_{Si} = N_{(i)} T_{clk\_mpp} + \delta_{(i)}$. Using this concept, small delay elements can be used and this saves a significant amount of power in the clock network. When the entire delay in clock path is derived using physical elements ($N_{(j)}$=0 or 1), the system can operate on any clock period (lower limit given by $T_{clk\_mpp} \geq d_{\max(j)} - d_{\min(j)} + t_s + t_h + 2\Delta_{clk}$ and there is no upper bound). However, for values of $N_{(i)}$ greater than 1, upper and lower bounds appear on clock period value. Great the value of $N_{(j)}$, tighter the bound on clock period. In practical situations, due to other design limitations, value of $N_{(i)}$ tends to be small.
- *Delay variations*: Variations in pipeline delay values can cause system failure. The system can be restored to working condition by modifying the clock period and clock path delay values. Digitally variable delay elements can be used in clock path for this purpose.

Chapter 5

8×8-bit CSA Multiplier

In this chapter we present an 8×8-bit multiplier pipelined in the conventional pipeline (CPP) scheme and the novel mesochronous pipeline (MPP) scheme, to compare its performance. The multiplier architecture chosen is the Carry-Save Adder Multiplier. The Carry-Save Adder technique, the CPP and MPP implementations of the multiplier, simulations of the basic cells and the performance of the two multiplier implementations are discussed in detail here.

5.1. Carry-Save Adder multiplier

Carry-Save Adder (CSA) technique [14], [15] is a well known technique often used to realize fast multipliers. The general architecture of a multiplier using CSA technique is shown in Fig. 5.1. In this technique, an *M*-bit multiplier requires *M* layers of 1-bit Full Adders (FA) to reduce *M*-partial products to two partial products. Until this point data flow (sum and carry signals from FA) is from one layer of adders to the next. To generate the final product, the two *M*-bit partial products have to be merged in the last layer of the multiplier as shown in Fig. 5.1. A fast *M*-bit adder can be used for the final merging; however, propagation of the carry signal in this adder would make it the bottleneck stage. Fast adder implementations like carry-look-ahead or carry-select structure can be used to reduce delay in this layer; however these structures increase in complexity for large word

lengths and produce diminishing returns. Instead of this, we added *M*-layers of 1-bit Half Adders (HA) to merge the final two partial products. Effectively the multiplier implementation has 2*M* layers of adders. This improves throughput, however there is increase in latency. Increase in latency can be tolerated as the idea behind pipelining is to increase the throughput.

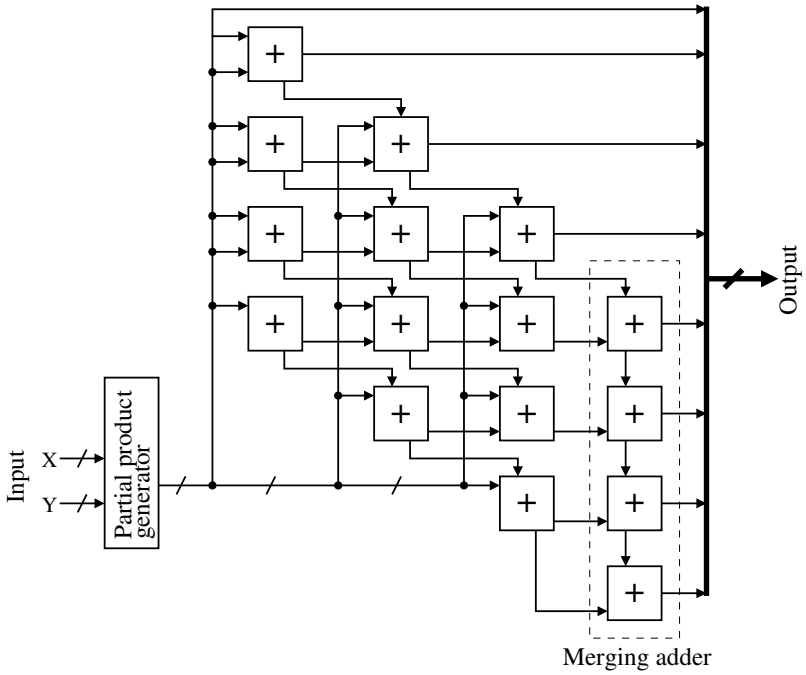


Fig. 5.1. Architecture of a multiplier using carry-save adder technique.

To achieve a fast multiplier, the CSA architecture must be pipelined. In CPP scheme according to (1.1)

$$T_{clk\_cpp} \geq D_{\max} + D_R + t_s + \Delta_{clk}$$

minimum clock period can be achieved by making each of the *2M* layers into stages of a pipeline, separated by pipeline registers. Effectively, an *M*-bit CPP multiplier would have *2M* stages with *2M+1* pipeline registers. An 8×8-bit pipelined multiplier implemented has 16 pipeline stages and 17 sets of inter-stage registers. The schematic of this multiplier

is shown in Fig. 5.2. To distribute the clock signal to all the pipeline register stages, a tree network has been used as shown in Fig. 5.2.

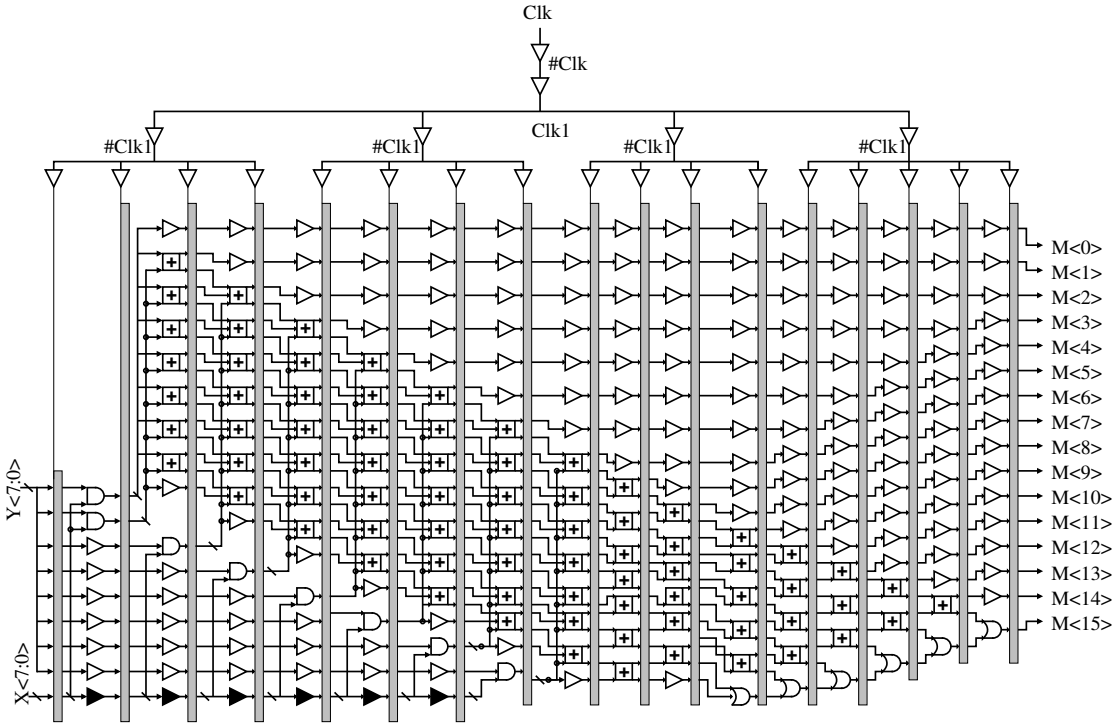


Fig. 5.2. 8×8-bit CSA multiplier implemented in CPP scheme.

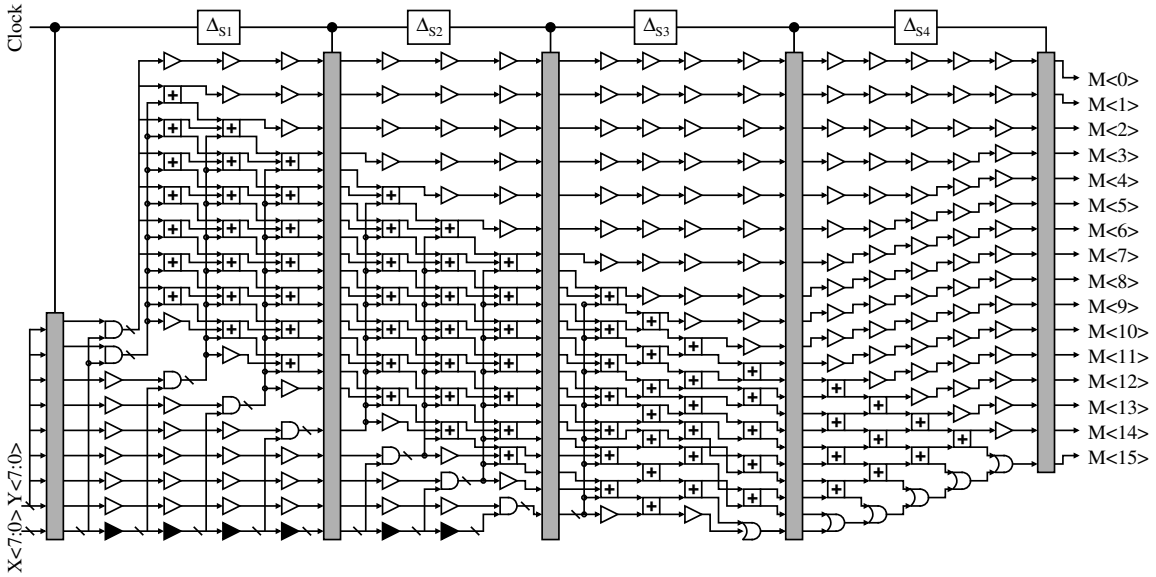


Fig. 5.3. 8×8-bit CSA multiplier implemented in MPP scheme.

Fig. 5.3 shows the schematic of the same 8×8-bit multiplier implemented in MPP scheme. Here the idea is to increase the amount of logic in a stage and clock the pipeline registers such that there are multiple data sets simultaneously present in a logic stage at different stages of processing. All of the logic enveloped between any two adjacent register stages supports multiple data sets simultaneously. Also, the number of register stages required to synchronize the data sets is small. In this implementation there are only 4 pipeline stages and 5 register stages. The placement of the registers is based on the maximum delay difference that can be handled for a target clock frequency. Unlike a tree distribution for clock signal in CPP scheme, the clock signal takes a linear path in MPP scheme as shown in Fig. 5.3. The clock travels close to the data path and includes delay elements realized using simple inverters.

A fast multiplier can be implemented if its basic cells have small propagation delays. The basic cells in the multiplier schematic shown in Fig. 5.2 and Fig. 5.3 are FA, HA, flip-flop, two input AND gate, two input OR gate, and buffers. The critical path in the multiplier includes FA and HA. In this implementation FA and HA have to generate the *Sum* (S) and *Carry* (Co) outputs simultaneously and the transmission-gate implementation of FA satisfies this requirement. To reduce propagation delay and avoid glitches, a differential implementation (complimentary inputs are used and complimentary outputs are generated simultaneously) is used. The FA with a carry-in of *logic 0* is used to realize HA. The transistor level implementation of the FA is shown in Fig. 5.4. The layout of this cell is shown in Fig. 5.11.

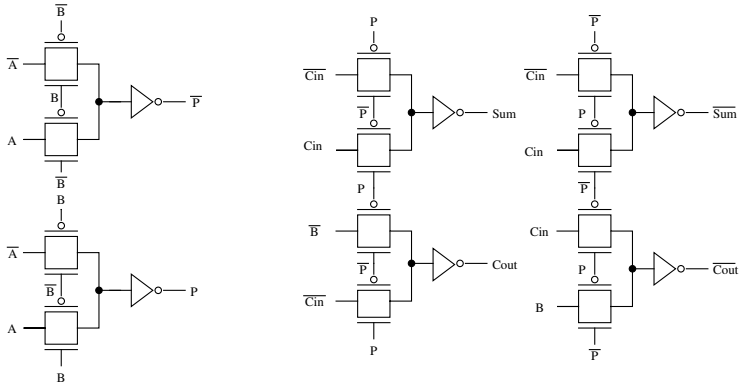


Fig. 5.4. Transistor level implementation of the full adder.

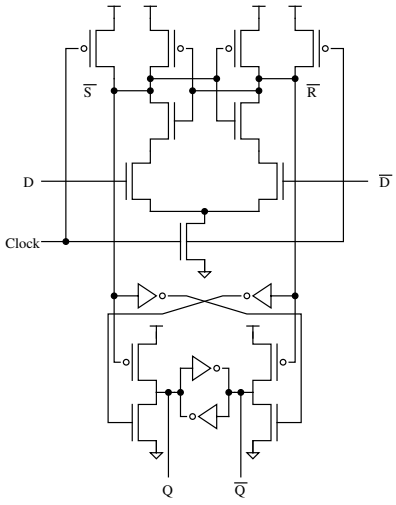


Fig. 5.5. Sense amplifier based flip-flop.

Since the FA and HA have been implemented in differential version, other basic cells are also differential implementations. The registers in the multiplier were realized using differential positive edge-triggered D flip-flop. A flip-flop samples its input at the clock rising edge, generates the output for the next stage. Since the sampling is done at the rising edge and all flip-flops in a register stage generate outputs simultaneously, the delay variations in the inputs to the register are eliminated when presented to the next stage i.e. the data is synchronized. An improved version of Sense Amplifier based Flip-Flop

(SAFF) with complementary push-pull [5], [16] is the flip-flop implemented in the register. The schematic of the SAFF is shown in Fig. 5.5 and layout is shown in Fig. 5.12

Since differential implementation has been chosen for FA, the SAFF is a good choice for this system due to its differential implementation. The SAFF accepts true and complimentary inputs and generates true and complimentary outputs simultaneously. It uses single-phase clock and is a small load on clock network. The first stage of the flip-flop is essentially a sense amplifier which assures accurate timing necessary in high speed applications [17]. This flip-flop also has short setup and hold times.

5.2. Basic cells simulation

Simulations have been performed on multiplier layout in TSMC 180nm (drawn length 200nm), 1.8V CMOS technology, using *SpectreS* under Cadence environment. The performance of the basic cells is presented in this section.

5.2.1. Full Adder

A number of simulations have been performed on the full adder to precisely characterize performance of this cell. Iterative process has been used to optimize the transistor sizes to achieve minimum propagation delay and delay variation. Co-incident inputs were applied to the full adder cell and propagation delay was measured. There are a total of 56 transitions possible for the 3 inputs to a full adder. Of these 56 transitions, only 32 transitions trigger a transition on the *Sum* (S) and/or *Carry* (Co) output. For these 32 transitions propagation delay of the full adder was measured. Propagation delay values obtained for these 32 transitions are graphically represented in Fig. 5.6. Using this plot,

minimum and maximum delays values and delay variation of FA can be calculated. These values are shown in Table 5.I.

TABLE 5.I . FULL ADDER DELAY VALUES

Maximum propagation delay (d_{max})	280ps
Minimum propagation delay (d_{min})	210ps
Delay variation ($d_{max} - d_{min}$)	70ps
Rate at which new inputs can be applied	175ps

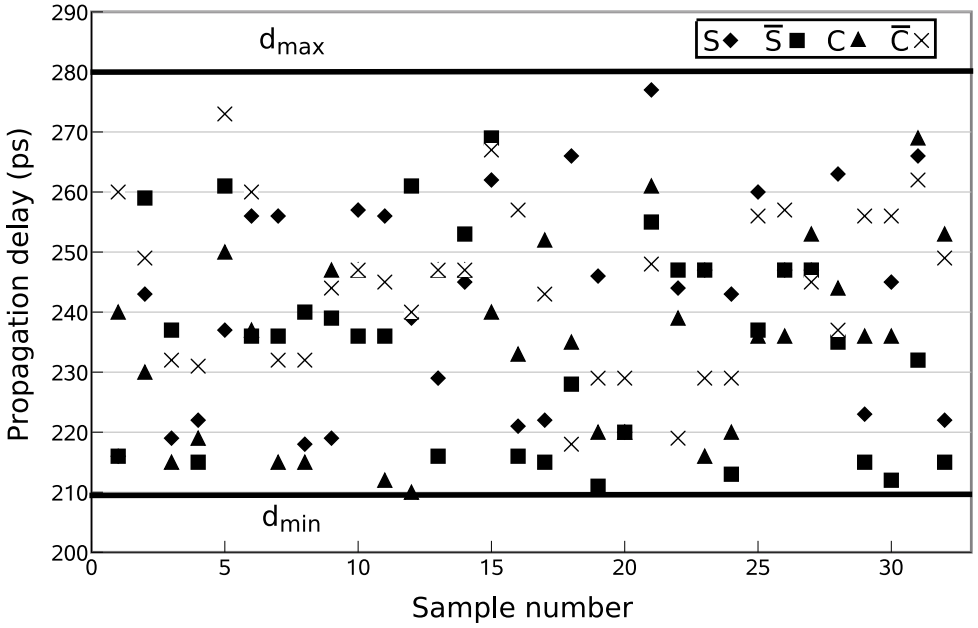


Fig. 5.6. Propagation delay of the full adder.

From Table 5.I we see that the propagation delay of the full adder varies from 210ps (d_{min}) to 280ps (d_{max}), resulting in a maximum delay variation of 70ps. Internal node constraints dictate the rate at which new inputs can be applied to the full adder and from simulations it was observed that the fastest rate at which inputs could be applied is once every 175ps.

In the multiplier schematic shown in Fig. 5.2 and Fig. 5.3, it can be observed that a layer of logic has FAs along with AND, OR gates and buffers. These AND, OR gates and buffers are designed to give a small propagation delay variation and since they are faster than FA, delay is added so that their propagation delay is close to that of the full adder. This would reduce the overall delay variation of a layer of logic.

5.2.2. Sense amplifier based flip-flop (SAFF)

The transistor sizes in SAFF [16] have been determined through an iterative process with knowledge of input signal driving strength and output drive needed. Simulations have been performed to determine the setup time (t_s), hold time (t_h) and the sampling time. Setup time is defined as the time for which data input must be stable before the arrival of active clock edge for the flip-flop to successfully store the data. Hold time is defined as the time for which the data must be held after the arrival of the active clock edge for the flip-flop to store the data. The setup time, hold time (t_h) and clock-to-output delay (D_R) are shown in Table 5.II. Simulations performed on the flip-flop revealed that the clock high time must be at least 160ps. Assuming a 50% duty cycle minimum clock period required is 320ps.

TABLE 5.II. SAFF TIMING VALUES

Setup time (t_s)	10ps
Hold time (t_h)	130ps
Clock-to-Q delay (D_R)	295ps
Minimum clock period required	320ps

5.3. Mesochronous pipeline multiplier

Simulations performed on the flip-flop revealed that the bottleneck in the system is the register, which dictated the minimum clock period time. Though the FA can accept inputs every 175ps, the flip-flop requires at least 320ps between successive samples. So, instead of logic dictating the clock period in the multiplier, the clock period (determined by flip-flop) determines the amount of logic that can be enclosed between any two adjacent registers. This is given by (2.5)

$$d_{\max(j)} - d_{\min(j)} \leq T_{clk\_mpp} - \left(t_s + t_h + 2\Delta_{clk}\right).$$

Since the clock period has to be at least 320ps, compensating for possible clock uncertainties a clock period of 350ps (≈2.86GHz) (T_{clk_mpp}) was targeted. Using the flip-flop delays obtained from simulations and (2.5)

$$d_{\max(j)} - d_{\min(j)} \leq 350 - (10 + 130 + 20) = 190\,ps$$

we know that the logic enclosed between any two adjacent register stages must have a delay difference less than 190ps.

The placement of registers as shown in Fig. 5.3 is based on this calculated limit on delay difference. The logic enclosed between any two adjacent register stages can handle multiple data sets simultaneously and has a delay difference less than 190ps.

Simulations performed on the entire system revealed that the system can successfully perform 8×8-bit multiplications every clock period i.e. 350ps [18], [19]. Some of the simulation waveforms are shown in Fig. 5.7 to illustrate the delay variation concept. The waveforms shown in Fig. 5.7 are of the first stage of multiplier.

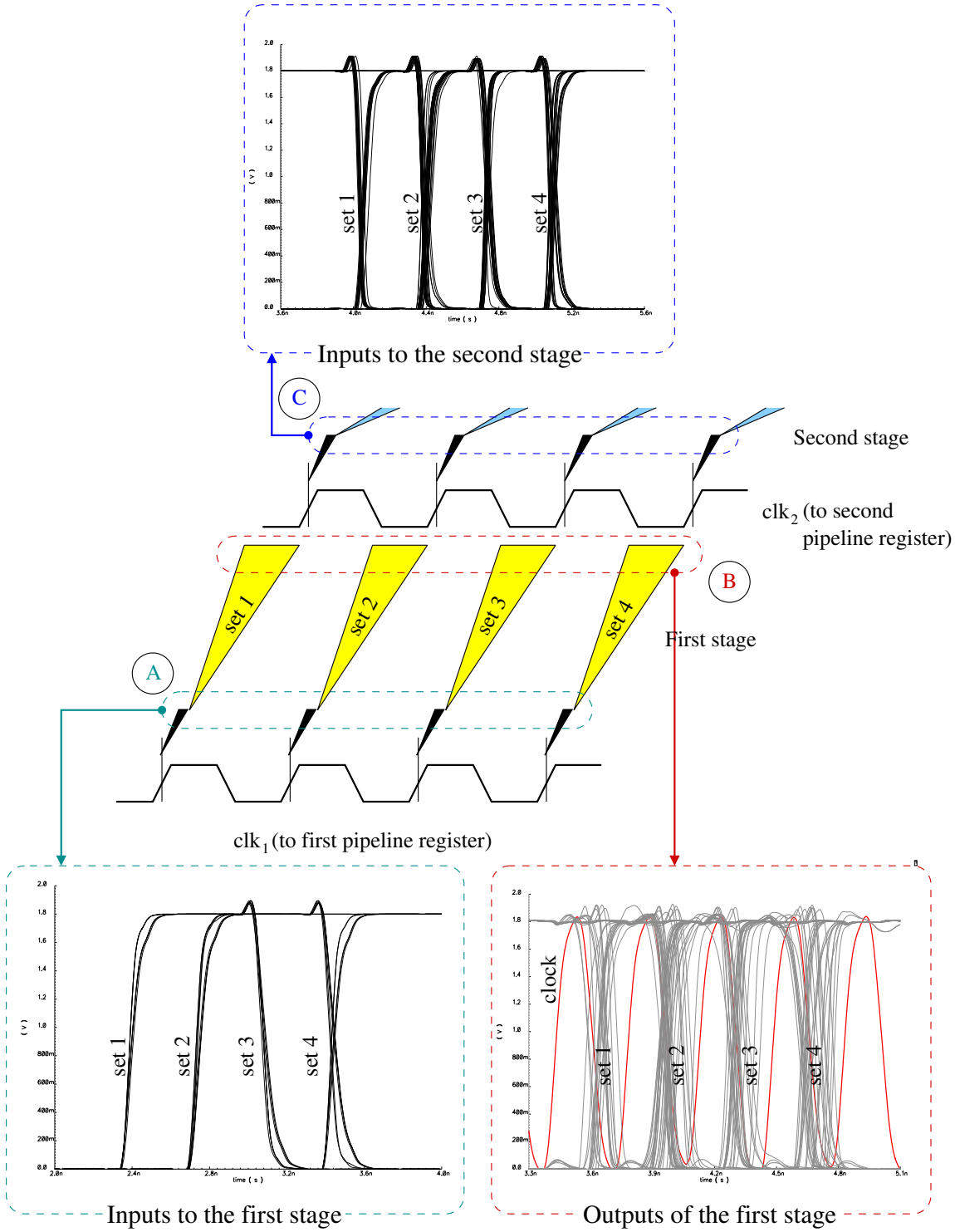


Fig. 5.7. Simulation waveforms.

There are four data sets simultaneously present in the first stage. In Fig. 5.7 at label (A) are the input data sets to the first stage of the multiplier. Each data set passes through the logic blocks shown in Fig. 5.3, and as the data set propagates, each data path adds different delay. As a result the delay variation of the data sets increases. In Fig. 5.7 at label (B) are the data sets with delay variations at the end of first stage (inputs to second register stage). Since the delay variation at this point is close to the calculated limit, a register stage is used to synchronize the data sets. The synchronized data sets as stored by the second register stage and presented to second stage at label (C) in Fig. 5.7. All the delay variations in the data sets from first stage are eliminated when presented to second stage. The small variation observed in the signals at label (C) is due to vertical clock skew and load variation of the register stage.

The MPP implementation of the multiplier is able to achieve a clock period of 350ps, with only 4 pipeline stages and 5 register stages. The layout of this multiplier is shown in Fig. 5.13. The load on the clock network is also small. The required delay in the clock signal path has been accomplished using inverters. Some important results of this multiplier implementation are summarized in Table 5.III

TABLE 5.III. MPP MULTIPLIER RESULTS

FA delay variation	70ps
SAFF setup time	10ps
SAFF hold time	135ps
SAFF Clk-Q delay	295ps
MPP multiplier pipeline stages	4
MPP multiplier pipeline registers	5
MPP multiplier clock frequency	2.86GHz

5.4. Conventional pipeline multiplier

Using the simulation results of the basic cells, performance of a super-pipeline implementation of the same multiplier can be accurately predicted. Best performance in CPP implementation would be possible if each layer of FA/HA is a pipeline stage. As stated previously, in such an implementation the number of pipeline stages would be 16 and number of register stages would be 17. The clock distribution in such an implementation is complex. According to (1.1)

$$T_{clk\_cpp} \geq D_{\max} + D_R + t_s + \Delta_{clk}$$

achievable clock period is only 595ps

$$T_{clk\_cpp} \geq D_{\max} + D_R + t_s + \Delta_{clk} = 280 + 295 + 10 + 10 = 595\,ps\,.$$

Using this clock period for CPP scheme, from (3.1)

$$Speedup = \frac{T_{clk\_cpp}}{T_{clk\_mpp}} = \frac{D_{\max} + D_R + t_s + \Delta_{clk}}{d_{\max(j)} - d_{\min(j)} + t_s + t_h + 2\Delta_{clk}}$$

we have a *Speedup* of 1.7 times, from the MPP scheme over CPP scheme.

In the calculated clock period value of CPP scheme, a significant portion of clock period is lost in the register delay. The amount of logic in a stage can be increased to mitigate the effects of the pipeline registers in super-pipelining. Let us consider *M* as the number of layers of FA considered as a single pipeline stage, $T_{clk\_cpp(min)}$ is minimum value of clock period achievable. As the logic depth in a stage increases the propagation delay of the logic influences the achievable clock period. $T_{clk\_cpp(min)}$ can be calculated as

$$T_{clk\_cpp(\min)} = \left(d_{\max\_FA} + D_R + t_s + \Delta_{clk}\right) + Md_{\min\_FA}$$

where d_{max_FA} and d_{min_FA} are the minimum delays of FA. Here we linearize the delay of additional layers of FA (for M >1) with d_{min_FA} instead of d_{max_FA}. This gives the least

possible delay and the smallest achievable clock period. The clock-period values for various values of M are shown in Table 5.IV. The results shown in Table 5.IV clearly indicate that the mesochronous pipeline scheme outperforms conventional pipeline scheme. In the multiplier, the MPP approach used fewer stages and gave higher frequency of operation, higher throughput and lower latency. A pipelining scheme similar to the proposed MPP scheme was used in the implementation of a network router [20].

TABLE 5.IV. CLOCK PERIOD OF CPP MULTIPLIER

M	No. of stages	Clock period
1	16	595ps
2	8	805ps
3	5	1015ps
4	4	1225ps

5.5. Mesochronous pipeline multiplier in ST Microelectronics 90nm technology

The 8×8-bit mesochronous multiplier has also been implemented in ST microelectronics 90nm technology, with supply 1.0V. The basic cells and multiplier have been simulated in the schematic tool. Some of the results obtained are discussed here.

A number of simulations have been performed on the full adder to precisely characterize performance of this cell. Propagation delay was measured for the 32 possible transitions that trigger a change in *Sum* (S) and/or *Carry* (Co) output. Propagation delay values obtained for these 32 transitions are graphically represented in Fig. 5.8. Using this plot, minimum and maximum delays values and delay variation of FA can be calculated. These values are shown in Table 5.V.

TABLE 5.V. FULL ADDER DELAY VALUES IN 90NM

Maximum propagation delay (d_{max})	100ps
Minimum propagation delay (d_{min})	62ps
Delay variation ($d_{max} - d_{min}$)	38ps

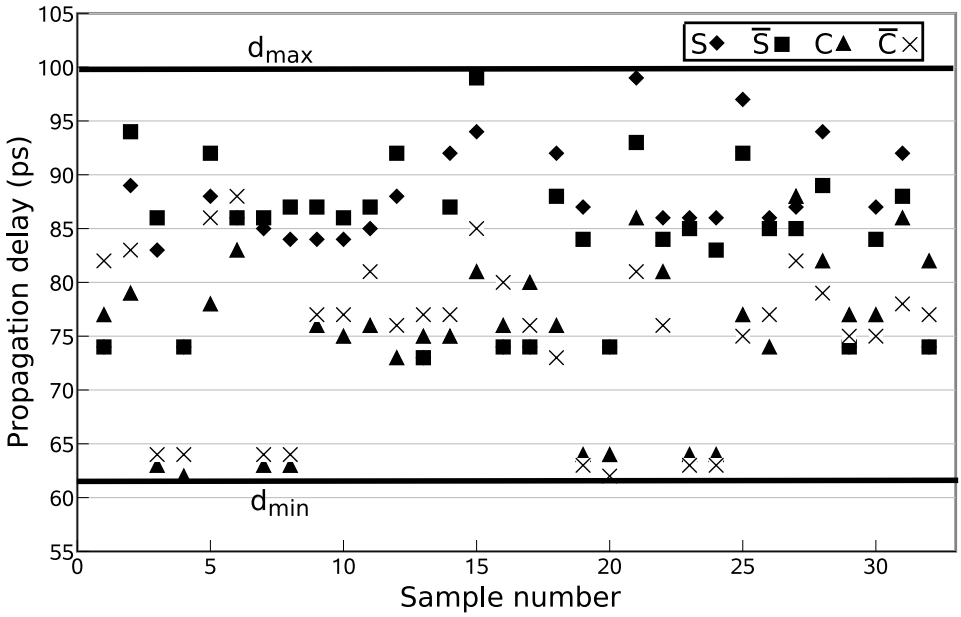


Fig. 5.8. Propagation delay of the full adder in 90nm technology.

From Table 5.V we see that the propagation delay of the full adder varies from 62ps (d_{min}) to 100ps (d_{max}), resulting in a maximum delay variation of 38ps.

Instead of the SAFF implementation used in TSMC 180nm implementation, a simpler dynamic two-phase D flip-flop [14], [15] has been used in this implementation. The schematic of this flip-flop is shown in Fig. 5.9. This cell is simple to implement and the minimum clock period requirement observed in SAFF implementation is less in the dynamic two-phase D-FF. Also, the flop-flop timing values like set-up time, hold time and clock-to-Q delay are less in the dynamic two-phase D-FF.

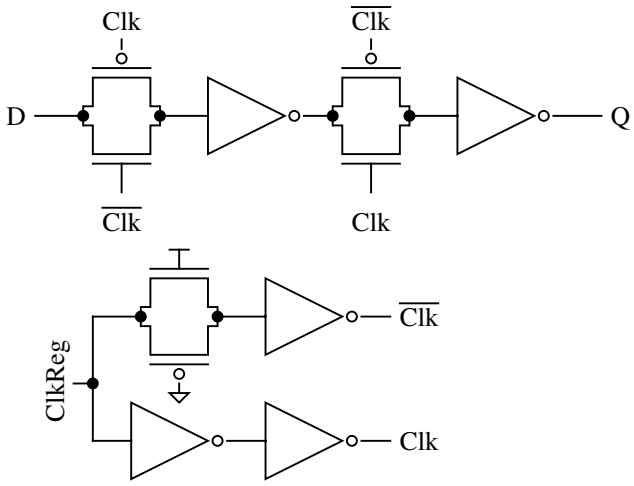


Fig. 5.9. D flip-flop and clk & $\overline{clk}$ circuit.

Simulations have been performed on this cell to obtain it's timing values. The simulation waveforms for various setup time values are shown in Fig. 5.10. From this waveform the setup time and clock-to-Q delay can be calculated.

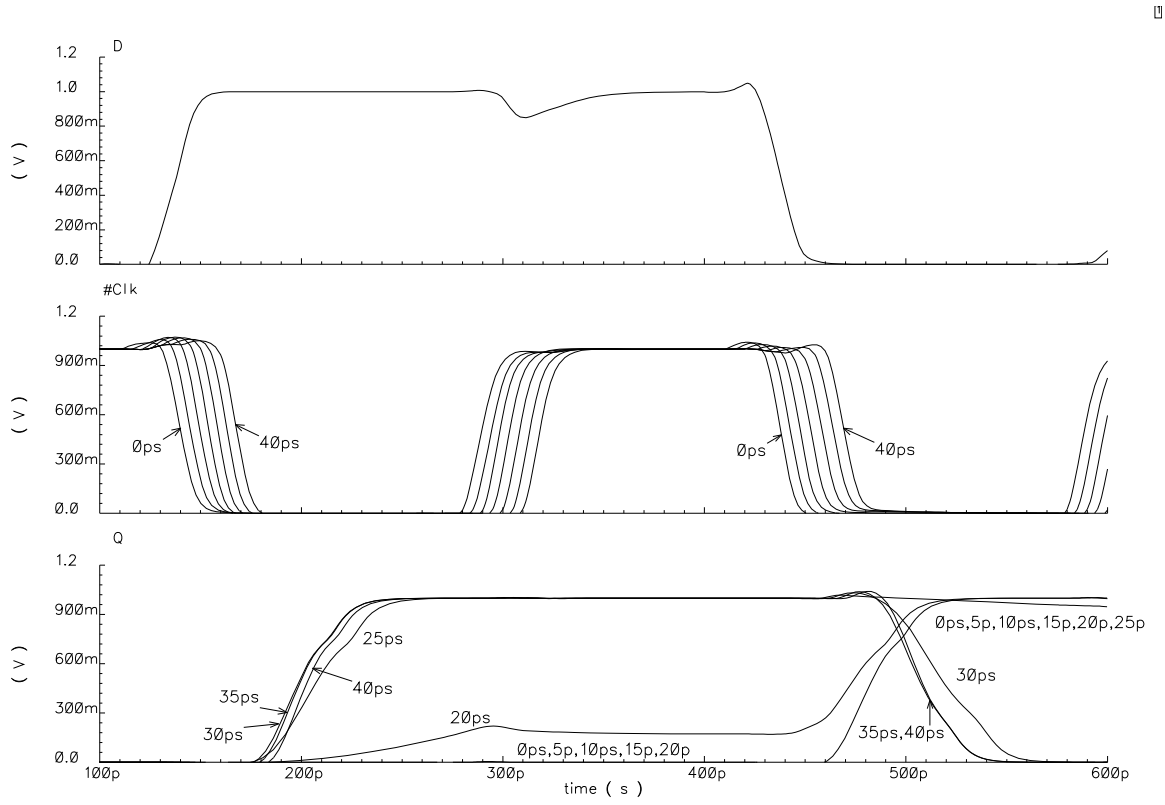


Fig. 5.10. Setup time of the dynamic two-phase D-FF.

The setup time, hold time (t_h) and clock-to-output delay (D_R) for this flip-flop obtained from simulations, are shown in Table 5.VI.

TABLE 5.VI. DYNAMIC TWO PHASE D-FF TIMING VALUES

Setup time (t_s)	35ps
Hold time (t_h)	5ps
Clock-to-Q delay (D_R)	37ps

The mesochronous multiplier implemented here is similar to Fig. 5.3. This implementation has 3 pipeline stages and 4 pipeline registers. Simulations performed on the entire system revealed that the system can operate with a clock frequency of 5GHz (clock period of 200ps). Some important results of this multiplier implementation are summarized in Table 5.VII.

TABLE 5.VII. MPP MULTIPLIER RESULTS IN 90NM

FA delay variation	38ps
SAFF setup time	35ps
SAFF hold time	5ps
SAFF Clk-Q delay	37ps
MPP multiplier pipeline stages	3
MPP multiplier pipeline registers	4
MPP multiplier clock frequency	5GHz

5.6. Summary

The following is a summary of important points from this chapter.

- *Mesochronous pipeline multiplier*: The Carry-Save Adder multiplier was pipelined using the mesoschronous pipeline scheme. To improve performance of basic cells of the multiplier i.e. full adder and half adder, fully differential transmission gate

implementations have been used. A full differential Sense Amplifier based Flip-Flop (SAFF) has been used in implementing pipeline registers. Due to the design limitations imposed by the SAFF, a maximum clock frequency of 2.86GHz could be used. Based on this limitation the multiplier was pipelined into 4 logic stages with 5 register stages. Each logic stage can handle 3 data sets simultaneously. Simulations performed in TSMC 180nm, 1.8V technology, on the MPP multiplier showed that it can operate at a maximum frequency of 2.86GHz (clock period of 350ps).

- *Conventional pipeline multiplier*: Based on the simulation results of the basic cells, the performance of a conventional pipeline implementation of the multiplier was calculated. The CPP multiplier can operate at a maximum clock frequency of 1.68GHz (clock period of 595ps). To achieve this performance, the multiplier should be split into 16 logic stages and 17 pipeline register stages.
- *MPP multiplier in 90nm technology*: The MPP multiplier schematic was simulated in ST microelectronics 90nm, 1.0V technology. The multiplier has 3 logic stages and 4 pipeline register stages and can operate on a clock frequency of 5GHz.

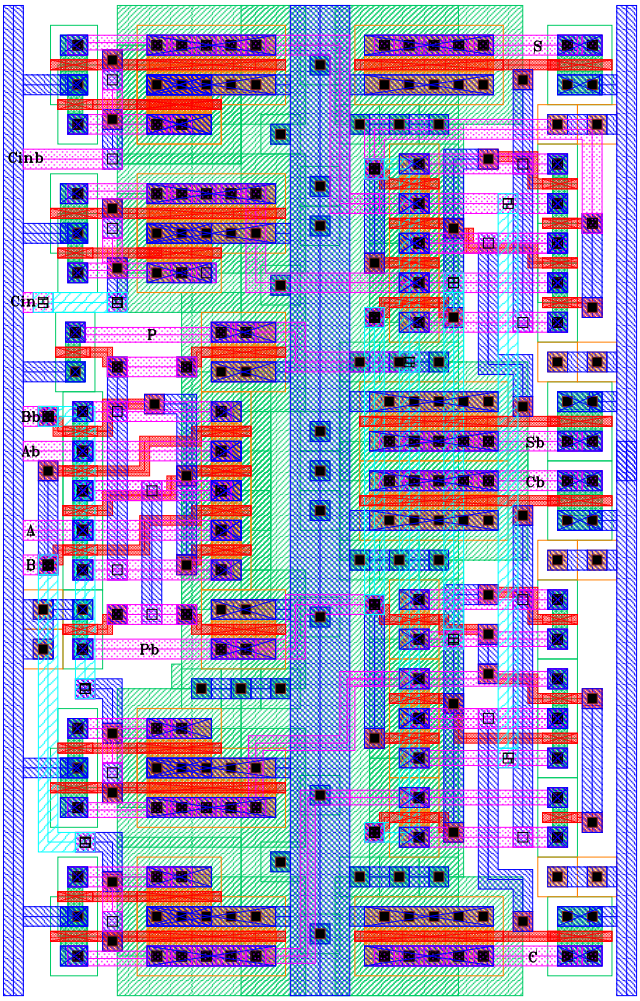


Fig. 5.11. Full Adder layout in TSMC 180nm technology.

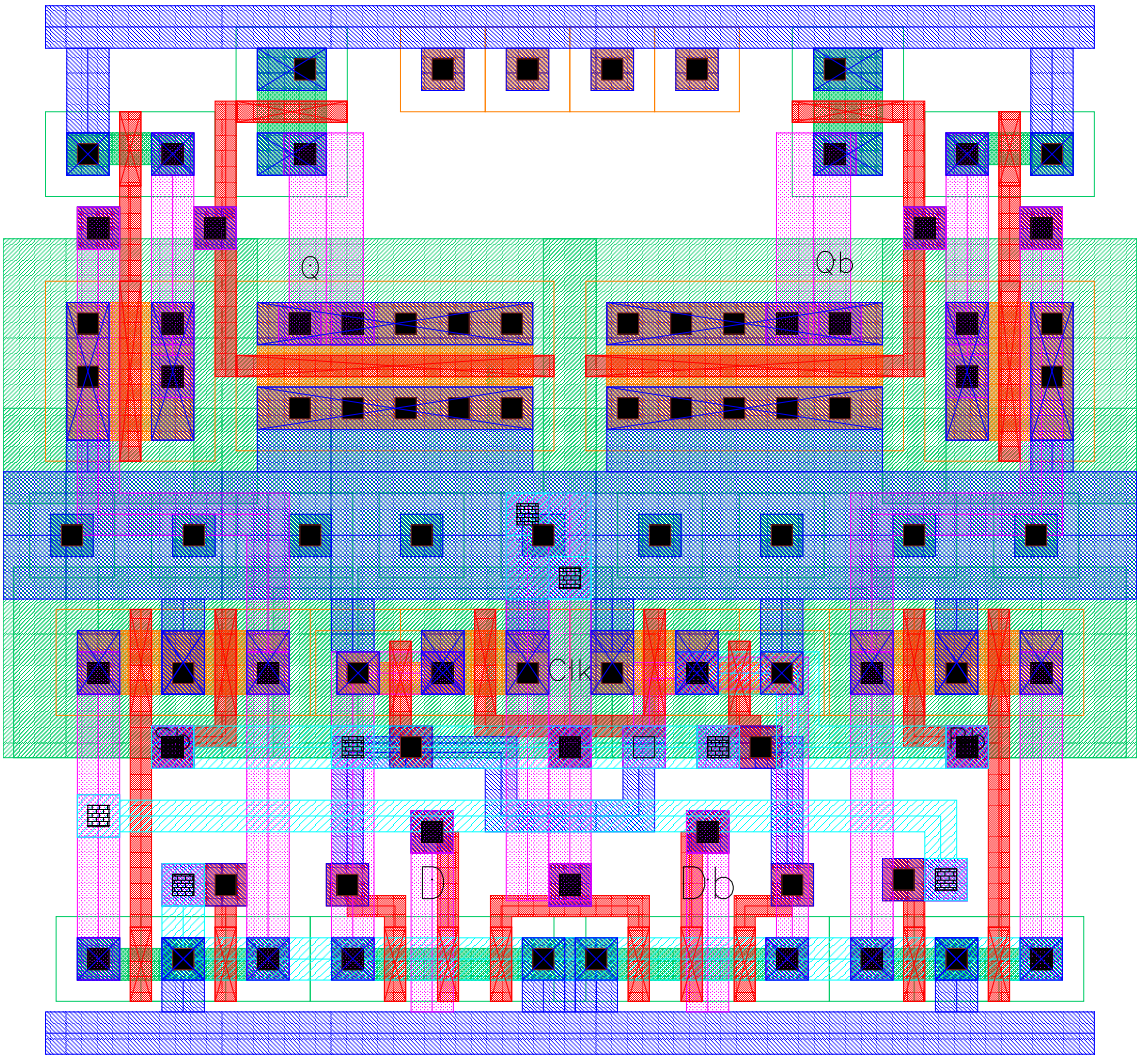


Fig. 5.12. Sense amplifier based flip-flop layout in TSMC 180nm technology.

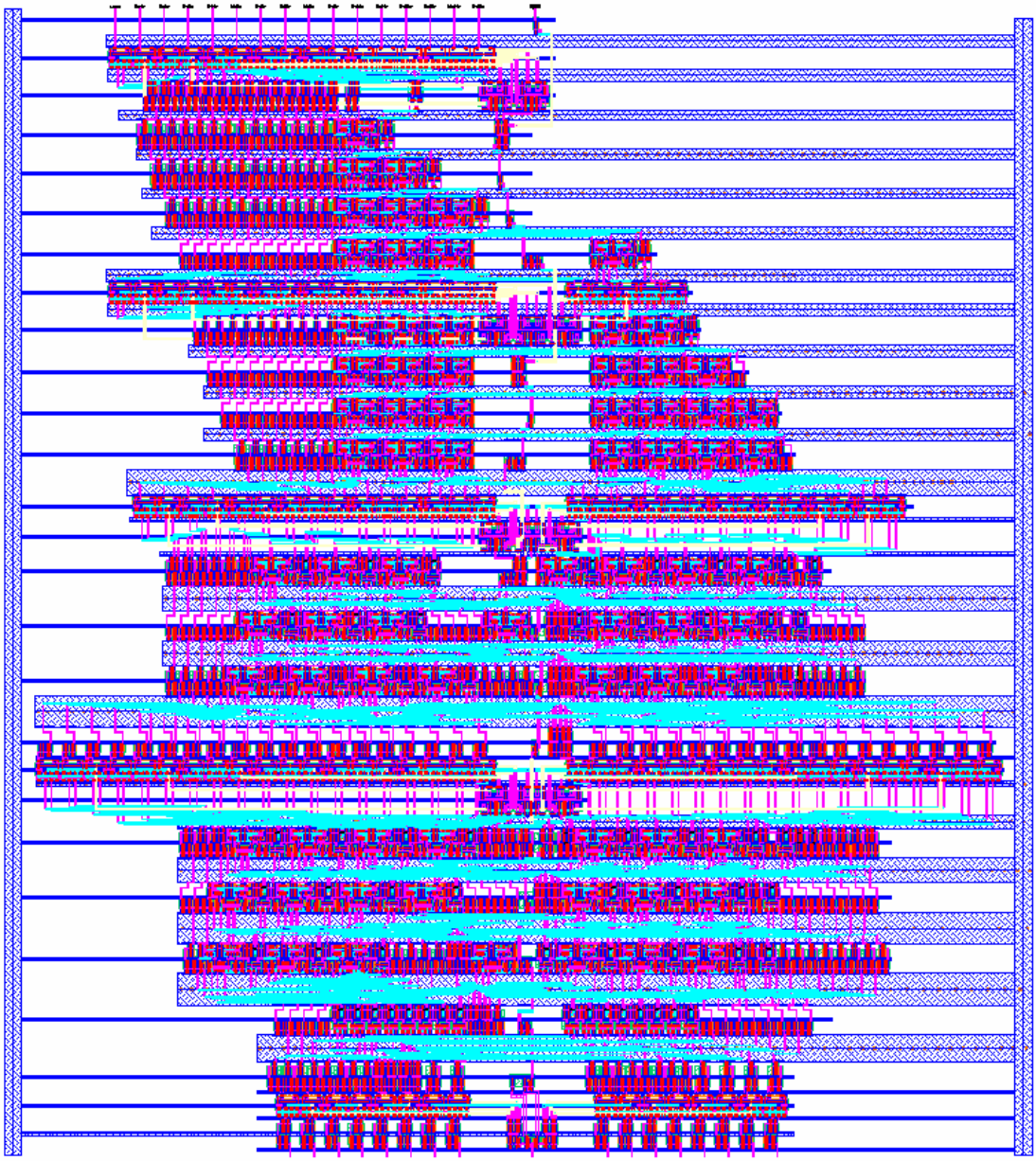


Fig. 5.13. 8×8-bit mesochronous pipeline multiplier layout (TSMC 180nm).

Chapter 6

Mesochronous power consumption and power supply current variation (di/dt)

In this chapter we present an 8×8-bit multiplier pipelined in the conventional pipeline (CPP) scheme and the novel mesochronous pipeline (MPP) scheme, to compare its power consumption. The power consumption is an important issue in chip design. In conventional pipeline scheme, huge currents draw by clock network and large number of pipeline registers is increasing the chip power consumption. Clock network's power consumption has increased to 50% of the total chip power consumption [7]. Power supply network is essentially a huge RLC network, and the huge currents drawn from it are causing higher *IR* drops in it. Increase in clock frequency, system size, and wire parasitic values is introducing power supply noise [21], [22]. Also, the large current slew rates (*di/dt*) coupled with on-chip inductance are generating significant amount of *Ldi/dt* noise on power supply. These power supply noise affect the power supply integrity and this is worsened due to decreasing supply voltage levels.

The results presented in this chapter prove that the mesochronous multiplier implementation consumes less power than conventional implementation. Also, the variation in current drawn from power supply is less in mesochronous scheme.

6.1. Carry-Save Adder multiplier implementation

6.1.1. Conventional implementation of CSA multiplier

To achieve a fast multiplier the CSA architecture must be pipelined. In CPP scheme according to (1.1) minimum clock period can be achieved by making each of the *2M* layers into stages of a pipeline, separated by pipeline registers. Effectively, an *M*-bit CPP multiplier would have *2M* stages with *2M+1* pipeline registers. An 8×8-bit pipelined multiplier implemented has 16 pipeline stages and 17 sets of inter-stage registers. The schematic of this multiplier was shown in Chapter 5 and is repeated here in Fig. 6.1.

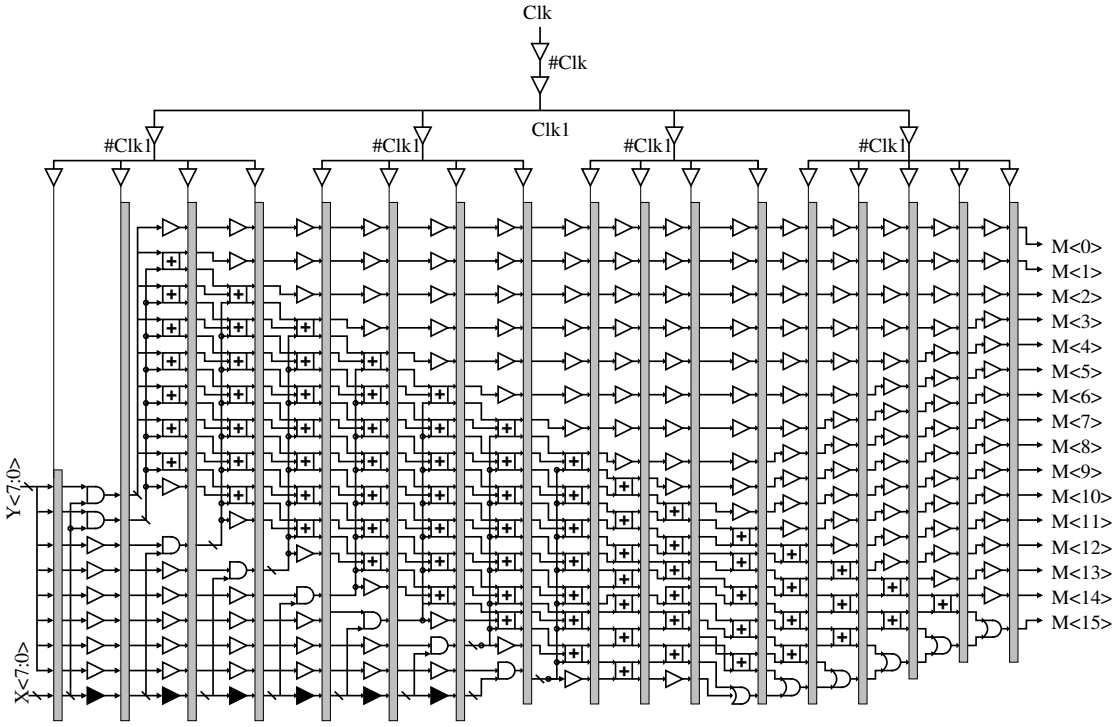


Fig. 6.1. 8×8-bit CSA multiplier implemented in CPP scheme.

To distribute the clock signal to all the pipeline register stages, a tree network has been used as shown in Fig. 6.1. Inverters have been used in place of buffers, and a fan-out of four has been used. The inverters in the tree network have sizes 50, 40, 25, 10 times the

minimum sized inverter. Each register stage has another small tree network to deliver the clock to all the flip-flops in that stage without any vertical skew.

6.1.2. Mesochronous implementation of CSA multiplier

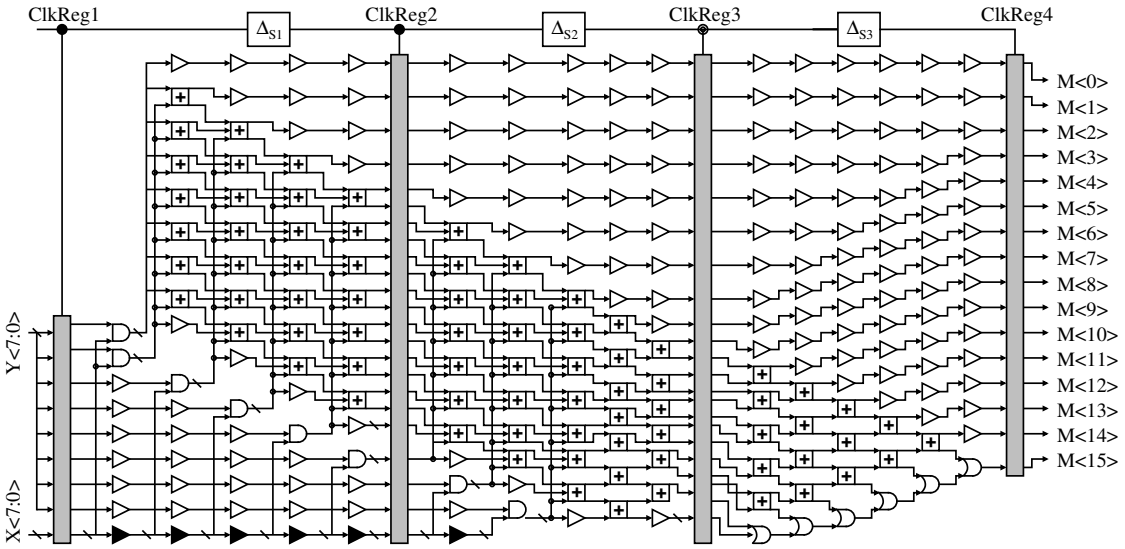


Fig. 6.2. 8×8-bit CSA multiplier implemented in MPP scheme.

Fig. 6.2 shows the schematic of the same 8×8-bit multiplier implemented in MPP scheme. Here the idea is to increase the amount of logic in a stage and clock the pipeline registers such that there are multiple data sets simultaneously present in a logic stage at different stages of processing. All of the logic enveloped between any two adjacent register stages supports multiple data sets simultaneously. Also, the number of register stages required to synchronize the data sets is small. In this implementation there are only 3 pipeline stages and 4 register stages. The placement of the registers is based on the maximum delay difference that can be handled for a target clock frequency. This implementation is different from the one presented in Chapter 5, and the reason for this will be explained later in this section. Unlike a tree distribution for clock signal in CPP

scheme, the clock signal takes a linear path in MPP scheme as shown in Fig. 6.2. The clock travels close to the data path and includes delay elements realized using simple inverters.

The registers in the multiplier have been realized using a dynamic two-phase D flip-flop [14], [15]. This cell is simple to implement and the minimum clock period requirement observed in SAFF implementation (Chapter 5) is less in the dynamic two-phase D-FF. Also, the flop-flop timing values like set-up time, hold time and clock-to-Q delay are less in the dynamic two-phase D-FF. The schematic of this flip-flop is shown in Fig. 6.3.

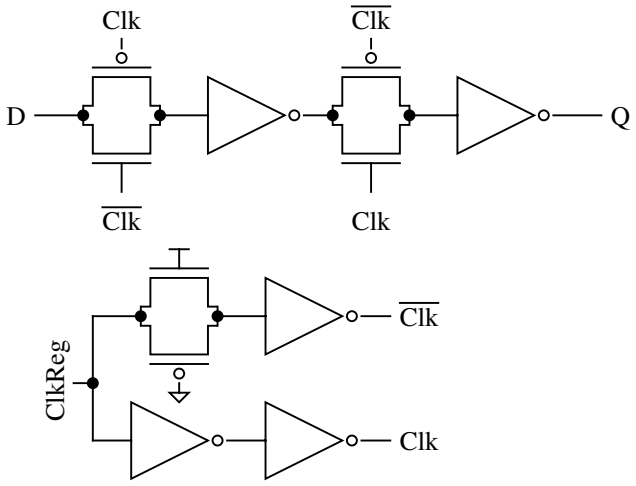


Fig. 6.3. D flip-flop and clk & $\overline{clk}$ circuit.

From simulations, the clock-to-output delay, set-up time, and hold time can be calculated. These values are shown in Table 6.I

TABLE 6.I. DYNAMIC TWO PHASE D-FF TIMING VALUES

Setup time (t_s)	65ps
Hold time (t_h)	5ps
Clock-to-Q delay (D_R)	130ps

In the CPP implementation of the multiplier, the minimum achievable clock period can be calculated from (1.1)

$$T_{clk\_cpp} > D_{max} + D_R + t_s = 280 + 130 + 65 = 475\text{ps}$$

A fair compare between the CPP and MPP schemes in terms of power consumption is when they are operating at the same clock period. For this purpose a clock period of 500ps (2GHz) has been chosen. In the MPP multiplier implementation, for a clock period of 500ps, the maximum delay variation of any stage can be calculated using (2.5) as 400ps.

$$d_{max(j)} - d_{min(j)} \leq T_{clk\_mpp} - (t_s + t_h + 2\Delta_{clk}) = 500 - 100 = 400ps$$

The placement of registers as shown in Fig. 6.2 is based on this calculated limit on delay difference. The delay variation of the FA is 70ps, and maximum calculated delay variation is 400ps, and so maximum number of FA layers in a stage is five. This placement also accommodates additional variations that can occur in a stage. From Fig. 6.2 it can be seen that stage 2 is the critical stage as it has five FA/HA layers combined into a single stage. The logic enclosed between any two adjacent register stages supports two or more data sets simultaneously and the stage delay difference is less than 400ps.

6.2. Power consumption and power supply current variation

Simulations have been performed to calculate the average current drawn by the clock network, registers, and logic in both the pipeline schemes. In this section the power consumption by the three components is discussed and the CPP and MPP schemes are compared.

6.2.1. Clock network

In the CPP multiplier, a tree network has been used to distribute clock to all the register stages. The small tree network used to distribute clock to all flip-flops in a register stage has also been included in the global clock network for power consumption calculations. The current drawn by the clock network in CPP scheme is shown in Fig. 6.4. In Fig. 6.4 the signals (Clk, #Clk, Clk1, #Clk1, Clk2, #Clk2) show the clock at various stages of the tree distribution network. The peak current drawn from power supply line and peak discharge current to the ground line clearly coincide with the switching event of the clock applied to the pipeline registers. This is due to the large number of pipeline registers that have to be driven simultaneously in CPP scheme. The average value of current drawn by the clock network is 86.9mA.

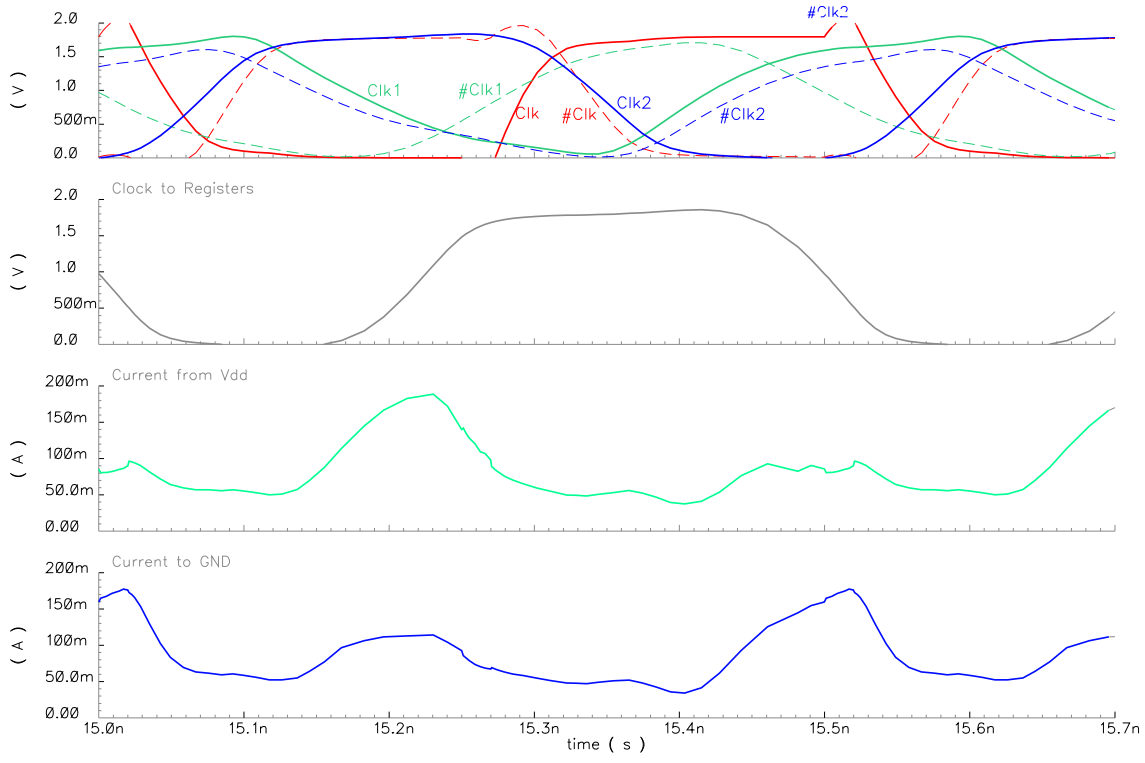


Fig. 6.4. Clock network current in CPP scheme at 2GHz.

In the MPP scheme, the clock signal takes a linear path and travels clock to the data path. The current drawn by the clock network in MPP scheme is shown in Fig. 6.5. The current draw here is for an implementation where maximum delay in clock path is derived using physical delay elements (N=1). So, large delay values are present in the clock path. In Fig. 6.5 ClkReg1, ClkReg2, ClkReg3, ClkReg4 signals are the clock signals applied to the first, second, third and fourth register stages respectively. Due to the clock distribution approach taken in MPP, the registers are not triggered at the same time, which is clear from Fig. 6.5. The average current drawn by the clock network in this implementation is 53mA. When compared to the CPP scheme, the current drawn in this case is less. This means significant power savings in clock network.

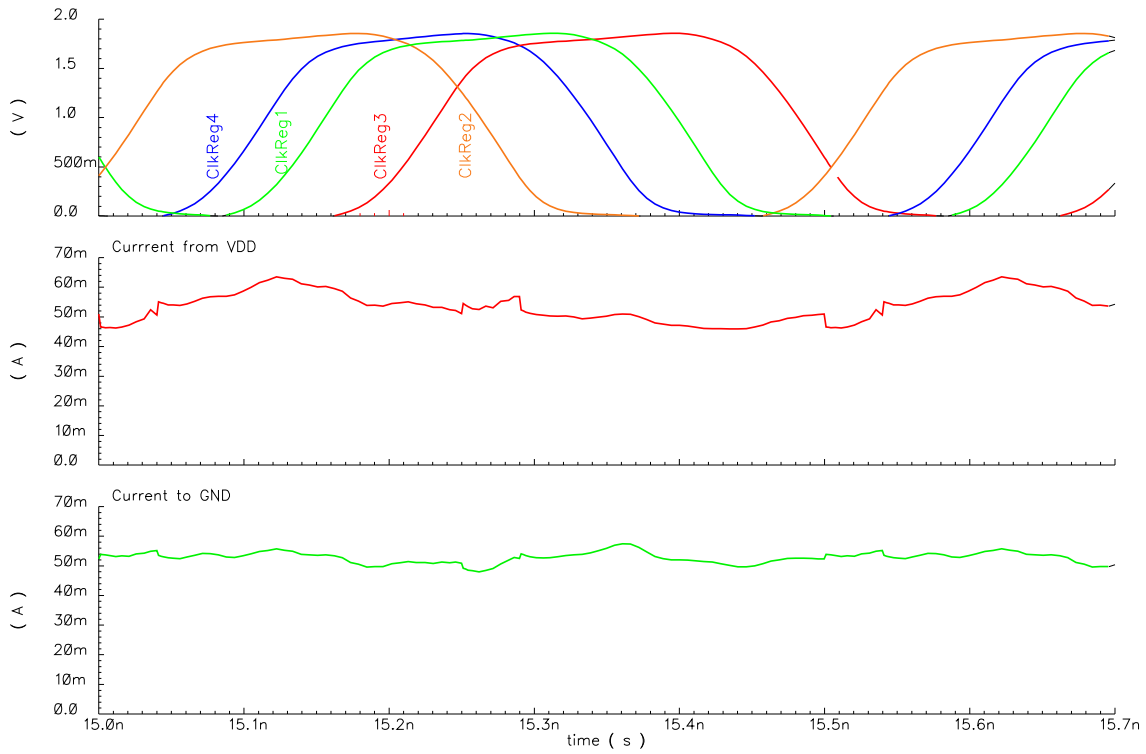


Fig. 6.5. Clock network current in MPP scheme at 2GHz.

The power consumed by the clock network in MPP scheme can be further reduced by taking advantage of the clock periodicity as discussed in Chapter 2. When the necessary delays in the clock signal path are realized using the periodic nature of the clock signal, small delay values are required in the clock path. This results in less power consumption. Fig. 6.6 shows the current drawn in this case (N=4) and the average current drawn is 24mA.

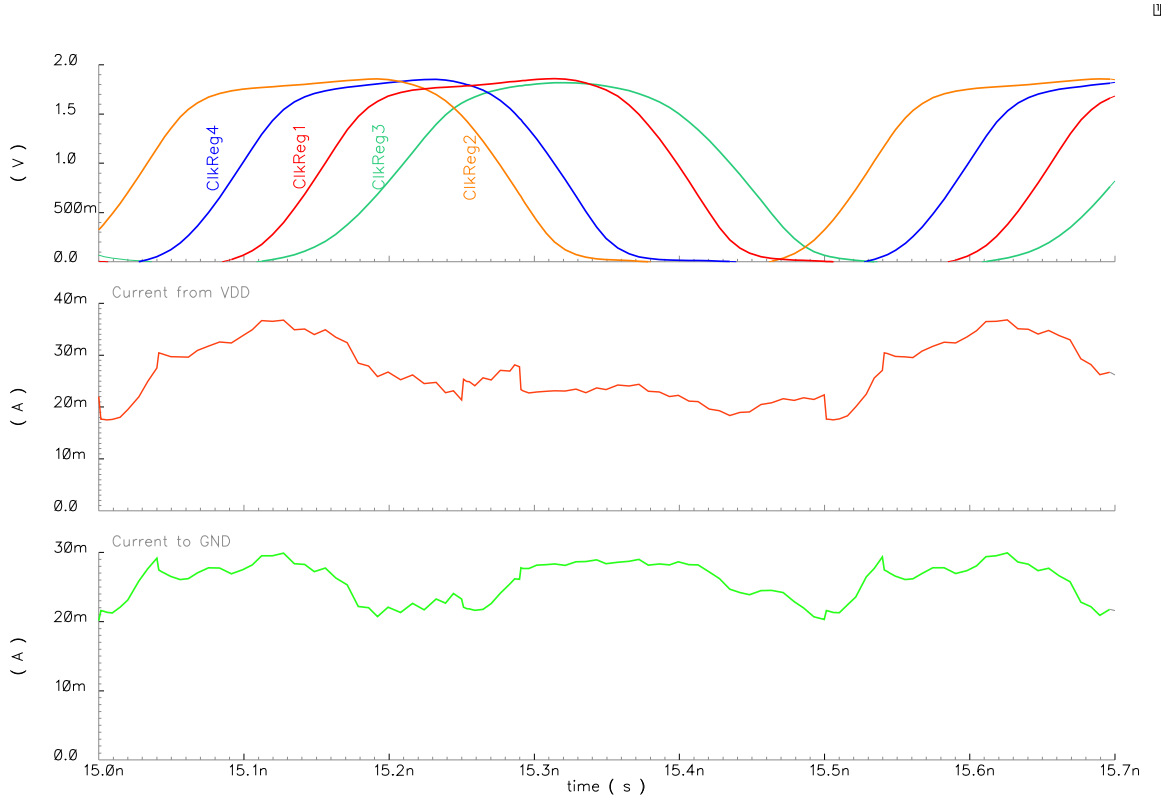


Fig. 6.6. Clock network current in MPP scheme at 2GHz with reduced clock delay.

Consider the current drawn by clock network in case of CPP scheme as shown in Fig. 6.7. The slew rate (di/dt) of the current from Vdd is approximately 1.23V/ns. Similarly the slew rate of current discharged into the ground rail is approximately 1.67V/ns (Fig. 6.4). The large currents drawn can induce a large IR drop on the supply network, while the large current slew rates (as shown in Fig. 6.7) can generate significant Ldi/dt noise

[21], [22]. These drops are aggravated by technology scaling, decreasing supply voltages and increasing clock frequencies. These voltage fluctuations can be suppressed by increasing the on-chip decoupling capacitance, however this results in increased die size and cost. Consider the case of MPP scheme as shown in Fig. 6.7, the current drawn by the clock network is relatively small and has less variation compared to current in CPP scheme. This means less power supply noise is induced in MPP scheme.

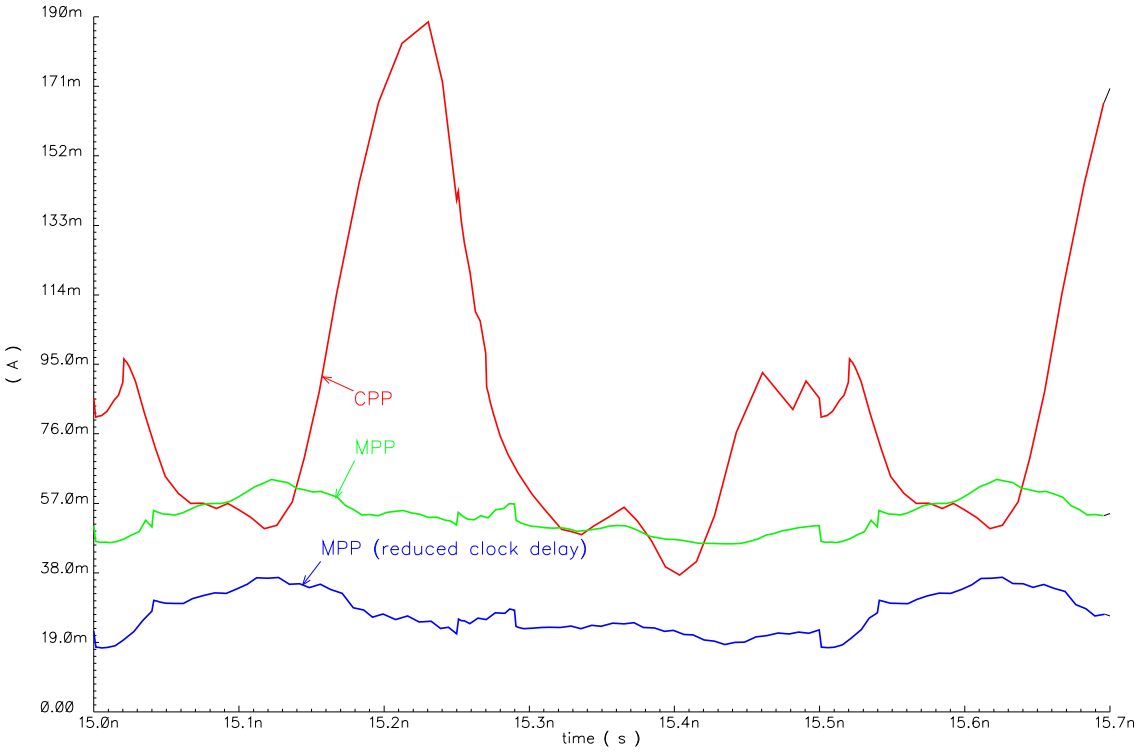


Fig. 6.7. Clock network current (from Vdd) at 2GHz.

The power consumption by clock network in CPP scheme can be reduced by operating the system at a low speed. The CPP multiplier when simulated at 667MHz, its clock network consumed an average current of 32.1mA which is close to the value achieved in the MPP scheme with reduced clock path delay. So to achieve similar power consumption, the CPP multiplier must be operated at one-third the speed of the MPP

multiplier. The clock network current consumption values are shown for various cases, in Table 6.II [17].

TABLE 6.II. CLOCK NETWORK CURRENT CONSUMPTION

Scheme	Current (mA)
CPP @ 2GHz	86.9
CPP @ 667MHz	32.1
MPP @ 2GHz	53.0
MPP @ 2GHz (reduced clock delay)	24.2

6.2.2. Pipeline registers and logic

The pipeline registers are the sources of high power consumption in CPP implementation after the clock distribution network. The average current drawn by the registers, and logic stages, is shown in Table 6.III [23]. The current drawn by the logic stages has been calculated for a significant activity in these stages. Fig. 6.8 and Fig. 6.9 show the plots of currents drawn by the register stages and logic stages in CPP multiplier and MPP multiplier implementations during a clock period.

TABLE 6.III. PIPELINE REGISTERS AND LOGIC CURRENT CONSUMPTION

Scheme	Current (mA)	
	Registers	*Logic*
CPP @ 2GHz	66.6	38.2
CPP @ 667MHz	21.2	9.3
MPP @ 2GHz	12.8	45.3

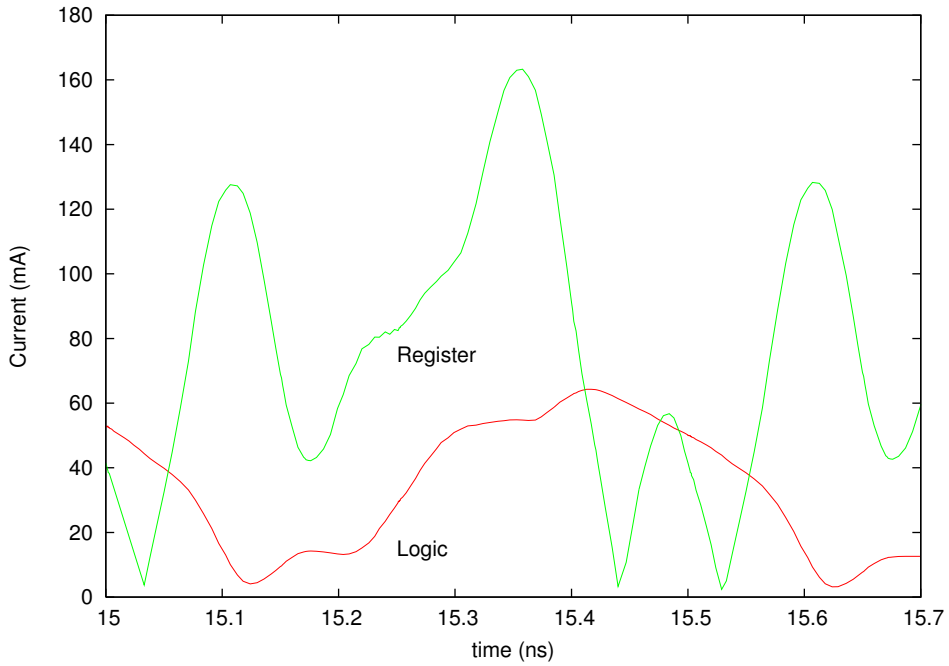


Fig. 6.8. Current drawn by registers and logic in CPP scheme at 2GHz.

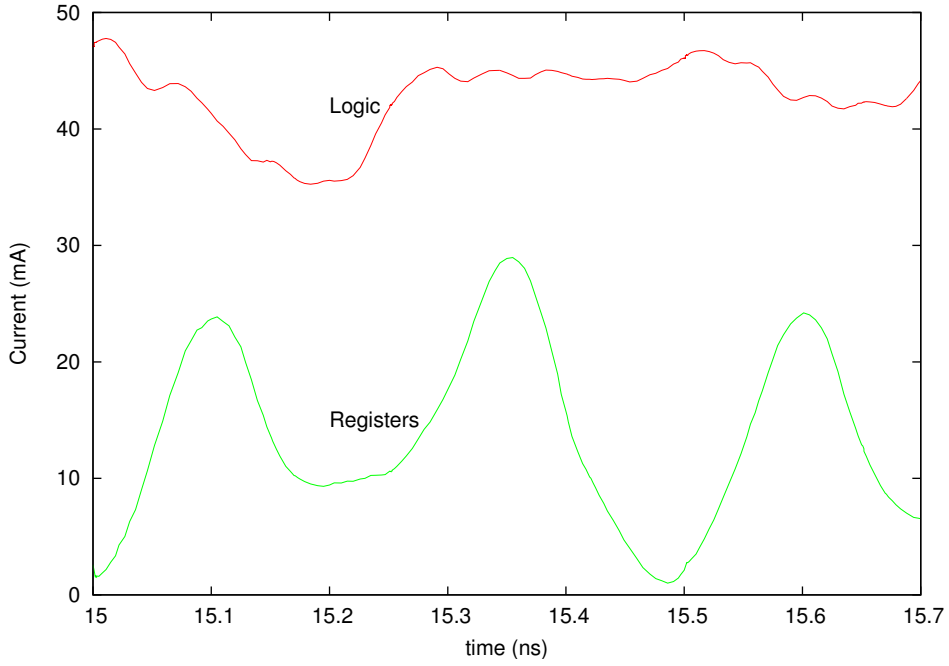


Fig. 6.9. Current drawn by registers and logic in MPP scheme at 2GHz.

In the CPP multiplier implementation shown in Fig. 6.1, there are 17 pipeline register stages, while in the MPP multiplier implementation shown in Fig. 6.2, there are only 4 register stages. Due to the small number of register stages in MPP multiplier the overall current consumed in the register stages is significantly less than in the CPP multiplier.

The current drawn by the logic portion of the multiplier should be similar in both the schemes, since the logic is identical. From Table 6.III it can be seen the current values are close in both the scheme. However the small increase in current drawn by logic in MPP multiplier can be attributed to the additional logic necessary to decrease the logic variation ($d_{max} - d_{min}$) in the pipeline stages.

6.2.3. Total power

The over-all current drawn by the CPP implementation is approximately 192mA, while the current drawn by the MPP multiplier is 82mA. This shows that significant power savings are possible in MPP scheme. Fig. 6.10 shows the plot of total current drawn by multiplier implemented in CPP and MPP scheme. The numerical values of currents drawn by the clock network, register stages and logic are shown in Table 6.IV [23]. The high currents drawn in CPP scheme imply higher power consumption and higher *IR* drops in the power supply network. Apart from drawing higher current, the variation in current drawn is higher in CPP scheme which could result in higher power supply noise.

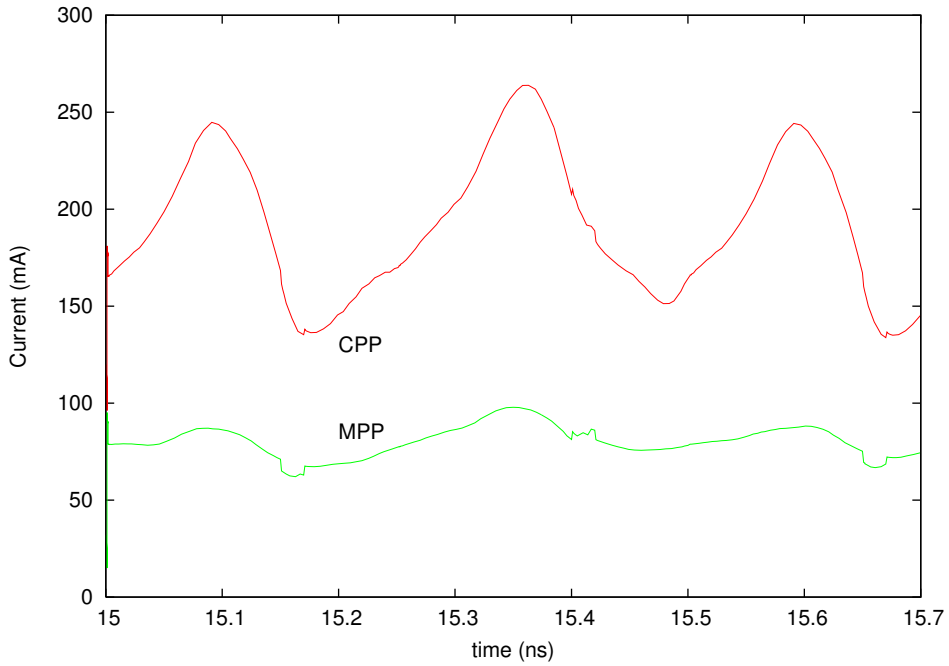


Fig. 6.10. Total current in CPP and MPP (reduced clock delay) schemes at 2GHz.

TABLE 6.IV. CLOCK NETWORK REGISTERS, AND LOGIC CURRENT

Scheme	**Current (mA)**			
	Clock network	***Registers***	***Logic***	***Total***
CPP @ 2GHz	86.9	66.6	38.2	191.7
CPP @ 667MHz	32.1	21.2	9.3	62.6
MPP @ 2GHz	24.2	12.8	45.3	82.3
CPP/MPP @ 2GHz	3.6	5.2	0.84	2.3

A graphical comparison of the current results is shown in Fig. 6.11. In CPP scheme, the amount of current drawn by the clock network and registers is greater than in MPP implementation. This is due to the complex clock distribution and higher number of register stages in CPP. The overall current drawn, in turn power consumption is significantly higher in CPP scheme. Fig. 6.12 shows a bar-graph of current drawn by clock network, registers and logic for both the schemes. On an average, the current drawn

by the logic stages in MPP scheme is higher than CPP scheme, which represents useful current drawn, as it is used for computation.

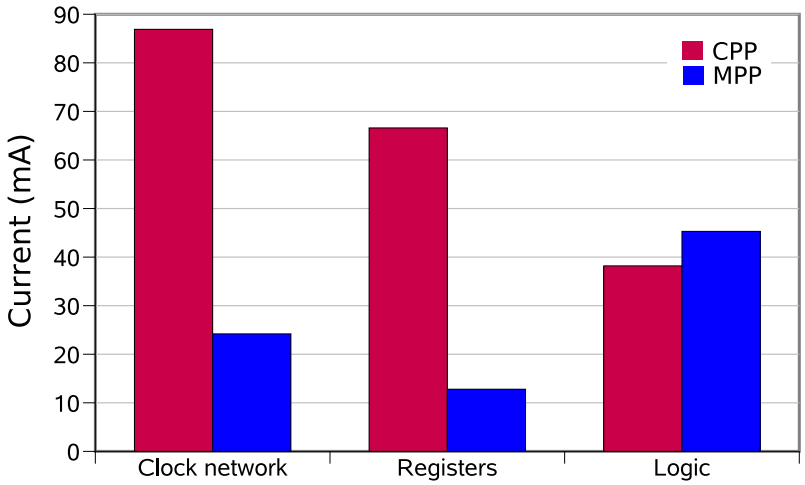


Fig. 6.11. Total current in CPP and MPP (reduced clock delay) schemes at 2GHz.

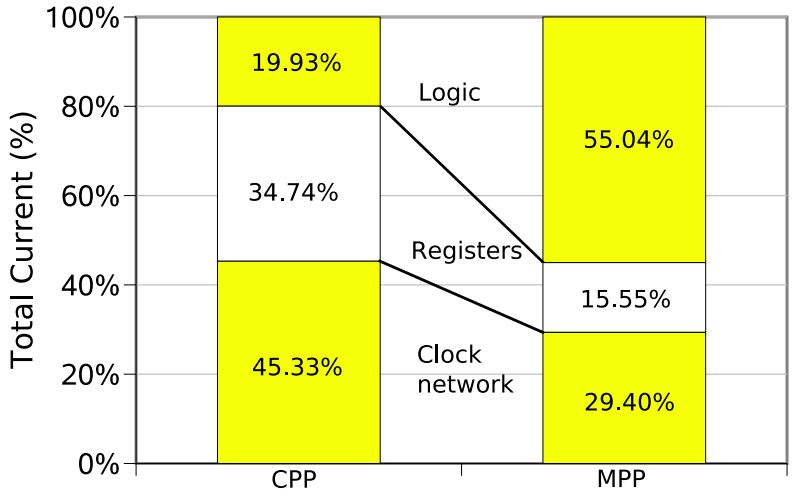


Fig. 6.12. Total current breakdown in CPP and MPP schemes @ 2GHz.

The CPP multiplier implementation has been simulated at clock frequencies 2GHz (clock period=500ps), 1.33GHz (clock period=750ps), 1GHz (clock period=1ns), and 800GHz (clock period=1.25ns). In CPP scheme, clock period is determined by the stage with largest delay value. For large values of clock period, more logic can be included per

stage and few stages are required to pipeline system. The clock frequencies show above, have been chosen according to the following equation.

$$T_{clk\_cpp} > d_{\max\_FA} + (M-1)d_{avg\_FA} + D_R + t_s$$

In the above equation, M is the number of adders (FA or HA) considered as a single stage and d_{avg_FA} is the average propagation delay of the FA. Considering the average delay value would give a typical estimate of clock period. Considering the maximum propagation delay (d_{max}) value would give a pessimistic estimate of clock period and considering the minimum propagation delay (d_{min}) would be an optimistic estimate. The possible clock periods for various values of M are shown in Table 6.V.

TABLE 6.V. CPP CLOCK PERIOD FOR VARIOUS VALUES OF M

M	No. of stages	Clock period	Clock period chosen
1	16	$T_{clk_cpp} > 475$ps	500ps
2	8	$T_{clk_cpp} > 720$ps	750ps
3	5	$T_{clk_cpp} > 965$ps	1000ps
4	4	$T_{clk_cpp} > 1210$ps	1250ps

Table 6.VI also shows the current drawn by the CPP multiplier at different clock frequencies. From the results shown in Table 6.VI, it is clear that CPP scheme consumes less current (power) than MPP scheme only if operated at half the speed of MPP.

TABLE 6.VI. CLOCK NETWORK, REGISTERS, AND LOGIC CURRENT (CPP SCHEME)

Scheme	No. of stages	Current (mA)			
		Clock network	***Registers***	***Logic***	***Total***
CPP @ 2GHz	16	86.9	66.6	38.2	191.7
CPP @ 1.33GHz	8	40.3	25.2	31.8	97.3
CPP @ 1GHz	5	23.8	13.1	22.9	59.8
CPP @ 800MHz	4	16.6	7.1	9.7	33.4
MPP @ 2GHz	4	24.2	12.8	45.3	82.3

The trend of current consumption of the CPP multiplier is shown in Fig. 6.13.

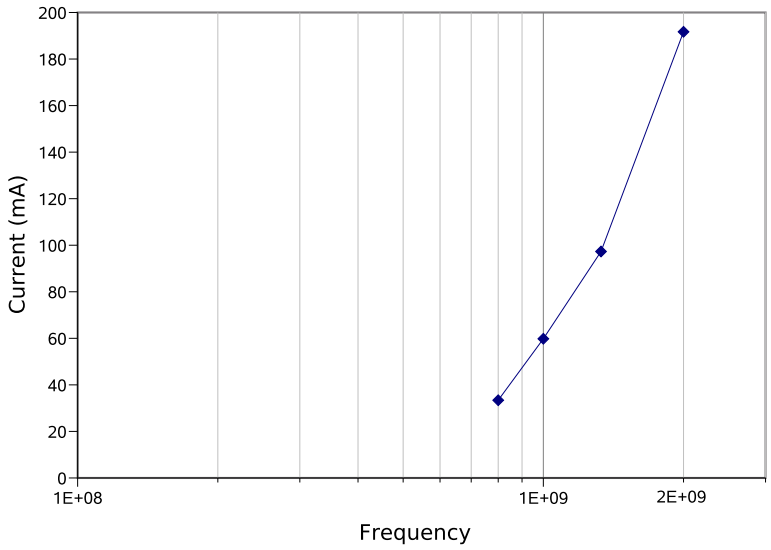


Fig. 6.13. Current consumption of CPP multiplier at various clock frequencies.

6.3. Summary

The following is a summary of important points from this chapter.

- *Simpler clock distribution*. In the CPP multiplier implementation, the clock signal must be distributed to all the 17 pipeline registers stages such that they are all triggered simultaneously. In the MPP scheme clock signal path is parallel to data path. Delays are included in the clock signal path so that clock signal can travel with data. Also, there are only 4 register stages in MPP multiplier implementation, so load on clock network is less. In implementing the clock path delay elements, periodic nature of clock signal can be used to further reduce power consumption.
- *Low power dissipation*. The average power dissipation in MPP multiplier implementation is 148.05mW, while in CPP implementation is 345.6mW at clock frequency of 2GHz.

- *Clock network and registers*: In the CPP multiplier, clock distribution network and registers account for 80% of total power consumption. In the MPP multiplier logic dissipates more power compared to clock network and registers.
- *Lower power supply noise*. In MPP multiplier implementation, due to the linear clock distribution approach, there are fewer register stages and they all are not triggered simulatneously. This reduces the current drawn and also the rate (*di/dt*) at which it is drawn. The result is less variation in current drawn by clock network. This means less power supply noise.
- *CPP Power-performance tradeoff.* CPP scheme can achieve similar power consumption as MPP scheme only when operated at a much slower speed. The CPP multiplier implementation consumes less power than the MPP implementation only if operated at half the frequency of MPP multiplier.

Chapter 7

Tiny Chip

In this chapter we shall discuss the implementation of a 4×4-bit mesochronous pipeline Carry-Save Adder (CSA) multiplier in AMI 0.5µm, 5.0V technology. The design has been fabricated through The MOSIS service. This chip has been tested using Onehotlogic chip tester and we shall discuss the results obtained from these tests.

7.1. 4×4-bit mesochronous CSA multiplier simulations

The schematic of a 4×4-bit CSA multiplier is shown in Fig. 7.1. This multiplier has to be pipelined to achieve high performance.

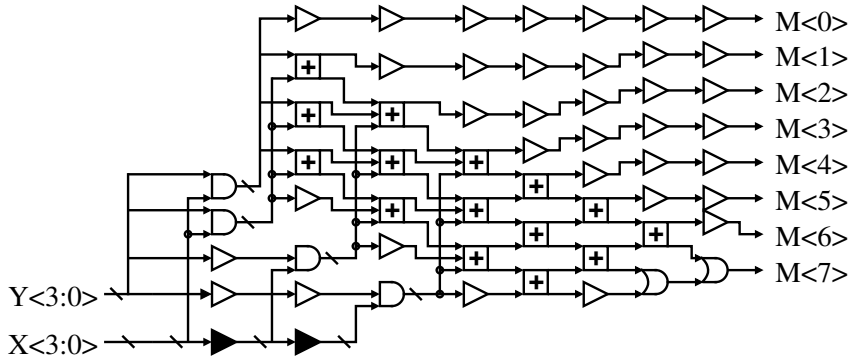


Fig. 7.1. 4×4-bit CSA multiplier schematic.

All the basic cells used in this implementation are same as the ones used in 8×8-bit CSA multiplier presented in Chapter 5 and Chapter 6. Extensive simulations have been performed on the differential transmission gate full adder (FA) in AMI 0.5µm, 5.0V

technology. For the 32 input transitions that trigger a change in one or both of the FA outputs, propagation delay was measured. Propagation delay values obtained for these 32 transitions are graphically represented in Fig. 7.2. Using this plot, minimum and maximum delays values and delay variation of FA can be calculated. These values are shown in Table 7.I.

TABLE 7.I. FULL ADDER DELAY VALUES

Maximum propagation delay (d_{max})	740ps
Minimum propagation delay (d_{min})	460ps
Delay variation ($d_{max} - d_{min}$)	280ps

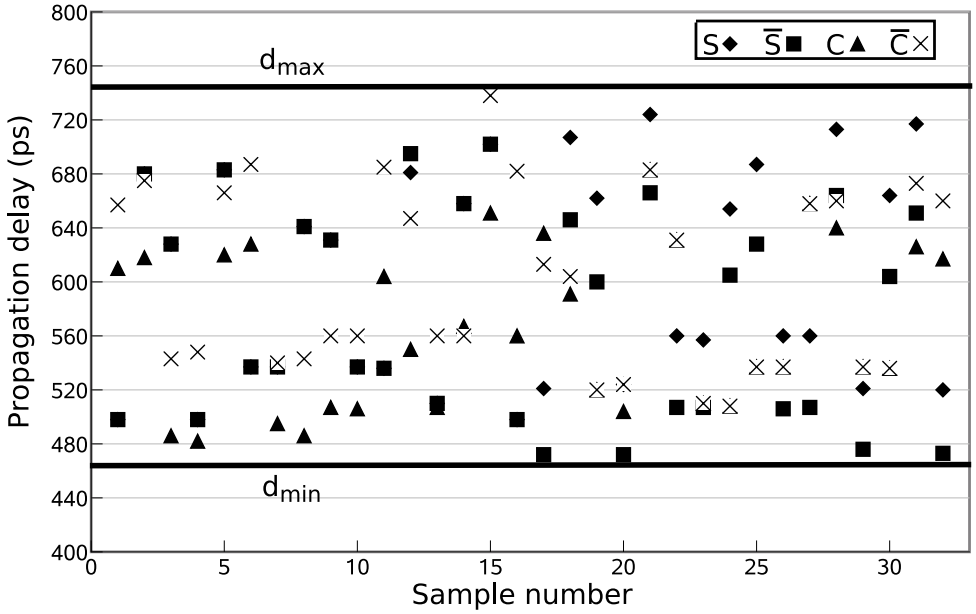


Fig. 7.2. Propagation delay of the full adder.

From Table 7.I we see that the propagation delay of the full adder varies from 460ps (d_{min}) to 740ps (d_{max}), resulting in a maximum delay variation of 280ps.

The limiting factor in this design is the clock generator. A ring oscillator with a multiplexer has been used to generate four different clock periods. The schematic of this

clock generator is shown in Fig. 7.3. The clock periods achieved from the clock generator for various values of the selection inputs (S1, S0) are shown in Table 7.II.

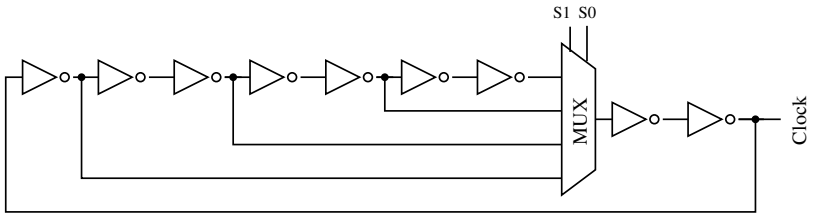


Fig. 7.3. Clock generator schematic.

TABLE 7.II. CLOCK GENERATOR RESULTS

Selection Inputs		Clock period	Clock frequency
S1	S0		
1	1	1.95ns	513MHz
1	0	2.22ns	450MHz
0	1	2.51ns	400MHz
0	0	2.88ns	347MHz

To view this clock signal externally, the clock was slowed down (by an order of 2^{18}), using a chain of JK flip-flops. For an internal clock period of 1.95ns, when multiplier by 2^{18}, the external clock period should be 458.75μs.

Since the minimum clock period is 1.95ns and maximum propagation delay (d_{max}) of the FA is 740ps, the best scheme to pipeline is to have two FA/HA per stage as shown in Fig. 7.4. From this schematic it is clear that system would have 4 logic stages and 5 pipeline register stages and would require a global clock distribution.

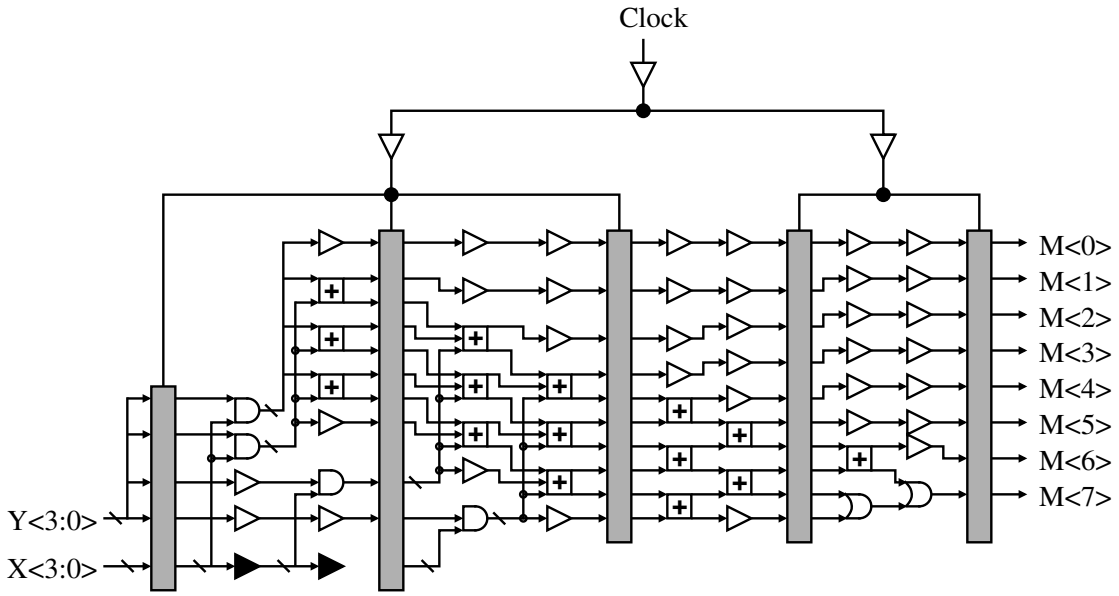


Fig. 7.4. Conventional 4×4-bit CSA multiplier schematic.

Using the mesochronous pipeline approach, the multiplier can be operated at the minimum clock period of 1.95ns, with only two pipeline stages and simple clock distribution. The schematic of this implementation is shown in Fig. 7.5.

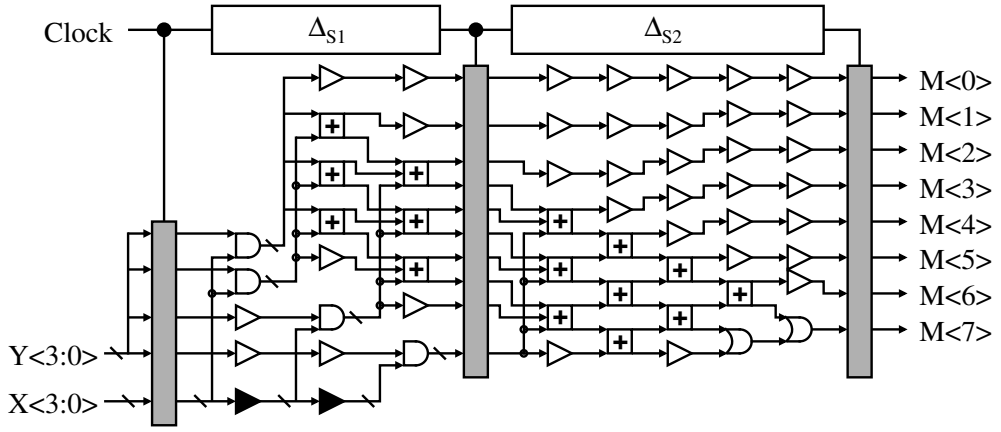


Fig. 7.5. Mesochronous 4×4-bit CSA multiplier schematic.

In this implementation the stage delay values have been calculated from simulations in AMI 0.5μm, 5.0V technology and are shown in Table 7.III.

TABLE 7.III. STAGE DELAYS IN MESOCHRONOUS CSA MULTIPLIER

Stage	Delay
1	2.85ns
2	3.3ns

From the stage delay values shown in Table 7.III and clock period of 1.95ns, it is clear that in the two logic stages two separate data waves can be present simultaneously. This mesochronous multiplier is successfully able to operate on a clock period of 1.95ns (513MHz) and only requires 2 logic stages and 3 pipeline registers. This is definitely a performance gain. Also, the clock distribution is simple and the delay elements in the clock signal path have been realized using simple inverters. The layout of this multiplier is shown in Fig. 7.10.

From the delay values, we can estimate the clock period of conventional multiplier with only 2 logic stages and 3 registers stages. The conventional multiplier can only operate at 303MHz (Stage 2 delay is 3.3ns), while the mesochronous multiplier can operate at 513MHz, which is a *Speedup* of 1.69. In Table 7.IV a comparison between the mesochronous and conventional multiplier implementations is presented.

TABLE 7.IV. PERFORMANCE COMPARISON

Scheme	Conventional		Mesochronous
No. of pipeline stages	4	2	2
No. of pipeline registers	5	3	3
Clock frequency	513MHz	303MHz	513MHz
Clock distribution	Complex	Simple	Simple

To facilitate the test of this design when fabricated, two slow speed memory banks have been incorporated into the multiplier. One bank is at the input, in which operands can be

stored and the other bank is at the output, which stores the multiplication result. Operands can be written to the input bank at very slow speed, using external control and data signals. Operands are read from this bank at the system speed and applied to the inputs of the multiplier. Similarly, the output bank stores the multiplier output at the system speed and can be read through external pins at a slower rate. The schematic of the memory element in these banks is shown in Fig. 7.6.

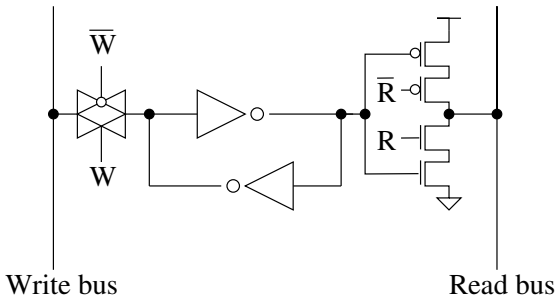


Fig. 7.6. Memory element in Input/Output bank.

7.2. 4×4-bit mesochronous CSA multiplier chip test results

The Mesochronous 4×4-bit CSA multiplier shown in Fig. 7.5 has been fabricated in AMI 0.5μm, 5.0V technology. This chip has been tested using Onehotlogic chip tester.

The SPICE parameters from the AMI fabrication run have been used to re-simulate the basic cells in the multiplier. Due to difference in the SPICE parameters from the fabrication run and the ones used for simulations, all the delays are scaled-up by a factor of 2.05 in the fabricated chip.

The chip test results of the externally monitored slow version (order of 2^{18}) of internal clock signal for various values of the control inputs (S1, S0) are shown in Fig. 7.7. The clock period values are shown in Table 7.V

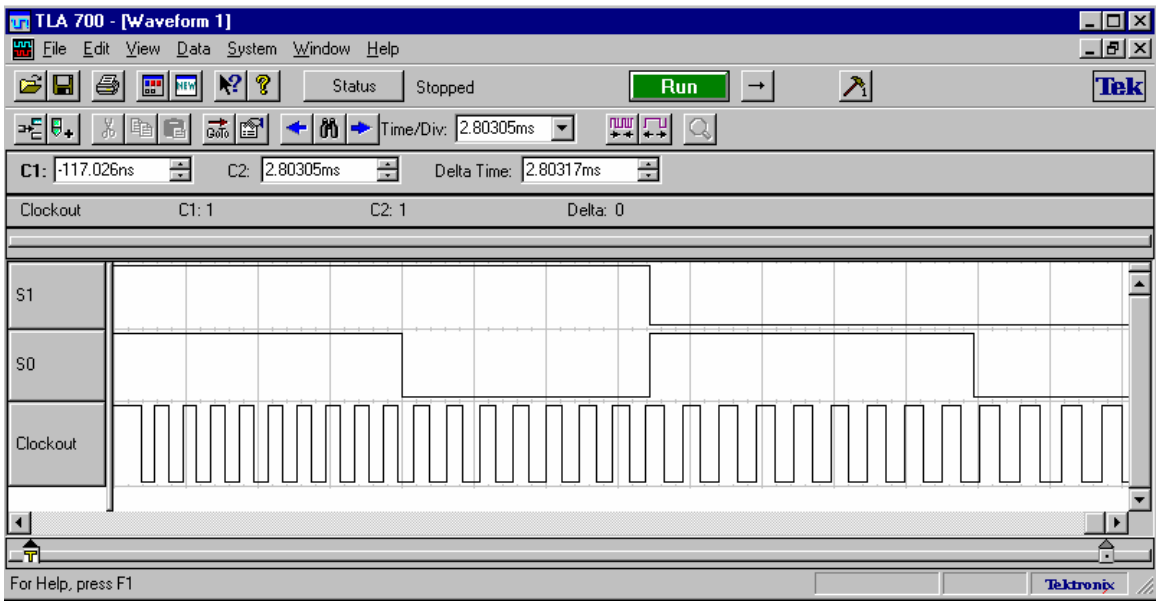


Fig. 7.7. Internal clock signal from the chip.

TABLE 7.V. SCALED INTERNAL CLOCK SIGNAL PERIOD

Selection Inputs		External clock period	Internal clock period
S1	S0		
1	1	1.04ms	3.97ns
1	0	1.21ms	4.62ns
0	1	1.34ms	5.11ns
0	0	1.56ms	5.95ns

Based on these results we can estimate the internal propagation delays. Some of the important delay values adjusted to the chip SPICE parameters are shown in Table 7.VI.

TABLE 7.VI. ADJUSTED DELAY VALUES

FA maximum propagation delay (d_{max})	2.05×740ps = 1517ps
FA minimum propagation delay (d_{min})	2.05×460ps = 943ps
FA delay variation ($d_{max} - d_{min}$)	2.05×280ps = 574ps
Mesochronous multiplier stage 1 delay	2.05×2.85ps = 5.84ns
Mesochronous multiplier stage 2 delay	2.05×3.3ns = 6.76ns
Internal clock period (S1=1, S0=1)	3.97ns (252MHz)

Tests performed on the chip with various input vectors proved that the system was able to operate on a clock period of 3.97ns (252MHz). Some of the chip test results are shown in Fig. 7.8 and Fig. 7.9. In these figures, the operands are shown with the label *Inputs(Y, X)*. The multiplicand is the most significant bits, while the multiplier is the least significant bits.

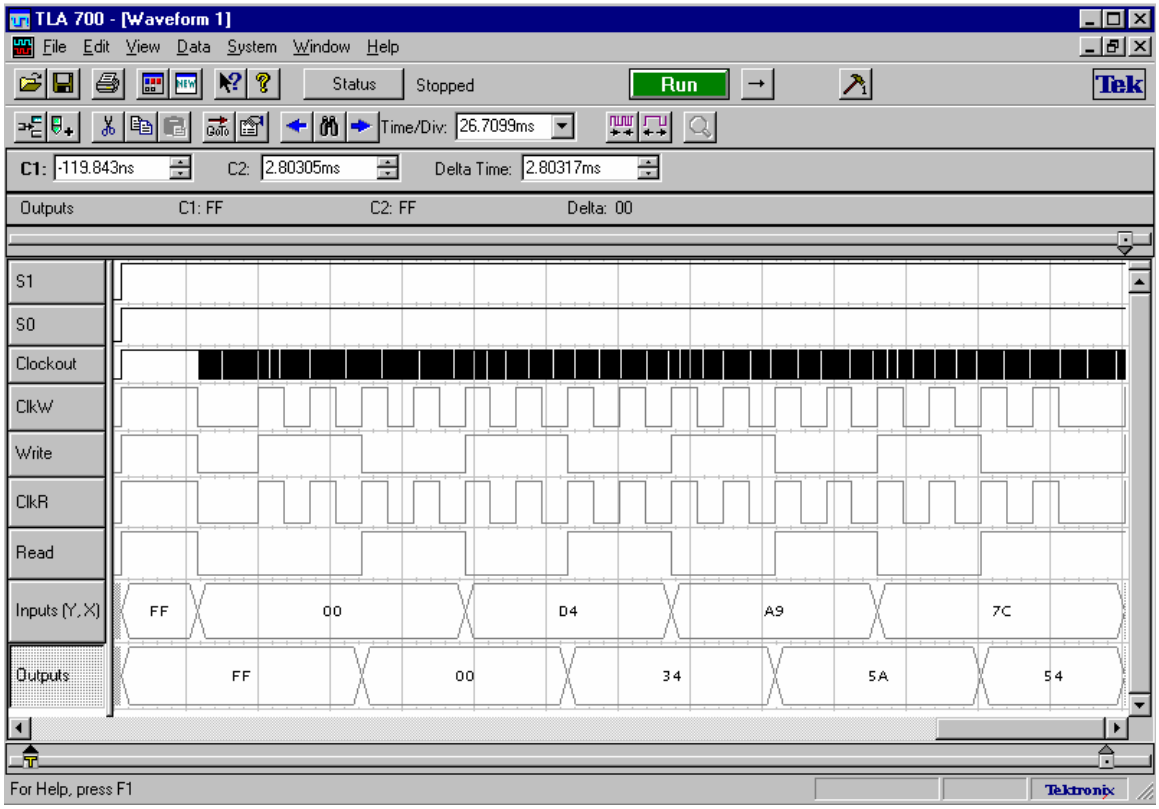


Fig. 7.8. Chip test results (Sample 1).

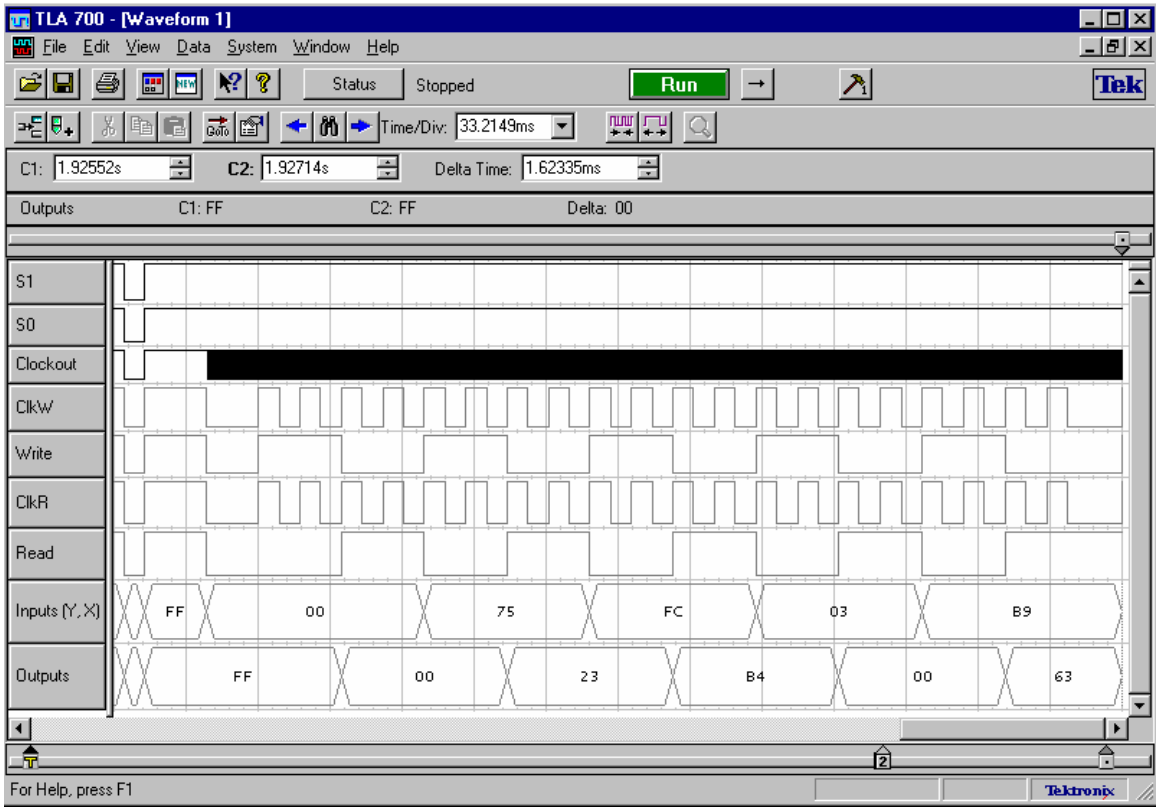


Fig. 7.9. Chip test results (Sample 2).

NOTE: In the chip implementation due to a faulty interconnect in partial product generation, some of the multiplication results are erroneous. However, this does not affect the performance of the system.

7.3. Summary

In this section we shall present a summary of important points from this chapter.

- *Tiny Chip*: A 4×4-bit mesochronous pipeline CSA multiplier has been fabricated in AMI 0.5µm 5.0V technology.
- *Higher performance*: The mesochronous multiplier has a *Speedup* of 1.69 over conventional pipeline implementation with only two pipeline stages and three pipeline registers. The performance of mesochronous multiplier can be achieved in

conventional scheme, however this would require the CSA multiplier to be split into four pipeline stages and five pipeline registers and requires a global clock distribution.

- *Chip test*: The fabricated chip has been tested and it works successfully at a frequency of 252MHz, which is significantly for an old technology.

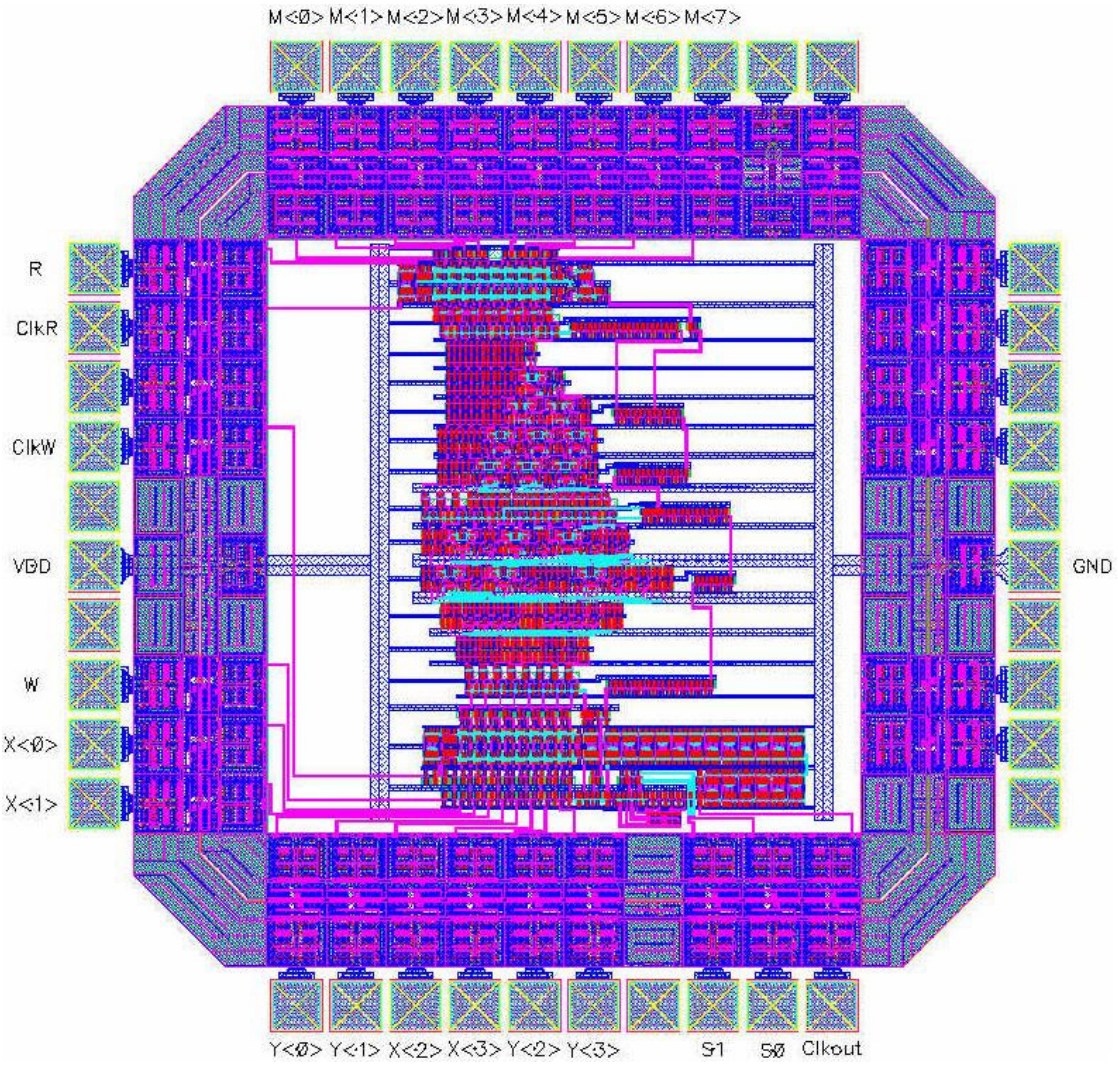


Fig. 7.10. Mesochronous 4×4-bit CSA multiplier layout.

Chapter 8

Concluding Remarks

In this dissertation, mesochronous pipeline (MPP) architecture has been presented which achieves better performance and power savings compared to conventional pipeline (CPP) architecture. The power savings, performance improvement and design aspects of this architecture have been discussed in detail here. A Carry-Save Adder (CSA) multiplier implemented in conventional and mesochronous pipeline architectures as a design example has been described in detail and the performance and power consumptions of the two implementations has been discussed. Following are the features of the MPP scheme in comparison with CPP scheme.

1) *Shorter clock period (T_{clk_mpp}).* The clock period in mesochronous pipeline scheme is determined by the pipeline stage with the largest difference between its minimum and maximum propagation delay. In conventional pipeline scheme, stage with maximum propagation delay dictates the minimum clock period achievable. Maximum delay difference is far less than maximum propagation delay, so smaller clock periods (i.e. higher clock frequencies) are possible in the proposed scheme.
2) *Smaller number of pipeline registers*. The performance achieved in conventional pipeline scheme can be easily achieved using mesochronous pipeline scheme with fewer pipeline stages and small number of pipeline registers.

3) *Simpler clock distribution*. In conventional pipeline scheme, clock signal must be distributed to all the pipeline registers stages such that they are all triggered simultaneously. In the mesochronous pipeline scheme, clock signal path is parallel to data path. Delays are included in the clock signal path so that clock signal can travel with data. This could cause the registers to be triggered at different times. Also because of fewer register stages in MPP scheme, load on clock network is less. This is a simpler clock distribution scheme and consumes less power compared to the distribution in a CPP scheme. In implementing the delay elements in clock path, periodic nature of clock signal can be used, so small delay elements are required. This helps further reduce power consumption in MPP scheme.

4) *Little influence of Clk-Q on* T_{clk_mpp}. The clock-to-output (*Clk-Q*) delay of pipeline registers has little influence on clock period in mesochronous pipeline as computation in a stage is spread over multiple clock periods. In conventional scheme, since computation in a stage is during a clock period, significant portion of clock period is lost in the *Clk-Q* delay and performance is affected. This is further aggravated by shrinking clock periods.

5) *Fast multiplier (350ps clock period).* A mesochronous pipeline implementation of an 8×8-bit Carry-Save Adder multiplier using modest TSMC 180nm technology, is able to operate on a short clock period 350ps (2.86GHz). In conventional pipeline scheme, the best clock period achievable is 595ps (1.68GHz). So, the mesochronous pipeline achieves a *Speedup* of 1.7 times. The number of pipeline stages and the number of pipeline registers required in this implementation is significantly less compared to conventional pipeline approach.

6) *Low power dissipation*. MPP implementation of the 8×8-bit carry-save adder multiplier using modest TSMC 180nm CMOS technology is dissipating less power compared to the CPP implementation. The average power dissipation in MPP implementation is 148.05mW, while in CPP implementation is 345.6mW at clock frequency of 2GHz. On an average 80% of total power consumption in CPP multiplier is consumed by clock network and pipeline registers. In the MPP multiplier, total power consumed is less than 50% of total power in CPP multiplier. Also, in MPP multiplier around 55% of total power consumption is in logic which represents useful power.

7) *Lower power supply noise*. In MPP scheme, due to the linear clock distribution approach, there are fewer register stages and they all are not triggered simulatneously. This reduces the current drawn and also the rate (*di/dt*) at which it is drawn. The result is less variation in current drawn by clock network. This means less power supply noise.

8) *CPP Power-performance tradeoff*. The CPP implementation can achieve similar power consumption as MPP scheme only when operated at a much slower speed. The CPP multiplier implementation consumes less power than the MPP implementation only if operated at half the frequency of MPP multiplier.

9) *Tiny chip*. A 4×4-bit CSA multiplier implemented in mesochronous scheme has been fabricated in AMI 0.5μm technology and successfully tested. This chip can operate at a clock frequency of 252MHz. This implementation has a speedup of 1.69 over conventional pipeline implementation and has only 2 pipeline stages and 3 pipeline registers. This multiplier can be pipelined in conventional scheme to operate at

252MHz, however it would have 4 pipeline stages, 5 register stages and a complex clock distribution.

8.1. Contributions of this research

In the design of high performance digital systems, pipelining is an import design concept. Pipelining essentially splits a single large combinational logic block into smaller sequential blocks. This process increases logic utilization, allows parallelism among independent operations. The result is higher performance, higher frequencies of operation. In a Conventional Pipeline (CPP) scheme each stage operates on a single data set or vector at any given time. Data movement between logic (pipeline) stages is synchronized by pipeline registers with the help of globally distributed clock signal. With continuing technology scaling, digital systems are increasing in area and are operating at higher clock frequencies. This is also increasing the complexity of clock distribution and chip power consumption. Novel pipeline architectures are required in future to gain higher performance.

In this research, we proposed a novel Mesochronous Pipeline (MPP) scheme. This scheme simplifies the pipeline architecture, achieves higher performance and higher power savings compared to conventional pipeline scheme.

In this section we shall present the contributions of this research. Some of these points have already been highlighted in the previous section.

1) *Novel high performance pipeline scheme*. Mesochronous pipeline scheme is a novel pipeline scheme to design high performance digital systems. In this scheme a logic system is divided into pipeline stages and clocked such that multiple data sets are present in a pipeline stage at various stages of computation. This means new data is

admitted into a pipeline stage before computation is complete on a previously admitted data set. For this to be possible the logic per stage is considerably more when compared to a conventional pipeline implementation. Fewer pipeline registers are used to synchronize data movement. The clock period in mesochronous pipeline scheme is determined by the pipeline stage with the largest difference between its minimum and maximum propagation delay. In conventional pipeline scheme, stage with maximum propagation delay dictates the minimum clock period achievable. Maximum delay difference is far less than maximum propagation delay, so smaller clock periods (i.e. higher clock frequencies) are possible in the proposed scheme.

2) *Reduced clock distribution network complexity*. In MPP scheme, the clock network design has been modified from the CPP scheme. Instead of a global equipotential clock distribution used in CPP scheme, clock signal travels along with data in MPP scheme. This means clock signal path is parallel to data path and it includes delay elements so that clock signal can travel with data.
3) *Mathematical analysis for the performance of MPP scheme*. Using extensive mathematical analysis, equations have been derived to determine the performance of the proposed MPP scheme. A detailed comparison of MPP scheme's performance with CPP scheme and other pipeline schemes like wave pipeline (WPP) scheme and micropipelines (μPP), proved that MPP can achieve higher performance. A new parameter *Speedup* has been defined to compare performance of MPP and CPP schemes.
4) *Boundaries of proposed scheme*. Issues like design of clock signal path, tackling variations in clock and delay values have also been addressed in detail. Also, detailed

mathematical equations have been provided for the boundaries of proposed scheme The bounds on clock period are

For $N_{(i)}$ value 0 or 1

$$d_{\max(j)} - d_{\min(j)} + t_s + t_h + 2\Delta_{clk} \le T_{clk\_mpp}$$

For $N_{(i)} > 1$

$$\max\left(\frac{d_{\max(i)} + D_R + t_s + \Delta_{clk} - \delta_{(i)}}{N_{(i)}}\right) \le T_{clk\_mpp} \le \min\left(\frac{d_{\min(i)} + D_R - t_h - \Delta_{clk} - \delta_{(i)}}{N_{(i)} - 1}\right).$$

5) *Power savings*. In MPP scheme the clock distribution network is simple and drives smaller load compared to CPP scheme. This helps in saving significant amount of power in clock network. Also, in MPP scheme there are fewer pipeline registers which add to the power savings. In MPP scheme, due to the clock distribution approach, it is possible that not all pipeline registers are triggered at the same time, so the rate at which current is drawn (*di/dt*) from power supply is less. This mean less variation in current and less noise is induced in power supply network.
6) *8×8-bit multiplier simulations*. Extensive simulations have been performed on a Carry-Save Adder (CSA) multiplier to validate the mathematical analysis of the novel pipeline scheme.
7) *Tiny Chip*. A sample 4×4-bit CSA multiplier in meschronous pipeline scheme has been fabricated in AMI 0.5µm, 5.0V technology and successfully tested.

8.2. Future Research

Mesochronous pipelining scheme is a novel pipeline architecture to design high performance and low power digital systems. Through extensive mathematical analysis we

have carefully characterized the proposed architecture and compared its performance with the conventional pipeline scheme and other alternate pipeline schemes like wave pipelining and micropipelines. Carry-Save Adder multiplier has been implemented in conventional and mesochronous pipeline architectures to prove the performance gain and power saving from the proposed scheme. In this section we shall look into some important issues that have to be researched to formalize the mesochronous pipeline (MPP) scheme.

8.2.1. Feedback

Feedback is essential to any pipeline system. Data generated in a stage from a data set has to be sent back to another stage to be used for computation on a different data set. Fig. 8.1 shows the block diagram of a MPP system with feedback loops.

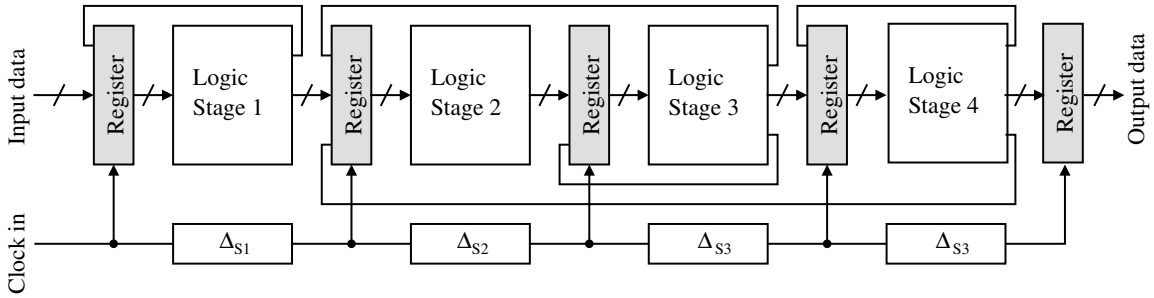


Fig. 8.1. Mesochronous pipeline scheme with feedback loops.

Presence of feedback loops introduces additional constraints on the clock period of the system. To determine clock period (T_{clk_mpp}) with feedback loops, it should be observed that latching at a particular pipeline register occurs exactly one cycle time later. Let S represent the feedback stage distance. To synchronize the forward and feedback paths, a minimum time of ST_{clk_mpp} has to occur between latching the output that needs to be fed back. To determine this time, all the maximum delays and register delays for each stage

that is included in the feedback loop are added. For the feed back from stage *k* to stage *i*, the following inequality must be valid.

$$S_{ki} T_{clk\_mpp} \geq S_{ki} \left(D_R + t_s + \Delta_{clk} \right) + \sum_{s=i}^{k} d_{\max(s)} \qquad (8.1)$$

Rewriting the above question we get the equation for clock period with feed back loops.

$$T_{clk\_mpp} \geq D_R + t_s + \Delta_{clk} + \frac{1}{S_{ki}} \sum_{s=i}^{k} d_{\max(s)} \qquad (8.2)$$

This limit on clock period has to be calculated for all the feed back loops. Let us consider feedback loop S_{ki} to be the longest feedback loop, and let stage *j* have the largest delay difference in a MPP system. Then clock period is given by

$$T_{clk\_mpp} = \max\left(\left(d_{\max(j)} - d_{\min(j)} + t_s + t_h + 2\Delta_{clk} \right), \left(D_R + t_s + \Delta_{clk} + \frac{1}{S_{ki}} \sum_{s=i}^{k} d_{\max(s)} \right) \right) \qquad (8.3)$$

There are several issues associated with feedback. Synchronization of both forward and feedback paths needs be addressed to guarantee system operation. Synchronization can be achieved using pipeline registers, insertion of delay circuits, and signal bypassing. A detailed study is necessary on synchronization mechanism. A delay variation in the feedback loop stages may affect performance and/or functionality of the pipelined system. An in-depth analysis of feedback placement and constraints would lead to a pipelined system that can tolerate delay variations in the feedback loop. Also, impact of the number of stages that can be included in the feedback loop must be studied, as increase in number of stages in the feedback loop makes synchronization of data difficult.

8.2.2. Testing

Testing circuits is a critical factor in all VLSI designs. It is important to incorporate testing methods at the architectural level itself. These test methods should be able to exhaustively test the system for faults. In the current high speed designs, it is extremely important to perform *at-speed* test to verify timing of hardware. Built-in Self-Test (BIST) is a well know Design for Testability (DFT) approach that can be used for *at-speed* testing. In the case of mesochronous pipeline scheme, it is extremely important to identify delay faults (variations in delay values), at its design speed. However, in the mesochronous pipeline architecture, *controllability* and *observability* are partially lost due to large pipeline stages and fewer registers. Observability of a particular logic node is the degree to which the node can be observed at the outputs of an integrated circuit [15]. Controllability of a circuit node is the ease of setting the node to logic 0 or logic 1 [15]. To solve this problem, shadow registers with scan chain capability, can be incorporated in pipeline stages. This concept is illustrated in Fig. 8.2.

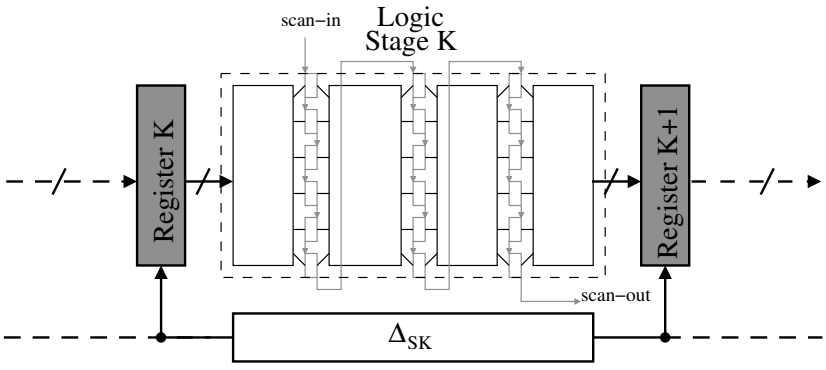


Fig. 8.2. Shadow registers and scan-based testing.

Use of shadow registers in logic stages introduces additional delays and additional power consumption. Also, any additional implications of these shadow registers have to be carefully studied.

In Chapter 4, tackling delay variations has been explained in detail. A possible solution to tackle delay variations involved use of variable delay elements in the clock signal path. This variable delay element has control inputs which have to be generated by the test circuitry. It is important for the test method to have the ability to detect delay faults and generate necessary control signal for the variable delay elements.

8.2.3. CAD tools

In current VLSI designs, it is necessary to have short design time. To facilitate ease of design, it is important to have CAD tools. Mesochronous pipeline scheme is a novel architecture, so CAD tools have to be modified or new tools have to be designed to support this scheme. Since MPP scheme is similar to conventional pipeline scheme, it should be easy to modify CAD tools to support it.

A detailed analysis of the proposed Mesochronous pipeline scheme and various simulations performed on MPP multiplier implementation, prove the feasibility of this scheme. Architectural improvements are required in future high speed designs and mesochronous pipeline offers a viable scheme to this need. We also discussed the future research potential in MPP scheme. The work from this research and some future work will help formalize the novel mesochronous pipeline scheme.

Bibliography

[1] J. L. Hennessy and D. A. Patterson, *Computer Architecture, A Quantitative Approach*, 3rd ed., San Francisco, CA: Morgan Kaufmann, 2002.

[2] P. J. Restle, and A. Deutsch, "Designing the best clock distribution network," *Symp. VLSI Circuits*, pp. 2 – 5, June 1998.

[3] S. Tam, R. D. Limaye, U. N. Desai, "Clock Generation and Distribution for the 130-nm Itanium 2 Processor with 6-MB On-Die L3 Cache," *IEEE J. Solid-State Circuits*, vol. 39, no. 4, pp. 636 – 642, April 2004.

[4] Y. I. Ismail, and E. G. Friedman, "Effects of Inductance on Propagation Delay and Repeater Insertion in VLSI Circuits," *IEEE Trans. VLSI Syst.*, vol. 8, no. 2, pp. 195 – 206, April 2000.

[5] G. Oklobdzija *et. al.*, *Digital System Clocking*, Wiley-Interscience, 2002.

[6] F. Klass, *et. al.*, "A New Family of Semidynamic and Dynamic Flip-Flops with Embedded Logic for High-Performance Processors," *IEEE J. Solid-State Circuits*, vol. 34, no. 5, pp. 712 – 716, May 1999.

[7] D. E. Duarte, N. Vijaykrishnan, and M. J. Irwin, "A clock power models to evaluate impact of architectural and technology optimizations," *IEEE Trans. on VLSI Syst.*, vol. 10, no. 6, pp. 844 – 855, Dec. 2002.

[8] C. T. Gray, W. Liu, and R. K. Cavin, "Timing constraints for wave-pipelined systems," *IEEE Trans. on Computer-Aided Design*, vol. 13, no. 8, pp. 987 – 1004, Aug. 1994.

[9] W. P. Burleson, M. Ciesielski, F. Klass, and W. Liu, "Wave-pipelining: A tutorial and research survey," *IEEE Trans. on VLSI Syst.*, vol. 6, no. 3, pp. 464 – 474, Sep. 1998.

[10] I. E. Sutherland, "Micropipelines," *Communications of the ACM*, vol.32 no.6, pp.720-738, June 1989.

[11] E. G. Friedman, "Clock distribution networks in synchronous digital integrated circuits," *Proc. IEEE*, vol. 89, no. 5, pp. 665 – 692, May 2001.

[12] N. R. Mahapatra, S. V. Garimella, and A. Tareen, "An empirical and analytical comparison of delay elements and a new delay element design, " Proc. IEEE Comp. Society workshop on VLSI, pp. 81-86, April 2000.

[13] M. Maymandi-Nejad and M. Sachdev, "A digitally programmable delay element: Design and analysis," IEEE Trans. on VLSI Syst., vol. 11, no. 5, pp. 871 – 878, Oct. 2003.

[14] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits*, 2nd ed., Upper Saddle River: NJ, Prentice Hall, 2002.

[15] N. H. E. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, 3rd ed., Addison Wesley, 2005.

[16] V. Stojanovic and V. G. Oklobdzija, "Comparative analysis of master-slave latches and flip-flops for high-performance and low-power systems," IEEE J. Solid-State Circuits, vol. 34, no. 4, pp. 536 – 548, April 1999.

[17] V. Stojanovic, V. G. Oklobdzija, "FLIP-FLOP," US Patent No. 6,232,810, May 15, 2001.

[18] S. B. Tatapudi and J. G. Delgado-Frias, "Designing pipelined systems with a clock period approaching pipeline register delay," *48th IEEE Intl. Midwest Symp. Circuits Syst.*, Aug. 2005.

[19] S. B. Tatapudi and J. G. Delgado-Frias, "A Mesochronous Pipelining Scheme for High-Performance Digital Systems," *IEEE Trans. Circuits Syst. I*, vol. 53, no. 5, May 2006.

[20] J. Nyathi and J. G. Delgado-Frias, "Hybrid-wave pipelined network router," IEEE *Trans. on Circuits Syst. I*, vol.49, no. 12, pp. 1764 – 1772, Dec. 2003.

[21] S. Zhao, K. Roy, and C-K Koh, "Estimation of Inductive and Resistive Switching Noise on Power Supply Network in Deep Sub-micron CMOS circuits," *Proc. Intl. Conf. Comp. Design*, pp. 65 – 74, Sept. 2000.

[22] W. H. Lee, S. Pant, and D. Blaauw, "Analysis and Reduction of On-Chip Inductance Effects in Power Supply Grid," *Proc. 5th Intl. Symp. Quality Electronic Design*, pp. 131-136, 2004.

[23] S. B. Tatapudi and J. G. Delgado-Frias, “A Mesochronous Pipeline Scheme for High Performance Low Power Digital Systems,” *IEEE Intl. Symp. Circuits Syst.*, May 2006.

Appendix A

Publications

Following is a list of papers published in reputed Journals and Conference proceedings, based on the research during M.S. and Ph.D. programs.

A.1. Journal

1. S. B. Tatapudi and J. G. Delgado-Frias, "A Mesochronous Pipelining Scheme for High-Performance Digital Systems," *IEEE Trans. Circuits Syst. I*, vol. 53, no. 5, May 2006.

2. J. G. Delgado-Frias, J. Nyathi, S. B. Tatapudi, "Decoupled Dynamic Ternary Content Addressable Memories," *IEEE Transactions on Circuits and Systems-I*, vol. 52, no. 10, Oct. 2005, pp. 2139-2147.

3. S. B. Tatapudi and J. G. Delgado-Frias, "A High Performance Mesochronous Pipeline Scheme with Reduced Power and di/dt," *IEEE Trans. Circuits Syst. I*, (submitted).

A.2. Conference

1. S. B. Tatapudi and J. G. Delgado-Frias, "A Mesochronous Pipeline Scheme for High Performance Low Power Digital Systems," *IEEE Intl. Symp. Circuits Syst.*, May 2006.

2. S. B. Tatapudi and J. G. Delgado-Frias, "A pipelined multiplier using a hybrid-wave pipelining scheme," *Proceedings IEEE Computer Society Annual Symp. VLSI*, pp. 282 – 283, May 2005.

3. S. B. Tatapudi and J. G. Delgado-Frias, "Designing pipelined systems with a clock period approaching pipeline register delay," *48th IEEE Intl. Midwest Symp. Circuits Syst.*, pp.871 – 874, August 7-10, 2005.

4. S. B. Tatapudi and J. G. Delgado-Frias, "A pipelined multiplier using a hybrid-wave pipelining scheme," *Proceedings 2005 Intl. Conf. On Computer Design*, pp. 191 – 197, June 2005.

5. J. Nyathi, V. Beiu, S. B. Tatapudi, D. J. Betowski, **"**A charge recycling differential noise immune perceptron," *Proc. IEEE Intl. Joint Conf. Neural Networks*, vol. 3, pp. 1995-2000, July 2004.

6. S. B. Tatapudi and J. G. Delgado-Frias, "A VLSI Self-Compacting Buffer for Priority Queue Scheduling," *3rd IASTED International Conference on Circuits, Signals and Systems (CSS)*, pp. 310 – 315, May 2003.

7. S. B. Tatapudi and V. Beiu, "Split-Precharge Differential Noise-Immune Threshold Logic Gate (SPD-NTL)," *International Work-conference on Artificial Neural Networks (IWANN)*, June 2003.

8. S. B. Tatapudi and J. G. Delgado-Frias, "A Reduced Clock Delay Approach for High Performance Mesochronous Pipeline," *49th IEEE Intl. Midwest Symp. Circuits Syst.*, August 6-9, 2006. (submitted)