A SYSTEM LEVEL PERFORMANCE MODEL FOR ASYNCHRONOUS MICROPIPELINE CIRCUITS

Bengt Oelmann[1,2], and Hannu Tenhunen[1]
1. Royal Institute of Technology, Department of Electronics, Electronic System Design Laboratory
Electrum 209, S-164 40 Kista, Sweden. E-mail: bengt|hannu|@ele.kth.se
2. Mitthögskolan, Sundsvall, Sweden.

ABSTRACT

In this paper we present how an asynchronous system, using micropipelines, can be modelled in a system level performance model. We have introduced structures for pipeline stages and feedback structures. The model has been used in order to find out at what complexity a micropipeline implementation can out-perform a synchronous one. We have also used it for examining if micropipelines can be used as an alternative to clock-gating as a method for saving power. Results from these simulations are presented and compared to measurements on a complex asynchronous circuit.

1. INTRODUCTION

It is often claimed that asynchronous circuits have properties suitable for low-power or/and high speed operation. However, so far it has only been demonstrated in a very few practical design cases [2]. A comparison on local function block level is easy to make by actually designing and simulate or measure on the circuit. If we want to study the behaviour of an asynchronous design technique at system level, the number of possible structures that could be interesting to examine are many. Also the design time gets unreasonable long for this type of experiments.
With system level performance models it is possible to study how changes in technology, circuits, or micro-architecture influence the overall system performance. Since the system is modelled in a very general way, it can be used for study tendencies. This type of modelling was first introduced in the SUSPENS model [1]. This was mainly used for predicting the performance of microprocessors in future technologies. It was also used for modelling larger systems containing several ASICs (Application Specific Integrated Circuits) mounted on MCMs (Multi Chip Modules) and PCBs (Printed Circuit Boards). Liu and Svensson use the same approach in [4], but they have refined the models of power consumption for different structures within a digital CMOS ASIC. It has been used for estimating how much power different parts in an ASIC consume.
In this paper we introduce models for micropipeline circuits and structures into the system level performance model. This includes models on gate level and register-transfer level. In order to make comparisons between synchronous circuits and micropipeline circuits, we also have functions for translating from a synchronous to an asynchronous circuit. When modules in a circuit are conditionally activated, clock gating is a technique to reduce the power consumption. If the number of places clock gating are getting large, it becomes complicated to handle. Here asynchronous circuits are regarded to have an advantage thanks to their event-driven nature. We are modelling circuits divided into modules running at different speeds, so that we can compare if any differences between the techniques exist.

2. MICROPIPELINE MODELS

Micropipeline circuits are built from a limited number of building blocks. How these building blocks are implemented is well established [6]. There is, however, no unified design methodology how to build a more complex design. Compared to synchronous circuits they are often irregular, and the structure is different for different function units. By limiting the number of structures, the design can be handled in a more uniform way. This approach gives us a structure that is similar to the structure of synchronous designs. And it has been successfully used in a large scale design [5]. In addition it gives us a possibility to compare the different techniques.

2.1. Gate level models

Throughout this paper we consider a standard cell approach. This means that we use static CMOS logic. We also assume a system optimized for speed. In this chapter we describe how logic gates and latches are modelled. Logic gates and control circuits are modelled as *buffered static and-gate*, see figure 1. It consists of two parts, first the logic function and then the driver that is optimized for the actual loading condition.
We refer to [1,4] for more detailed information on how estimation of power and speed of buffered static logic can be made.

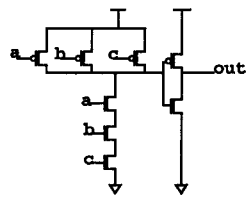

Figure 1. Buffered static logic (AND-gate)

We have chosen to use a latch based on Sutherland's *capture-pass* latch [6]. It is described by three different components. It consists of storage elements, control circuits, and driving circuits, see figure 2.

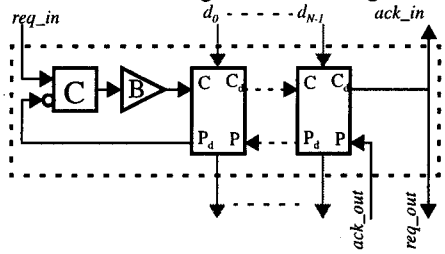

Figure 2. Control circuit (C-element), driver (B), and storage elements (n_b=N-1).

We are interested in getting an expression for the time it takes to latch a new datum in the latch. The latch delay in the control path is the time from signal *req_in* to *req_out*. This delay is a sum of the delay in the C-element and the delay from driving a number of one-bit storage elements. The delay from the C-element is modelled as the logic gate. The driver is driving a load composed by wire segments that connect each of the inputs of the storage elements. The interconnection wires are modelled as distributed RC lines.

There are three source of power dissipation in the latch. First we have the C-element ($P_{control}$), the power consumption of which is the same as for a buffered logic gate. The power in the driving network is,

$$P_{drv} = \frac{1}{2}\left[4 \times \frac{1}{2}(1 + k_{drv})\, n_b (C_l + C_{int} l_{av})\right] f V_{dd}^2$$

where k_{drv} is the driver capacitance [3], n_b is average bit-width, C_{int} is interconnection capacitance per unit length, l_{av} is the average wire length, f is operation frequency, and V_{dd} is power supply voltage. The factor 4 is the number of control signals that are driven, but they are only switching once every second cycle (factor 1/2). The control lines are loaded with the input capacitance C_l of the storage elements. The storage elements are basically built from tri-state inverters (switches). These are modelled as

$$P_{storage} = \frac{1}{2}\left(21 C_{tr} + 4 C_d^{switch} + C_d^{inv}\right) n_b \alpha f V_{dd}^2$$

where C_d indicates the output capacitances and α is the switching activity. The total power dissipation for one latch is

$$P_{latch} = P_{control} + P_{drv} + P_{storage}$$

2.2. Register transfer level models

A complete system can be described by pipeline stages and feedback structures (like state-machines) and legal interconnections of these.

2.2.1 Pipeline stage

A single pipeline stage, shown in figure 3 below, consists of a *normally transparent latch*, control path, and combinational logic.

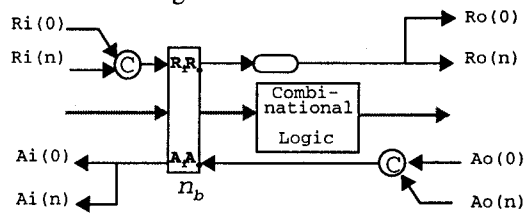

Figure 3. Micropipeline stage

A pipeline stage may have several sources of data. The computation cannot proceed until data from all the different sources are valid. The C-element on the input side will join all the input sources (*Ri(0)* to *Ri(n)*). On the output side there is a similar situation. The receivers of the output data must acknowledge the data after capturing them. The number of modules connected to the inputs and the outputs will affect the cycle time. A parameter - f_{gm}, indicating average fan-in fan-out of the modules, is introduced.

The bundled data convention requires a matched delay in the control path. We assume that, for all systems considered, the performance is limited by delay in the combinational logic. Another assumption on the control path made is that the size of the logic (number of logic gates) in the control path is proportional to the logic depth in the combinational logic.

Compared to synchronous designs, an additional parameter - n_b, indicating the average bus-width of the data, is needed. There are two reasons for this. First, n_b determines delay and power dissipation in the local driving circuitry in the latches. Second, it is used to determine the size of the control logic. The size of the system is given to the model in the number of transistors for a synchronous system. It is then translated into a micropipeline system, and the size of the asynchronous control logic is calculated here.

2.2.2 Feedback structures

The feedback structure describes state-machines. Externally it looks like a pipeline stage. The primary data inputs are latched in a *normally transparent latch*. The current state is stored on the outputs of the state-variable register. The next state is computed when a new current state is valid and new data have been latched from the primary inputs. These two paths are joined in the C-element. The state-variable register is an asynchronous master-slave register and is built from a *normally transparent latch* and a *normally opaque latch*. In order to describe the feedback structure an additional parameter must be added. We must separate the latches (for pipelining) from the

registers (storing state-variables). We introduce a parameter describing how large part is used for state-variables, n_{fb}.

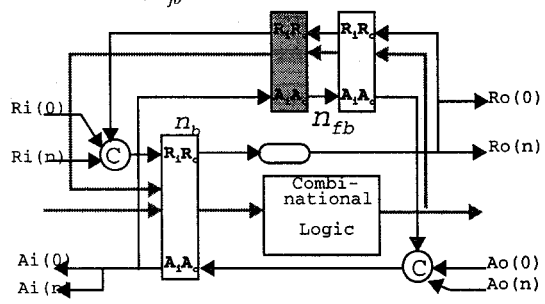

Figure 4. Feedback structure

In a synchronous circuit where edge-trigged flip-flops are used it is no need to make a distinction between pipeline registers and registers in feed-backs. The feedback registers are composed of two latches.

2.3. System level models

In order to model a system where different parts operate at different speeds, we must divide it into a number of modules. The operational speed within a module is constant, but different modules are running at different speeds. Since we do not want to introduce any structural information about the system, we have described the system in the following way. The system is divided into a number of equal sized modules. These are physically organized in a regular way. The smallest size of a module is one single pipeline stage. The largest possible number of individually clocked modules in the system are:

$$n = \frac{D_c^2}{n_b (f_{ld} + 1) d_g^2}$$

where D_c is the chip die size and f_{ld} is the logic depth, and d_g is the gate pitch.

The extreme situation is when each of the pipeline stages is conditionally clocked, which means that the clock signal is gated when a specific condition is satisfied. In this case we have *fine grained* clock-gating. Of course, the clock gating can be made with coarser granularity. To be able to express the degree of granularity we introduce a parameter g:

$$g = lg_2(d_m) - 1$$

where d_m is the module die size in units of minimum module size. In the description above we have used terms normally associated with a synchronous system. The same approach is also applied for the micropipeline system.

Even if the modules are operating with different average speeds, it is sufficient to use only maximum speed f_{max} and average speed f_{av} where

$$f_{av} = \frac{1}{n} \sum_{i=1}^{n} f_{av(i)}$$

and $f_{av(i)}$ is the average speed in module i.

With the assumptions we made, f_{max} is of importance in the synchronous case. In figure 5a a clock distribution net organized as an H-clock tree is shown with processing elements indicated as gray boxes. The upper left part of the clock tree consists of processing elements with fine granularity. It means that the fast clock (f_{max}) must be distributed to each processing element. If we have coarser granularity, the clock distribution net that carries f_{max} does only have to reach the boundary of the module. From that point the clock can be conditionally gated, and a clock signal with a lower average frequency ($f_{av(i)}$) is distributed.

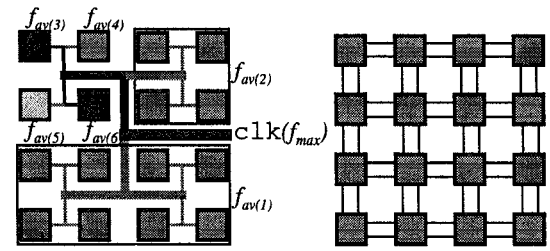

Figure 5. a) Clock distribution as an H-clock tree, b) asynchronous model.

In the micropipeline case we assume that each one of the modules controls its data rate locally, and the architecture is organized in a way that communication between the modules is exclusively local, see figure 5b. In order to also reduce the data flow, the outputs of each module must pass an asynchronous master-slave register. This will introduce extra circuitry for each of the modules. We estimate the number of inputs and outputs for a module by using Rent's rule [1]

$$n_o = \frac{1}{2} \times \frac{1}{2^{g-1}} \frac{N_g^{chip}}{N_g^{module}} K_p \left(2^{g-1} N_g^{module} \right)^{\beta}$$

where N_g is the number of gates, K_p is a proportionality constant, β is Rent's constant.

The output data rate from each of the modules is assumed to be different for all modules. Rent's rule gives the number of IO:s, so the factor 1/2 gives the number of outputs, n_o, only. For each output, a one bit asynchronous master-slave register must be added.

2.4 Verification of the model

The system model is controlled by a large number of parameters on different abstraction levels. Some parameters are direct input values, and others are calculated. At all of these levels there are descriptions on how different things affect the physical characteristics, like wire length, area, speed, and power. The model was verified through comparison of the simulation results to measurements on a micropipelined DSP for direct-sequence spread-spectrum (DS-SS) receiver [5], see table below.

	Estimated asynch.	Measured asynch.	Estimated synch.
Speed [Ms/s]@5V	75	48	115
Area (core)[mm^2]	20	21	15
Power@48Ms/s,3V [mW]	250	210	200

3. Simulation Results

The circuits in this work are modelled as standard cell designs in a 0.8µm CMOS technology. The synchronous circuits are using static D-Flip-Flop of master-slave type. The average logic depth is 10 gates. The average bit width is 10 bits and 25% of the registers is used in feedback structures. First we look the upper limit for the clock frequency and power consumption for circuits of different sizes, see figure 6a-b. The transistor count on the x-axis in figures 6a-b indicates the values for the synchronous circuit. Due to circuit overhead (approximately 35%) the asynchronous circuit actually contains more transistors. A graph for conversion from synchronous transistor count to asynchronous is given in figure 6c. Next we look at how the granularity of clock gating and the size of asynchronous modules operating at different speeds affect the system performance. Figures 6e-f show power consumption and circuit overhead as a function of granularity for a 2M transistor circuit. The clock-gating (and asynchronous control) reduces the average clock frequency to 50% of max. clock frequency in this experiment.

4. Conclusions

In this paper we have presented how micropipeline systems can be modelled in a system level performance model. The estimated results from it have been compared to measured data, and the estimation errors are reasonable low. The results indicate that micropipeline circuits will only be more efficient at extremely high complexity. Also we could conclude that micropipelines do have a problem when data rate must be controlled for every small module. We will then get an increase in area overhead, which will lead to increase in power consumption. One of the benefits of asynchronous circuits often claimed, is that they only run at the actual needed speed. For micropipelines, that is true for the asynchronous control path, but not for the data path.

5. References

[1]H.B. Bakoglu, Circuits, Interconnections and Packaging for VLSI,Addison-Wesley, ISBN 0-1-201-06008-6

[2]K. Berkel, et al., A Fully-Asynchronous Low-Power Error Corrector for the DCC Player, Proc. ISSCC, February 1994, pp.88-89

[3]N. Hedenstierna and K.O. Jeppson, CMOS circuit speed and buffer optimization, IEEE Trans. on Comp., vol. 35, 1986, pp. 880-895.

[4]D. Liu, and C. Svensson, Power Estimation in CMOS VLSI Chips, IEEE J.of Solid-State Circuits, vol. 29, no. 6, June 1994.

[5]B. Oelmann, H. Martijn, and H. Tenhunen, VLSI Implementation of a DS-CDMA receiver using asynchronous design techniques, 6th IEEE Int. Symp. on Personal, Indoor and Mobile Comm. 27-29 Sept. 1995.

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

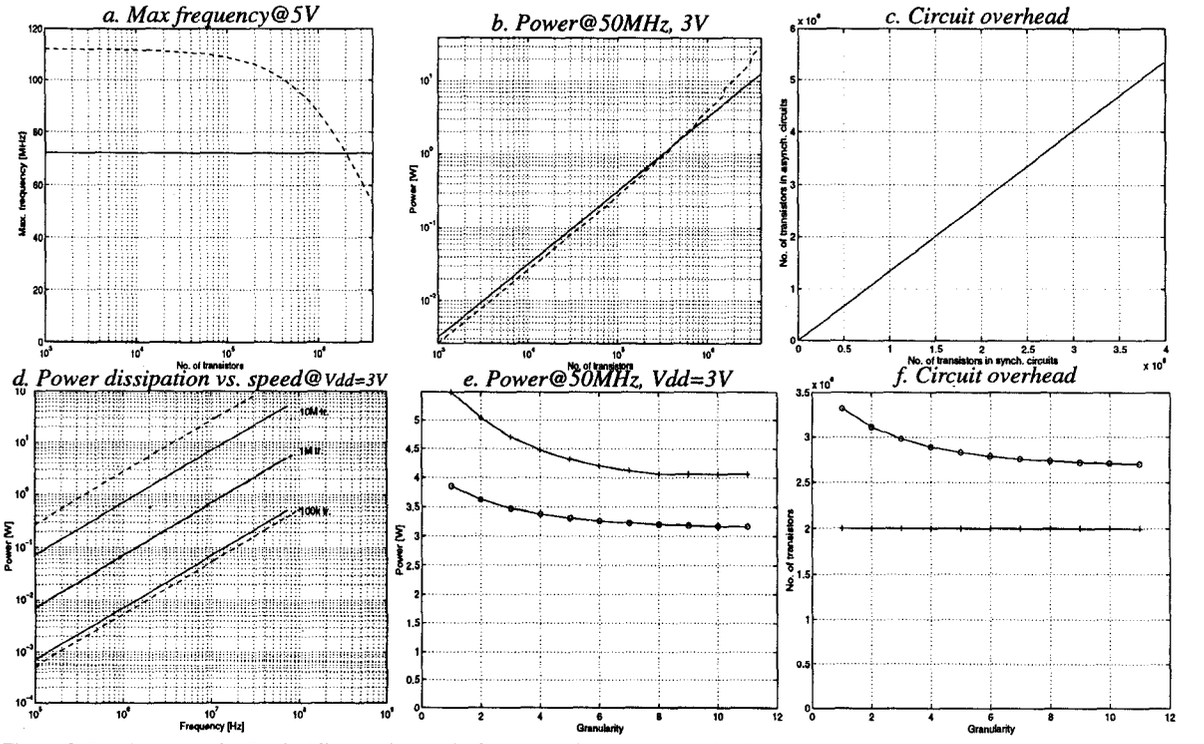

Figure 6. Simulation results. Dashed lines and +-marks for the synchronous case, solid lines and o-marks for the micropipeline case.