International ASIC/SOC Conference, pp. 203-207, September 2002, Rochester, New York

HYPIPE: A NEW APPROACH FOR HIGH SPEED CIRCUIT DESIGN

Jos B. Sulistyo and Dong Sam Ha
VTVT (Virginia Tech VLSI for Telecommunications) Lab
Department of Electrical and Computer Engineering
Virginia Tech, Blacksburg, VA 24061

Abstract—Wave pipelining improves the throughput of a circuit by exploiting the delays of combinational elements, rather than register clocks, for synchronization. We proposed a new design approach for high speed circuits which combines the conventional register-based pipelining with wave-pipelining. Our approach called HyPipe aims to take the advantage of both pipelining methods. We applied our method to 1-bit and 2-bit adder cells, which can be used as building blocks for larger size adders and multipliers. Our experimental results show that the 1-bit adder achieves the throughput of 2.4 billion additions/second and the 2-bit adder achieves 2.2 billion additions/second for TSMC 0.25 μm technology. Furthermore, they have potential for even higher throughputs provided registers are able to operate faster.

I. INTRODUCTION

Pipelining is widely employed for circuits to increase the throughput. The throughput of a pipelined circuit is determined by the worst delay of a pipeline stage, and register clocks are responsible for synchronization. Another method for improving throughput is *wave-pipelining*, also called *maximum-rate pipelining* [1]. Wave-pipelining exploits the delays of combinational circuits to process multiple data simultaneously. To distinguish the two pipelining schemes, we call the former method *register-pipelining* and the latter one wave-pipelining.

Wang et al. used the register-pipelining to design an 8b × 8b multiplier, which reaches the speed of 630 MHz in a 0.6 μm technology through bit-level pipelining, namely inserting a register stage after every adder cell [6]. Ghosh and Nandy [2] attained the same speed using a 0.8 μm technology using wave pipelining. However, wave pipelining requires delay balancing, and it is difficult for CMOS circuits because of the inherent input-pattern dependent delay variation [1]. Ghosh and Nandy suggested use of complementary logic (requiring both logic values x and /x) based on pass transistors (or possibly transmission gates), which incur reasonably balanced delays for logic 1 and logic 0, and hence it may attain better delay balance [2]. However, the pass-transistor approach increases transistor counts significantly compared with other logic styles such as fully complementary or dynamic logic. Furthermore, pass transistors degrade rise/fall times of signals, and hence may not be appropriate for high speed.

We proposed a new design approach for high-speed circuits based on pipelining. Our method called *HyPipe* combines the two pipelining methods, register pipelining and wave pipelining. HyPipe aims to take the advantages of both pipelining methods to improve the throughput of a circuit, while relaxing the requirement for delay balancing necessary for wave pipelining. We successfully applied our approach for 1-bit and 2-bit adders, which can be used as building blocks for larger adders or multipliers. Our experimental results show that HyPipe double the speed of 1-bit and 2-bit adder cells, which achieve the throughput of 2.4 billion additions/second and 2.2 billion additions/second for 1-bit adders and 2-bit adders, respectively, for TSMC 0.25 μm processing under 2.5 V supply voltage.

The paper is organized as follows. Section II provides background for wave pipelining, and Section III discusses the proposed approach for pipelining, HyPipe, and present adder designs based on HyPipe. Section IV contains experimental results and observations, and Section V concludes the paper.

II. BACKGROUND

In this section, we briefly describe terms and conditions for wave-pipelined circuits and discuss the bound of the operating frequency of wave-pipelined circuits. For details, an excellent tutorial on wave pipelining is available in [1].

The general structure of wave pipelining is shown in Fig. 1. It consists of input and output registers and a combinational logic block (CLB). The following terms are defined for wave-pipelined circuits.

S_{MIN}, S_{MAX}	minimum and maximum setup times of registers (i.e., flip-flops)
R_{MIN}, R_{MAX}	minimum and maximum clock-to-output delays of registers
C_{MIN}, C_{MAX}	minimum and maximum propagation delays of the combinational logic block

W_{MIN}, W_{MAX} minimum and maximum propagation delays from the input register to the output register
T_{RISE}, T_{FALL} rise and fall times of a signal line
T_{CK} clock period of a wave-pipelined circuit

It is important to note that a minimum (maximum) delay in the above should be obtained considering all possible pairs of input transitions, since a delay often depends on the sequence of input patterns applied.

Since the hold time often does not contribute to the path delay from the input register to the output register, we ignore the hold time of registers. Then, the following identities hold:

$$W_{MIN} = R_{MIN} + C_{MIN} + S_{MIN}$$
$$W_{MAX} = R_{MAX} + C_{MAX} + S_{MAX}$$

Note that the propagation delay of a register is the sum of its setup time and the clock-to-output delay [4]. The minimum allowable clock period for a non-wave-pipelined circuit should be W_{MAX}. In contrast, the clock period for a wave-pipelined circuit is limited by the difference between the slowest and the fastest delays as shown below.

$$T_{CK} \geq W_{MAX} - W_{MIN} \quad (1)$$

The above clock period determines the maximal operating frequency at which a wave-pipelined circuit can operate.

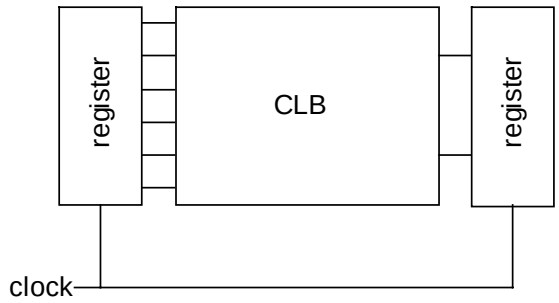

Fig. 1. The Structure of a Wave-Pipelined Circuit

The *wave number N* of a wave-pipelined circuit is defined as the number of clock cycles needed for a signal to propagate through the combinational logic block before latched by the output register. It represents the degree of wave-pipelining. Two conditions should be satisfied for the clock period T_{CK} for a given wave number N. A data applied at clock k should be available at NT_{CK} clocks later for sampling at the output register as shown in Fig. 2. Further, the next earliest wave of the data applied at clock (k+1) should have not arrived yet at the output. The two conditions are expressed as

$$W_{MAX} \leq NT_{CK}$$
$$W_{MIN} \geq (N\text{-}1)T_{CK}$$

Therefore,

$$W_{MAX}/N \leq T_{CK} \leq W_{MIN}/(N\text{-}1) \quad (2)$$

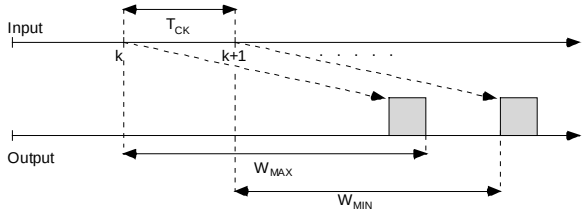

Fig. 2. Time Window for Sampling of Data

In addition, the clock period is bounded by rise/fall times of gate input/output signals, gate delays, and the delays of the input/output registers:

$$T_{CK} \geq \max(T_{RISE}, T_{FALL})_{all\ nodes} \quad (3)$$
$$T_{CK} \geq R_{MAX} + S_{MAX} \quad (4)$$
$$T_{CK} \geq \text{the largest gate delay of the circuit} \quad (5)$$

The rise and fall time of all nodes in (3) include input signals of a wave-pipelined circuit. The four constraints of (2)-(5) on the clock frequency make the operable frequency of a wave-pipelined circuit much smaller than the theoretical maximal frequency given in (1).

III. PROPOSED METHOD

Although wave pipelining offers potential for high speed, delay imbalances accumulated over signal paths of a large combinational logic block reduce the actual operable frequency significantly. We propose to partition a combinational logic block into smaller blocks and to apply wave pipelining to each smaller block, while the entire combinational block operates in a register-pipelined manner as shown in Fig. 3.

The finer granularity for the proposed method decreases the minimum and maximum delays of a subblock, W_{MIN} and W_{MAX} of each pipeline stage, resulting in a significant increase of the maximum operable clock speed. Additionally, a smaller building block makes it easier to balance delays as the paths are shorter. The cost for the proposed method compared with wave pipelining is a larger number of registers and a longer latency. Note that a longer latency for the proposed method may not be

severe, as the proposed method can operate at a higher speed.

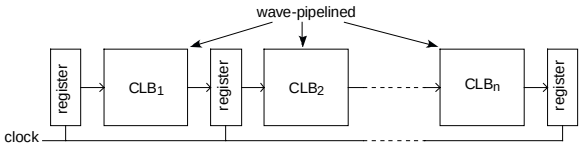

Fig. 3. Block Diagram of the Proposed Pipelining Method HyPipe

HyPipe is particularly attractive for a combinational logic block which is composed of a few identical building blocks such as ripple-carry adders and array multipliers, since it requires well crafted design of a few building blocks. We applied HyPipe to ripple carry adders. The goal of our design is to achieve the highest possible operating speed in CMOS. The adder design is explained in the following.

A. 1-bit and 2-bit Adders

We considered several logic candidates, primitive gates versus complex gates, use of pass transistors, and static versus dynamic logic. Complementary static CMOS was chosen for two reasons. First, unlike a transmission-gate, it does not degrade the slew rate of signals as it propagates through a circuit. This reason also rules out the use of complex gates. Second, static designs dissipate less power than dynamic ones due to lack of precharge operation. The number of fan-ins for our complementary static gates is limited to three, as the delay variation of a static gate increases with the increased the number of inputs.

The resultant circuits for 1-bit 2-bit adders are given in Fig. 4 through Fig. 6. The circuits are designed to attain a good delay balance. For the 1-bit adder, the b input for the sum generator is delayed using two buffers in series to balance the input delays of the XNOR gate. Further, both the sum and carry-out outputs have inverter drivers to equalize their drive strengths as well, so that their delays will remain almost the same even under a moderately heavy load. The two buffers in series for the carry-out generator also function to pad the delays. For the 2-bit adder, a1 and b1 inputs are delayed, so that all the three inputs for the upper-bit adder (c1, a1, b1) arrive almost simultaneously. Here the delay of /c1 is used for balancing the delay of c2 by temporally tiling the 2-bit following the idea illustrated in [3].

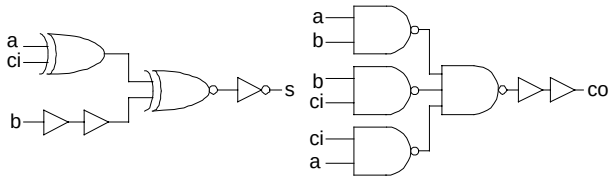

Fig. 4. The Proposed 1-Bit Adder

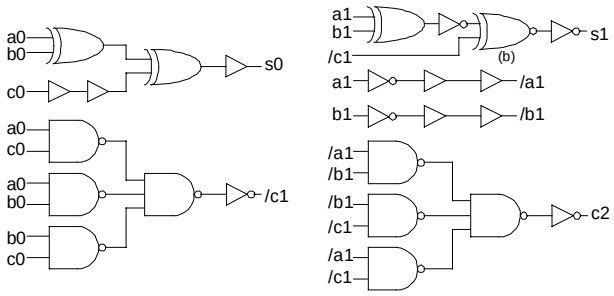

Fig. 5. The Proposed 2-Bit Adder

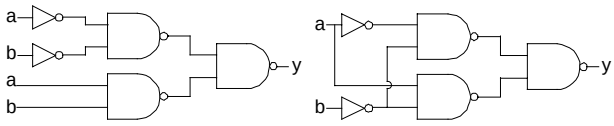

Fig. 6. XNOR (left) and XOR (right) Gates Used

B. Registers

We consider a couple of design styles to implement the registers, namely a C^2MOS flip-flop and a simple dynamic flip-flop shown in Fig. 7. The simple dynamic flip-flop was for two reasons: first, it is faster, and second, it has a stronger output drive (due to the inverter output stage) and it is approximately equally strong for both rising and falling transitions. Therefore, both the D-to-Q latency and the rise/fall times remain short under moderately heavy loading conditions. In contrast, a C^2MOS flip-flop is slower, and its output also tends to display longer rise/fall times, since both the p-tree and the n-tree consist of two series transistors. As an illustration, a simple dynamic flip-flop considered by us attains (R_{MAX} + S_{MAX}) of 281 ps, compared with 324 ps for a C^2MOS flip-flop.

We laid out both 1-bit and 2-bit adders including registers targeting TSMC 0.25 μm technology with the supply voltage of 2.5 V. Figure 8 shows the layout of the 1-bit adder with output flip-flops. Its size is 29.16 × 35.64 μm^2.

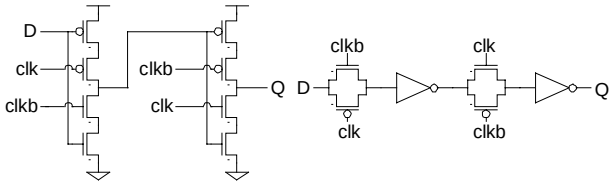

Fig. 7. C^2MOS (left) and Simple Dynamic (right) Flip-flops

IV. EXPERIMENTAL RESULTS

The netlists of the proposed 1-bit and 2-bit adder cells were extracted and simulated using HSPICE. The first experiment is to estimate the delay spreads of our adders, which are the most important parameters affecting the performance. To find out the delay spreads, we simulated the 1-bit adder with all the 56 possible input transitions. For the 2-bit adder, 168 patterns were considered. Specifically, 56 patterns to determine the fastest and slowest delays of s0 and c1, another 56 patterns to determine the delay variations of s1 and c2 under the application of the fastest input pattern to c1, and the last 56 patterns for s1 and c2 delays under the slowest input pattern to c1. The experimental results are shown in Table I.

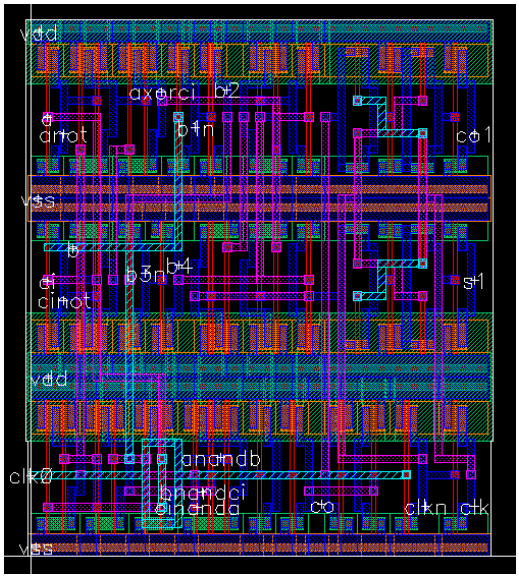

Fig. 8. Layout of a 1-Bit Adder

Table I: Delay Spreads of the Proposed 1-bit and 2-bit Adders (Unit: ps)

Output	1-bit adder		2-bit adder	
	Min/Max delay	Spread	Min/Max Delay	Spread
s0	322-407	85	384 - 488	104
c1	291-409	118	190 - 291	101
s1	-	-	363 - 546	183
c2	-	-	374 - 533	159

The table shows that the delay spreads of the 2-bit adder is wider than that of the 1-bit adder. This is mainly because the delay variation in node /c1 (refer to Fig. 5) causes further delay variations for c2 and s1. This justifies our decision to limit the width of the building block to just two bits. If a wider block (e.g. three bits) is used, the accumulation of delay spreads along the paths of the carry may cause the MSB of the output to display a large delay variation.

Next, we estimated the performance of HyPipe with one pipeline stage (i.e., input/out registers and an adder bit) of our 1-bit and 2-bit adders. In order to compare the performance of our adder designs, we also considered a modified Ghosh/Nandy adder [2] and Shams/Bayoumi adder [5]. Ghosh/Nandy adder cell is aimed for high performance using wave pipelining, while Shams/Bayoumi adder is mainly for low power dissipation. .

Ghosh/Nandy adder was modified in our work by using transmission gates instead of n-pass transistors proposed in [2]. This enables the rail-to-rail swing of internal signals, and therefore allowing the use of smaller p transistors for inverters. For Shams/Bayoumi adders, we used larger than minimal size transistors, but the same proportions as suggested in [2] were used. Shams/Bayoumi adder cannot be used for wave pipelining for two reasons. First, it has a large delay spread. Second, its output has long rise/fall times.

We generated the netlist of the Ghosh/Nandy and Shams/Bayoumi adder cells manually rather than from layouts. So the performance of the two adders would be slightly degraded when parasitics are included. The smallest transistors are of the same size in all the four adder cells for fair comparison. Finally, it should be noted that both Ghosh/Nandy and Shams/Bayoumi adder cells are one bit adders.

The experimental results are given in Table II. To make the comparison meaningful, the power dissipation shown in the table was measured at 200 MHz for all the designs. For the HyPipe adders, we estimate

conservatively the center frequencies under N=2 as the operating speed, which are 2.4 GHz and 2.2 GHz for the 1-bit and the 2-bit adders, respectively.

Table II. Performance and Power Dissipation of Several Adder Cells

	Delay spreads (ns)	Power per bit (μW)	Speed (GHz)	
			N=1	N=2
Prop. 1-bit adder	116	97	1.43	1.96 - 2.82
Prop. 2-bit adder	183	90	1.21	1.85 - 2.5
Ghosh/Nandy	142	138	1.35	1.23 - 1.88
Shams/Bayoumi	219	41	1.40	N/A

From the experimental results, we made the following observations. First, the proposed 1-bit adder gives the best delay balance. Its delay spread is lower than that of the 2-bit adder as expected. The Shams/Bayoumi adder does not have a tight delay balance due to uneven logic depths. The delay balance of the proposed 1-bit adder is even better than the modified Ghosh/Nandy adder. It appears that the delay imbalance of Ghosh/Nandy adder is due to the imbalanced loads faced by its first-level gates.

Second, Shams/Bayoumi adder attains low power dissipation through the use of transmission gates, which result in a low transistor count and small short-circuit current. However, it is relatively slow.

Third, the maximum delay of the proposed 2-bit adder is longer and its throughput is lower than those of our 1-bit adder. However, since it processes two bits in the same cycle, the latency of the adder is half that of the 1-bit adder.

Finally, the flip-flops are the hindrance to operation at higher wave numbers for our adders. It is possible to further equalize delays, so that C_{MIN} is closer to C_{MAX}. A wave number of 3 is attainable if the flip-flop can operate at $T_{CK} \le R_{MAX} + S_{MAX}$ = 273 ps, or 3.66 GHz. The fastest flip-flop which we have designed so far, however, can operate only up to 3.56 GHz under the loads encountered.

V. CONCLUSION

We proposed a new design approach for high speed circuits which combines the conventional register-based pipelining with wave-pipelining. Our approach called HyPipe aims to take the advantage of both pipelining methods. To demonstrate the effectiveness and practicality of our design method, we applied our method to 1-bit and 2-bit adder cells, which can be used as building blocks for larger size adders and multipliers. Our experimental results show that HyPipe doubles the speed of 1-bit and 2-bit adder cells, which achieve the throughput of 2.4 billion and 2.2 billion additions/second, respectively, for TSMC 0.25 μm processing technology under the 2.5 V supply voltage. Furthermore, they have potential for even higher throughputs provided registers are able to operate faster.

REFERENCES

[1] W. Burleson, M. Ciesielski, W. Liu, and F. Klass, "Wave-Pipelining: A Tutorial and Research Survey", *IEEE Trans. VLSI Systems*, Sept. 1998, pp. 464-474.

[2] D. Ghosh and S. K. Nandy., "Design and Realization of High-Performance Wave-Pipelined 8×8b Multiplier in CMOS Technology", *IEEE Trans. VLSI Systems*, March 1995, pp. 36-48

[3] S. Mahant-Shetti, P. Balsara, and C. Lemonds, "High Performance Low Power Array Multiplier Using Temporal Tiling", *IEEE Trans. VLSI Systems*, Mar. 1999, pp. 121-124.

[4] V. Stojanović and V. G. Oklobdžija, "Comparative Analysis of Master-slave Latches and Flip-flops for High-performance and Low-power Systems", *IEEE J. Solid-State Circuits*, Apr. 1999, pp. 536-548.

[5] A. M. Shams and M. A. Bayoumi, "A Novel High-Performance CMOS 1-Bit Full-Adder Cell", *IEEE Trans. Circuits and Systems – I: Analog and Digital Signal Processing*, May 2000, pp. 478-481.

[6] J.-S. Wang, P.-H. Yang, and D. Sheng, "Design of a 3-V 300-MHz Low-Power 8-b × 8-b Pipelined Multiplier Using Pulse-Triggered TSPC Flip-Flops", *IEEE J. Solid-State Circuits*, Apr. 2000, pp. 583-592.