Low-Power High-Level Synthesis Using Latches

Wooseung Yang, In-Cheol Park and Chong-Min Kyung
CHiPS, Department of EECS, KAIST
373-1, Kusong-dong, Yusong-gu, Taejon 305-701, Korea
Tel:+82-42-866-0843 Fax:+82-42-866-0702
e-mail:woosee@duo.kaist.ac.kr

Abstract – High-level synthesis using latches has many merits in power, area and even in speed. But latches cannot be read and written at the same time and usually requires two-phase non-overlapping clock that is unpleasant choice for short-term design. In this paper we propose a storage allocation method that makes it possible to use latches as storage elements in single clocking scheme. The proposed method modifies the lifetime of variables slightly so that it can be applied to any high-level synthesis systems with small modification. The experimental results show 39 ~ 65% reduction in power consumption within almost same area compared to the conventional power management scheme using clock gating.

I. Introduction

High-level synthesis (HLS) has been intensively researched for the productivity of VLSI design, and many high-level synthesis systems have been introduced. It has been shown that HLS tools are especially good choice for specific application domains such as digital signal processing (DSP) circuits or control-dominated circuits. These application areas usually require low power-consumption for its mobile characteristics. There were many approaches in high-level synthesis to reduce the power consumption of the generated circuit. In [5][6] functional unit allocation and storage unit allocation algorithms to minimize the switching activity of the nets are introduced. Various low power techniques are applied iteratively based on the switched capacitance calculation in [7]. In [3] inputs of functional units are suitably controlled by changing storage allocation so that functional units can execute only necessary operations. All these schemes assumes flip-flops as storage elements for several reasons. First, using flip-flops is easy to think and to apply algorithms since flip-flops can be read and written at the same time. Second, use of latches usually requires two phase clocking scheme which designers are not pond of.

But power consumption of latches is about one third smaller than that of flip-flops and area is also about one third smaller than flip-flop[8]. In addition, the smaller clock capacitance in latch can reduce further power consumption in clock tree. Since much of power consumption in recent chips are due to the clock tree, using latches in high level synthesis of data path will be very beneficial in power point of view. There were some approaches to use latches instead of flip-flops to get these merits. One example in [1] showed up to 50% power saving by using latches. But using multiple-clocking scheme is very unpleasant choice for chip designers. In [2], storage elements implemented with flip-flops are replaced with latches if the input/output behavior of the circuit is not affected. It showed good results for control-dominated circuits, but in data-dominated circuits not many of the storage elements meets the conditions for substitution and the change in the waveform of internal signals may cause increase in functional unit power consumption.

With these in mind we will propose a storage allocation method considering the latch-based storage element implementation with single clocking scheme. This paper is organized as follows: Section II describes previous approaches in more detail and Section III gives motivating example showing the possibility of power reduction using latches. Section IV described the binding algorithm used in our approach. Some excremental results are showed and analyzed in Section V and conclusion is made in Section VI.

II. Previous Works

In [1], circuits are partitioned into n sub-networks so that the storage elements of each partition can be activated at n distinct time step and each partition is clocked by n non-overlapping clock of frequency of f/n, where f is the operating frequency of the circuit before partitioning. With this architecture, area can be increased in some degree since registers and functional units cannot be shared by different partitions, but the power consumption can be reduced since the sum of power consumption in each partition is usually smaller than that of the original circuit for the reduced load capacitance.

One more rationale for the power reduction in that architecture is the reduced operating frequency. Reduced frequency means smaller signal transition. But this is true only when the power management scheme is not applied to the original circuits i.e. all storage elements are clocked at every clock cycle. If power management scheme is applied to the original circuits, the storage elements change their values only when new values hvae to be updated by using gated clock or input multiplexers. The minimal required number of value change is not determined by the architecture but by the algorithm and a simple power management scheme guarantees that minimal required transition in the storage elements. So the transition number of the storage element in the partitioned circuit is not smaller than that of the original circuits.

In spite of this, the simulation results show that the partitioned circuits consume smaller power than that of the original circuits with conventional power management scheme. It is in one hand due to the effect of the reduced load capacitance mentioned above and in the other hand due to the reduction of power consumption in storage elements. Since the storage elements in each partition are guaranteed not to change their values in consequent clocks, all the storage elements can be implemented with latches without affecting the behavior of the circuit.

Similar approach to reduce power consumption in storage elements using latch is found in [2]. Some flip-flops in synthesized circuits are replaced with latches after high-level synthesis. Since latches are transparent during the clock is low (or high in positive level sensitive latches), the waveforms of the outputs of the storage elements can be changed after replacement. So this approach tried to find out the flip-flops that do not change the behavior of the primary output ports when replaced with latches. Two basic conditions for the flip-flops to be safely replaced with latches are that they should not be read and written at the same clock cycle and that they should not be connected to the primary output directly or through combinational logic circuits. The first is because the storage elements can make a combinational loop during transparent operation when they are implemented with latches and the second is because the primary output can be advanced half cycle earlier after replacement than before replacement. Sometimes it is difficult to find out the flip-flops that meet those conditions. In data-dominated applications many storage elements have self-loops, i.e. they can be read and written at the same time thus violate the condition. In control-dominated circuits, an additional condition on control data flow graph (CDFG) limits the substitution chance. Another demerit of this scheme is that unnecessary signal propagation during the latches are transparent can cause addition power consumption in functional units. In control-dominated circuits the power consumption in combinational logic generating next input values of storage elements are not so great, but in data dominated circuits, this power consumption cannot be ignored.

In contrast with the above approaches that tries to reduce the power consumption in storage units, there are other power management schemes to minimize the power consumption in functional units with the sacrifice in storage unit area. Among those, perfect power management scheme in [3] shows most noticeable results with negligible overhead. The key idea of perfect power management is to allocate storage units so that all functional units connected to each storage elements generate useful outputs. It is done by extending the lifetime of each variable until all next time input variables of the corresponding functional unit are available. This method can perfectly eliminate the spurious operation of all functional units but its lifetime modification causes increase in the number of registers. The area increased by the added registers is not so critical, but the power consumption in storage elements is comparable to that of the combinational parts in many circuits. So it gives more chance to reduce power consumption.

With all these in mind, we proposed an allocation scheme that allows all the storage elements to be implemented in latches while minimizing the spurious operations in functional units so that we can achieve low power architecture.

III. Motivating Example

Data flow graph in Fig. 1 will be used as an example in this section. Seven variables and three operations are there. Assume that scheduling and functional unit allocation is already done. Three operations will in executed in consequent control steps (**S1**, **S2**, **S3**) and the number in each node represents the number of functional units to execute the operations.

Fig. 2 shows the lifetime of each variable, storage allocation result of left edge algorithm and timing diagram of the final circuit, for original and two modified version of lifetimes. In Fig. 2 (a), only two storage elements are required according to the left edge algorithm. All the storage elements are implemented with positive edge triggered flip-flops and none can be replaced with latches; for **R1** which stores **a**, **h**, **i**, **k**, the destination variable and source variable of the operation are assigned to same storage element (**a** and **h** for **FU1** and **h** and **i** for **FU2** are assigned to **R1**) and for **R2** which stores **b**, **c**, **j**, the storage elements is written at the consequent control steps—replacing the flip-flop with a latch in this case causes the output value of the functional unit change before that value is transferred to the storage element.

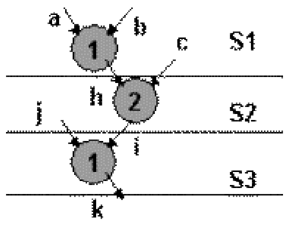

Fig. 1. Example data flow graph: variable names are attached along edges, the kind of operations are represented with numbers inside the nodes and control steps are notified on the right side.

In Fig. 2(b) lifetimes are modified a little bit to be able to use latches as storage elements. The death time of each variable is extended one control step more to hold the value one more cycle after the cycle it was last used. The timing diagram shows the signal status when all storage elements are implemented with negative level sensitive latches. Assuming that all the primary inputs and control signals are available from the negative edge of the clock signal, we can assert that all values will be available before the next negative edge of the clock causing no glitch at the positive edge of the clock.

Shaded region of the timing diagram represents spurious operation of functional units. In Fig. 2(a), functional units evaluate 8 times in total and 5 of them are unnecessary evaluation caused by the sharing non-relevant variables in one register.
Many of them are eliminated in Fig. 2(b) since sharing is avoided by extending lifetimes of variables. Further extending the lifetime of variable h and c, we can remove all spurious functional unit operation with the sacrifice of one more storage element as in Fig. 2(c).

As we can see in the example, with some modification in the lifetimes of variables, it is possible to synthesis circuits so that all storage elements can be implemented with latches and spurious operation of the functional units can be minimized. In next section, we will describe how to modify the lifetime to incorporate above features.

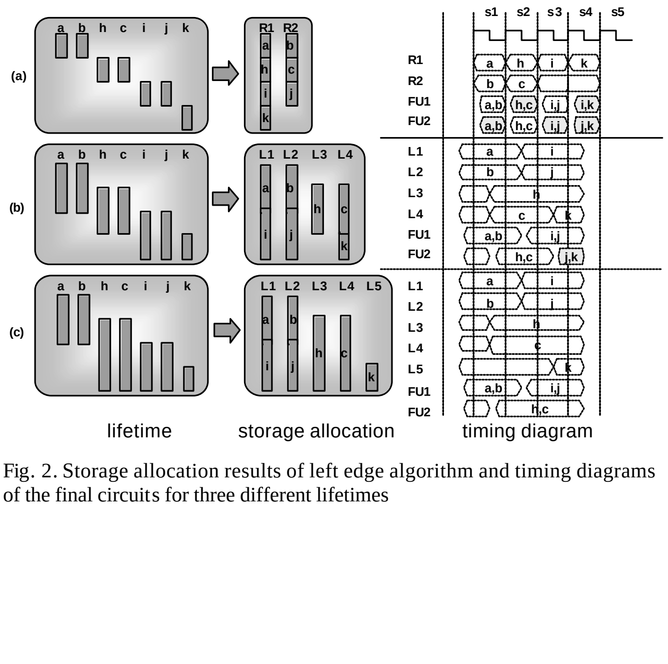

Fig. 2. Storage allocation results of left edge algorithm and timing diagrams of the final circuits for three different lifetimes

IV. PROPOSED METHOD

The condition for a storage element to be safely implemented with a latch is that it should not change the output value until one more extra cycle after its death time. Two contrasting situations are depicted in Fig. 3. Fig. 3(a) shows the case when the two variables (**a** and **b**) are allocated to one register so that the register changes its value as soon as the death time of the assigned variable arrives. In the first row labeled **Reg**, the output waveform of a register implemented in a flip-flop is shown. At the first cycle the register holds variable **a**, and the functional unit **FU1** begins evaluation from the positive edge of the clock and after a mean time output is stabilized. At the next cycle, the **FU1** output is transferred to another register **x,** and at the same time the register holds new variable **b** since the death time of the variable **a** arrived. In this case the flip-flop cannot be replaced with a latch since the latch may change its output before the relevant functional units transfer their results to the other storage elements.

Fig. 3(b) shows the case when the two variables are allocated to one register so that the register changes its value only after one extra cycle since the death time of the assigned variable. In this case the storage element satisfies the replacement condition.

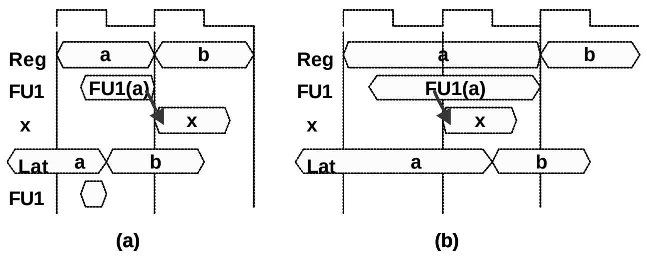

Fig. 3. Condition for safe replacement of a flip-flop with a latch

This scheme can be applied to any high-level synthesis system with small modification in lifetime of variables with the following equation. V means the set of all variables in source description and $t_d(v_i)$ and $t_d'(v_i)$ means death time and modified death time of the variable v_i.

$$t_d'(v_i) = t_d(v_i) + 1, \text{ for ? } v_i \text{ ? } V \qquad (1)$$

Every flip-flop can be replaced with latch after the lifetime modification. The replacement results in the reduction of overall circuit area and power consumption. However, for the circuit in which the functional unit power consumption is comparable to the consumption in storage units, latch replacement can cause additional power consumption in functional units since the latch output changes value twice in one clock cycle; once at the positive edge of the clock and once during the clock is low. This increase in functional unit power consumption can be kept minimum with the help of technique in [3]. The key idea is to allocate storage units so that all functional units connected to each storage elements generate useful outputs. By extending the lifetime of each variable further to meet the following condition, the spurious operation in functional unit can be minimized.

$$t_d'(v_i) = \max\{\, t_d(v_i) + 1, \max_{v_j \text{ ? } NEXT(v_i)}\{\, t_b(v_j)\,\}\,\} \qquad (2)$$

In the expression, operator NEXT (v_i) means a set of variables that was used as the next operands of the functional unit processing v_i and $t_b(v_j)$ means birth time of variable v_j. The experimental results will be mentioned in the next section.

V. EXPERIMENTS & RESULTS

We experimented our scheme for two HLSynth92 benchmarks[4], DIFFEQ and ELLIPF. DIFFEQ is a differential equation solver example and ELLIPF is an elliptical filter example. These benchmarks are given also with suitable test vectors. Fig. 4 shows the procedure used in our experiment. In the first step, input behavior code described in simple VHDL-like code is read in and data flow graph (DFG) is constructed. In the next steps, given the resource constraints, scheduling is done with ASAP algorithm and the functional units are allocated with the appearance order in the source description. Since the proposed scheme is only related with the storage allocation step of high-level synthesis, we didn't make much effort on scheduling and functional unit allocation step. In the forth and the fifth steps, the lifetime of each variable is analyzed and modified with the equations in the previous section. With the modified lifetime minimum number of storage units are allocated using left-edge algorithm. After all these, RTL description in Verilog HDL is generated using latches, multiplexers and functional units such as adder, multiplier as building blocks.

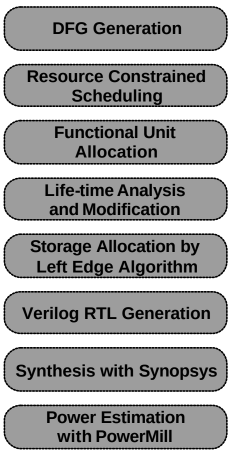
DFG Generation
Resource Constrained Scheduling
Functional Unit Allocation
Life-time Analysis and Modification
Storage Allocation by Left Edge Algorithm
Verilog RTL Generation
Synthesis with Synopsys
Power Estimation with PowerMill

Fig. 4. The experiment procedure

Generated RTL code is synthesized using Synopsys Design Compiler for 0.65um standard cell library. After gate level simulation for the synthesized design, the power consumption is estimated using Epic PowerMill. The test vectors supplied with the benchmarks are used for the gate level simulation and power estimation.

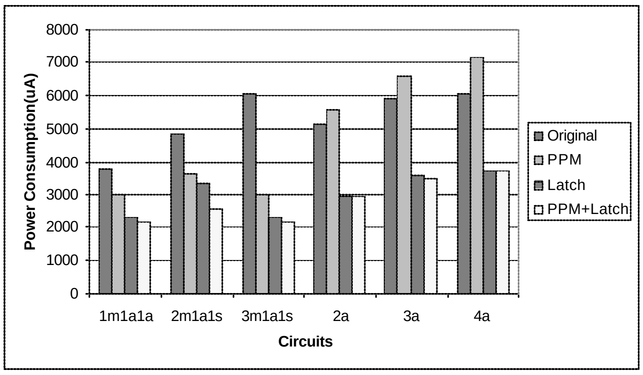

Fig. 5. Power consumption: The first three circuits are DIFFEQ and the next three circuits are ELLIP in HLSynth92 benchmark

Fig. 5 shows the power simulation results of the two HLSynth92 benchmark circuits for different resource constraints. The first three circuits are DIFFEQ example synthesized with different number of multipliers; one, two and three multipliers each. The last three circuits are ELLIP example with two, three and four adders. The bar graphs tagged 'Original' represents the circuit with simple clock gating power management scheme. The next bars tagged 'PPM' represents the circuit with perfect power management scheme applied. All the storage elements in Original and PPM are implemented with flip-flops. In the third and forth bars tagged 'Latch' and 'PPM+Latch', the storage elements are implemented with latches by applying Equation (1) and Equation (2) respectively. In DIFFEQ example, the three low power schemes gradually decrease power consumption. But in ELLIP example where storage power consumption is dominant as we can see in Table 1, PPM scheme increases power consumption by using more storage elements than the original circuit. In any cases circuit implementation with latches consumes less power compared to the circuit with other power management schemes. In Fig. 6, the areas of the synthesized circuits including the estimated interconnection area are shown. Since the number of storage elements is increases when we use latches, the overall area is not so much reduces. When storage elements are more dominant than the functional units as in ELLIP example, the area may be increased on the contrary.

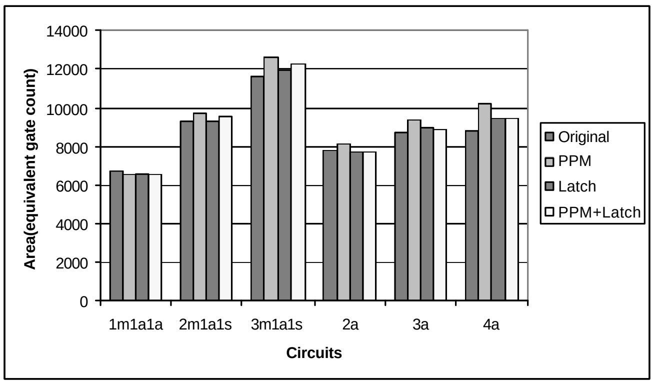

Fig. 6. Area of synthesized circuits

VI. Conclusion

In this paper, we proposed a storage allocation method to enable the storage element being implemented with latch. In general, latch implementation of the circuits requires two phase clocking scheme to correctly pass variables to the next storage elements in consequent operation of the two storage elements. In the proposed scheme, all the storage elements can be implemented with latches using single clock scheme, since the lifetime of variables are modified to guarantee that the variables are valid until it is safely passed to the next storage elements. Since the lifetimes of the variables are slightly modified, the proposed scheme can be applied to any high-level synthesis systems.

According to our experiments, the proposed scheme is equally helpful either when the storage unit power is dominant or when the functional unit power is dominant, contrary to the perfect power management that is useful only when functional unit power is dominant. The power consumption in proposed low power scheme using latch is reduced to 39 ~ 65% of the power consumption in the circuit with simple power management by clock gating.

Table 1

AREA AND POWER CONSUMPTION OF BENCHMARK CIRCUITS FOR EACH LOW POWER SCHEME

Circuits	Resources	LowPowerScheme	Area(GC)	Power(uA)	StoragePower(uA)
DIFFEQ	1m1a1s	Original	6710	3760	1823
		PPM	6557	2984	1714
		Latch	6621	2278	985
		PPM+Latch	6536	2139	971
	2m1a1s	Original	9288	4817	1996
		PPM	9759	3619	2182
		Latch	9328	3361	1235
		PPM+Latch	9559	2561	1170
	3m1a1s	Original	11645	6060	1927
		PPM	12622	2984	1714
		Latch	11973	2278	985
		PPM+Latch	12283	2139	971
ELLIP	2adder	Original	7793	5159	4207
		PPM	8125	5570	4626
		Latch	7710	2972	2154
		PPM+Latch	7710	2925	2129
	3adder	Original	8772	5905	4675
		PPM	9384	6584	5451
		Latch	9004	3594	2478
		PPM+Latch	8882	3491	2460
	4adder	Original	8849	6063	4673
		PPM	10267	7151	6022
		Latch	9448	3710	2482
		PPM+Latch	9440	3702	2541

References

[1] A Multiple Clocking Scheme for Low-Power RTL Design,Christos A. Papachristou, Mehrdad Nourani, and Mark Spining, IEEE TVLSI, VOL. 7, NO. 2, JUNE 1999
[2] Storage Optimization by Replacing Some Flip-Flops with Latches, Tsung-Yi Wu, Youn-Long Lin, EDAC-96
[3] Power Management in High-Level Synthesis, Ganesh Lakshminarayana, Anand Raghunathan, Niraj K. Jha, Sujit Dey, TVLSI March 99
[4] "Benchmarks for the 6th International Workshop on High-Level Synthesis," ftp://ics.uci.edu, 1992
[5] Module Assignment for Low Power, Jui-Ming Chang and Massoud Pedram, EDAC-96
[6] Register Allocation and Binding for Low Power, Jui-Ming Chang and Massoud Pedram, DAC-95
[7] SCALP: An Iterative-Improvement-Based Low-Power Data Path Synthesis System, Anand Raghunathan and Niraj K. Jha, TCAD November 1997
[8] 0.65um Cell Based Library Databook, LG Semicon, 1995