Automatic Synthesis of Clock Gating Logic with Controlled Netlist Perturbation

Aaron P. Hurst
University of California, Berkeley
Berkeley, CA

ABSTRACT
Clock gating is the insertion of combinational logic along the clock path to prevent the unnecessary switching of registers and reduce dynamic power consumption. The conditions under which the transition of a register may be safely blocked can either be explicitly specified by the designer or detected automatically. We introduce a new method for automatically synthesizing these conditions in a way that minimizes netlist perturbation and is both timing- and physical-aware. Our automatic method is also scalable, utilizing simulation and satisfiability tests and necessitating no symbolic representation. On a set of benchmarks, our technique successfully reduces the dynamic clock power by 14.5% on average. Furthermore, we demonstrate how to apply a straightforward logic simplification to utilize resulting don't cares and reduce the logic by 7.0% on average.

Categories and Subject Descriptors
B.6.3 [**Logic Design**]: Design Aids

General Terms
Algorithms, Design.

Keywords
Clock Gating, Low Power, Dynamic Power, Logic Optimization.

I. INTRODUCTION

The dynamic switching of the clock network typically accounts for 30-40% of the total power consumption of a modern design, and with the proliferation of low-power requirements and thermal limitations, minimizing this portion of the power consumption is imperative. One of the most effective and widely adopted techniques is clock gating, whereby the clock signal is selectively blocked for registers in the design that are inactive or do not otherwise need to be updated, thus reducing the average capacitance that must be switched per cycle. The most common approach is to manually identify architectural components that can be deactivated, but in this work, we focus on automatic techniques that can be applied to netlist-level circuits.

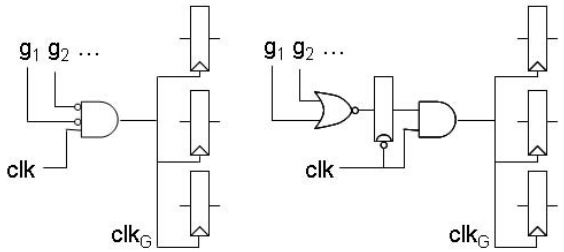
DAC 2008, June 8–13, 2008, Anaheim, California, USA

The condition under which a register's clock can be stopped may be derived from its current and next state functions, and previous methods have considered such direct computation and synthesis [4][7]. However, the limits of such symbolic functional manipulation are typically reached far below the size of many moderately sized designs, especially when multiple registers must be gated simultaneously. Furthermore, once a gating function has been selected it must be synthesized, and in general, this requires additional logic. Even if the physical design is not disrupted by the additional area and routing, its dynamic power consumption eats away from the power-saving benefits that it seeks to provide. The coverage of the function can be safely pruned to save implementation cost, but determining a good balance between coverage and cost is a difficult synthesis problem.

We propose an approach that addresses the dual problems of gating condition selection and synthesis by constructing these functions out of signals in the existing logic network. While the maximal gating condition is rarely present in the network, there are often many signals whose functions (or complements) are strictly contained within it; these provide a set of sufficient conditions to determine if the register will not switch and can therefore be safely gated. While this is less flexible than the synthesis of an arbitrary function, the result is still quite good and, importantly, scalable to large designs with very predictable results.

The identification of the potential components of a gating condition is accomplished through the combination of simulation and SAT testing. The resulting candidates are then collected and grouped, and a subset of these is selected to inhibit the greatest degree of register switching with the smallest number of clock gates. A byproduct of using simulation in the generation of the clock gating logic is the probabilistic information it provides about the sequential behavior of the circuit; an accurate model of the power savings can be used as the optimization objective.

Because the next state of a register is not dependent upon its input when the gating condition is active, clock gating also introduces observability don't cares into its next state function. When the gating condition is a disjunction of functions already present in the logic network, a straightforward structural simplification can be applied to minimize the logic. We demonstrate a moderate reduction in the total network size after clock gating.



(i) (ii)

Figure 1. Two common implementations of clock gating for positive edge-triggered registers. The gated clock signals clk_G are inhibited when $G = g_1 \vee g_2$ is true. If G is monotonic or stable during either half-period of the clock, the gating can be implemented with only a logic gate. Otherwise, circuit (ii) is a glitch-safe version. Several standard cell libraries provide such a combined cell.

II. CLOCK GATING

Clock gating involves the insertion of conditions on the propagation of a clock to one or more registers in the design. By limiting any unnecessary switching, the dynamic power required to charge and discharge the capacitive load of the register inputs is reduced. The condition under which a clock transition is inhibited is known as the *gating condition*, *clock disable*, or *activation function* [9]. In general, the condition may a sequential function of variables from previous time frames, though we restrict the problem to the combinational version.

Let x be the set of external inputs and current state variables. x_R is the current state of register R, and $F_R(x)$ is its next state function. To maintain the functional correctness of the circuit, each register's gating condition $G_R(x)$ must only be active when the register does not change state. This functional *correctness condition* is described by Equation 1.

$$G_R(x) \Rightarrow \overline{F_R(x) \oplus x_R} \tag{1}$$

If $G_R(x) = \overline{F_R(x) \oplus x_R}$, then it is the unique *maximal gating condition*. Because the timing requirements of the clock gate typically necessitate that G_R is available earlier than F_R, it is desirable to find an *incomplete gating condition* that can be generated early in the clock cycle with maximal coverage and minimal implementation cost. This problem was studied in [4] and [9].

The local dynamic clock power saved by gating the clock at R with function G_R is proportional to the probability that G_R is true, $\mathcal{P}(G_R)$, and the register input capacitance c_R (which includes internally switched capacitance). The dynamic clock power consumed to generate each gated clock signal clk_G is proportional to the additional capacitance seen by the clock network, c_G. The resulting power objective is expressed by Equation 2. Additional dynamic power may be dissipated in the logic network by increasing the fan-out loads, but we restrict our power consideration to the clock network. In practice, the overall dynamic power of the logic network is also decreased through the optimizations described in Section 4. Additional power is also saved by moving any common clock drivers behind the clock gates.

$$\sum_{\forall R} c_R \mathcal{P}(G_R) - \sum_{\forall (\text{unique } G)} c_G \tag{2}$$

Typical values of c_G and c_R imply that the gated clock signals must be shared amongst multiple registers, for each of which the corresponding gating condition must be valid. To be effective, clock gating synthesis problem must span multiple registers at a time.

III. ALGORITHM

We model a circuit to be clock gated as a hierarchical hypergraph whose nodes may be either single-bit registers or single-output combinational logic nodes. If there are multiple clock domains, each group of registers must be gated separately. We define a literal to be either one of these node outputs or its complement.

The goal of the gating algorithm is to maximize the power savings by finding an incomplete gating condition G_R for each register R, such that G_R is the disjunction of up to M literals, as described in Equation 3.

$$G_R(x) = \bigvee_{i=1..M} g_i^R(x) \tag{3}$$

The steps of our technique are summarized in Algorithm 1 and described in the following sections in sequential order.

A. Candidate Identification

The first step consists of extracting a set of *candidate literals* for the $g_i^R(x)$ terms of the gating signal G_R, for each register R. All literals could be initially be considered as candidates for each register, but it is useful and necessary to immediately narrow the set by removing ones that violate either timing, physical, or structural constraints.

i) Timing constraints. The added delay of the clock gating logic and the stricter timing constraints of the clock signal dictates that the clock gating condition be available earlier than latest next state function. It is therefore only necessary to select from amongst signals with early enough arrival times to meet these requirements.

ii) Physical constraints. It is undesirable to route gated clock signals over large distances, and constraints between the proximity of the candidate gating signals and the gated registers are necessary to prevent difficult or unroutable connections. This type of constraint provides a worst-case linear bound on the number of pairs that must be considered.

iii) Structural constraints. The candidate literals can be restricted to those whose structural support are partially common to the next state function or include the register output. This is implied by Equation 1.

B. Candidate Pruning

Because the sum of a set of terms satisfies the correctness condition (Equation 1) only if each term satisfies it, a literal is only kept as a candidate if it is not inconsistent with this condition. Simulation is applied in several passes to prune the set of candidate literal/register pairs. The pruning passes are quite fast and effective, and if any literal is found to violate the correctness condition, it is immediately removed from consideration.

C. Candidate Proving

Once the set of candidates has been reduced with pruning to literal/register pairs that are reasonably likely to be legal, these are proved to satisfy the correctness condition using a satisfiability solver in incremental mode. Because the portion of the problem describing the circuit functionality does not change, learned clauses are kept to speed up future runs. The test structure is depicted in Figure 2. If the output is satisfiable, then there exists an input that violates the correctness condition; otherwise, *g(x)* in now known to be a valid gating condition for register *R*.

D. Covering

Besides generating counterexamples to the correctness condition for illegal candidate literals, simulation provides a probabilistic estimate of the number of unnecessary clock transitions that each legal literal g will block, $\mathcal{P}(g)$. This provides more accurate information than assuming that the size of the Boolean ON-set of a gating condition correlates to its actual ON-probability (e.g. [4]).

However, assuming that it is not possible to keep the full set of simulation values for every net, we lack any information about the correlations between these probabilities: the gating probability of any disjunction of these signals is unknown. To overcome this limitation, we group them into useful candidate sets $g_{1..M}$; the details of this heuristic are not described here.

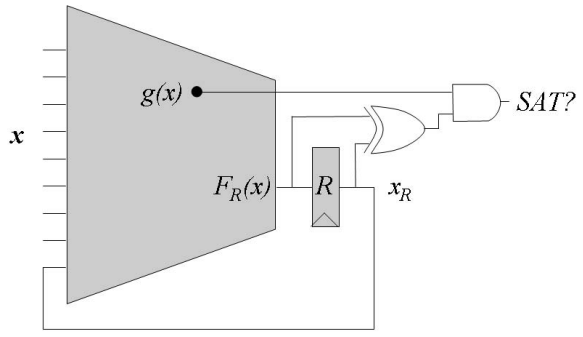


Figure 2. A candidate literal *g*(*x*) is being tested for validity as a gating cover component for register *R*. If the output is satisfiable, the candidate is discarded. This is first checked with simulation and then conclusively proven with SAT.

The circuit is simulated again—with actual simulation traces, if available—and probabilistic information is collected about the candidate sets, thereby capturing the correlated probabilities. The correlation *between* sets is not pertinent: each register can only be switched by a single gated clock, generated by the one cover that is chosen for it. This restriction can be relaxed by amending our technique to employ hierarchical clock gating.

The problem now reduces to the weighted maximum set cover problem, where the weight of each element set is exactly its net contribution to Equation 2, the total dynamic power. If an insufficient number of registers or clock transitions are gated, the net weight of an element may be negative; these will never be selected. The maximum set cover problem is NP-hard, but there exist good heuristics. The problem is also less difficult for practical circuits because of the relatively small number of partially overlapping covers. We utilize the greedy-addition heuristic [5].

Once a subset of candidate sets has been selected, each of these is used to drive a clock gate and produce a single gated clock signal. This clock is then connected to the covered registers.

IV. POST-GATING OPTIMIZATION

The insertion of a clock gating condition creates a set of *observability don't care*s (ODCs) for the next state function $F_R(x)$ at the input of register *R*. When the gated clock signal is inactive, the value of the next state function is irrelevant; the output of the register will remain constant. This fact can be used to minimize the logic implementation of the next state function.

In general, the task of reducing a large logic network with ODCs is difficult, but in this specific case, a structural simplification can be immediately applied. Let h be an immediate fan-out of the node of literal $g(x)$. If the combinational transitive fan-out of any h does not include any (i) primary outputs, (ii) clock gate inputs, or (iii) register inputs not gated by *g(x)*, this connection can be replaced with a constant. The inserted constants are then propagated forward in the network and any dangling portions dropped. In many instances, the function G_R is constructed of terms entirely from within *R*'s fan-in cone.

Multiple ODC-based simplifications can generally not be simultaneously applied, but in this case, their mutual compatibility is guaranteed because the structure of all G_R signals is perfectly preserved.

Table 1: Results of Fast Clock Gating Optimization

Name	Size		Struct Gating		New Gating				Post-Gating Opt.	
	|and|	*|reg|*	*Gated Regs*	*ΔClkPow*	*Gated Clks*	*Gated Regs*	*ΔClkPow*	*Runtime (s)*	*|and|*	*%Improv*
oc_ssram	274	95	0	0.00%	2	32	13.71%	0.78s	179	34.7%
oc_sdram	894	112	54	23.26%	10	91	34.38%	2.56s	720	19.5%
oc_hdlc	1873	426	62	3.23%	16	133	9.16%	5.03s	1734	7.4%
oc_vga_lcd	6923	1108	659	26.44%	35	707	28.44%	44.0s	6555	5.3%
oc_ethernet	8926	1272	254	4.66%	35	367	8.04%	40.5s	8890	0.4%
oc_cfft	9177	1051	143	3.71%	16	279	7.02%	26.3s	9124	0.6%
oc_8051	9746	754	236	11.03%	34	339	17.92%	58.9s	9622	1.3%
oc_fpu	16260	659	8	0.76%	2	56	2.46%	40.7s	16179	0.5%
radar20	60835	6001	2679	15.06%	81	2968	17.74%	475.3s	60576	0.4%
uoft_raytracer	138895	13079	2216	5.28%	102	2355	5.72%	1691.1s	138542	0.3%
AVERAGE				**9.34%**		**2.1x struct**	**14.46%**			**7.04%**

V. EXPERIMENTAL RESULTS

A set of benchmarks [2] was synthesized to minimize area and delay using the ABC logic synthesis package [3]. The mapped Verilog was then imported into OAGear Func package [6], the platform on which the algorithm was implemented. The internal random simulator and integrated MiniSat package [10] were used. All experiments were done on 2.66Ghz x64 machines.

A representative subset of the results of the gating algorithm is presented in Table 1. The first two columns describe the size of the original circuit. First, a purely structural gating approach was applied to identify MUX-loops and synchronous enables that can be used as gating conditions. Our clock gating algorithm was then applied. The resulting number of gated clock signals is listed in column *Gated Clks*; these signals were shared by the number of registers in column *Gated Regs*. Based upon the simulation data and the capacitance and power values from [1], the estimated power savings of both techniques was measured and presented in the columns labeled *ΔClkPow*.

The post-gating optimization described in Section 4 was then applied, and the resulting netlist optimized with the ABC package using the same procedure described above. The number and improvement in the number of AND nodes are reported in the final two columns. On average, the size of the combinational logic was reduced by 7.0%. In five of the benchmarks, the depth of the combinational logic was also reduced, resulting in a potential performance improvement.

The correctness of both the gating conditions (modeled as synchronous enables) and the logic optimization was successfully verified using comb. equivalence checking.

VI. CONCLUSIONS

We introduced a method for clock gating synthesis that constructs the gating condition out the disjunction of signals and their complements that are already present in the existing logic. Applied to a set of industry-supplied benchmarks, the dynamic clock power consumption is reduced by 14.5% and the size of the logic network by 7.0%.

ACKNOWLEDGEMENT

This work has continued to be developed in collaboration with Christoph Albrecht, Arthur Quiring, and Andreas Kuehlmann. Many wonderful suggestions have also been provided by Alan Mishchenko and Robert Brayton.

REFERENCES

[1] C. Albrecht, IWLS 2005 Benchmarks, http://www.iwls.org/iwls2005/benchmarks.html.

[2] Altera Corp., Quartus II University Interface Program, www.altera.com/education/univ/research/unv-quip.html

[3] Berkeley Logic Synthesis and Verification Group, ABC: A System for Sequential Synthesis and Verification, Release 61225. http://www.eecs.berkeley.edu/~alanmi/abc/

[4] L. Benini, G. De Micheli, E. Macii, M. Poncino, and R. Scarsi, "Symbolic synthesis of clock-gating logic for power optimization of synchronous controllers", ACM Trans. Des. Autom. Electron, Oct. 1999.

[5] S. Khuller, A. Moss, and J. Naor, "The budgeted maximum coverage problem", Information Processing Letters, 1999.

[6] A. Hurst, "OpenAccess Gear Functionality: A Platform for Functional Representation, Synthesis, and Verification," presentation at IWLS, 2006. http://www.iwls.org/challenge/iwls06-oagearFunc.pdf

[7] W. Qing, M. Pedram, and W. Xunwei, "Clock-gating and its application to low power design of sequential circuits", IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, Vol.47, Iss.3, Mar 2000.

[8] P. Babighian, L. Benini, and E. Macii, "A scalable algorithm for RTL insertion of gated clocks based on ODCs computation", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 24, Jan. 2005.

[9] L. Benini, G. De Micheli, "Automatic synthesis of low-power gated-clock finite-state machines", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol.15, Iss.6, Jun 1996.

[10] N. Een and N. Sorensson. An extensible SAT solver. In SAT 2003, volume 2919 of LNCS, pages 502–518, 2004.