Sequential Equivalence Checking for Clock-Gated Circuits

Hamid Savoj[1] David Berthelot[2] Alan Mishchenko[3] Robert Brayton[3]

SavojSolutions[1], CoderCharts[2], and Department of EECS, University of California, Berkeley[3]

hamid@savojsolutions.com david@codercharts.com {alanmi, brayton}@eecs.berkeley.edu

Abstract

We review some general theorems, proved in a previous paper, about situations when SEC can be reduced to a combinational equivalence checking (CEC) problem. Then we show how these can be applied to the verification of sequentially clock-gated circuits.

A new method based on these theorems was applied to 12 large industrial examples which had been combinationally and sequentially clock-gated by a commercial clock-gating tool. These examples lead to SEC problems which are problematic for a general state-of-the-art SEC engine. The new approach completed on all examples and was about 30x faster on the 3 cases where the conventional SEC engine was able solve the problem within a time-out limit of one hour. Compared to the verification of a circuit when only combinational clock-gating was used, the new method for verification on the circuit, which was sequentially clock-gated as well, was only 3-5 times slower.

1 Introduction

Combinational synthesis changes only the combinational logic and keeps the PO and NS functions (as functions of PI and CS) unchanged, although the network computing these functions can be completely restructured. In this case, equivalence checking consists of just checking the equivalence of the combinational parts of the two circuits.

In contrast, sequential synthesis, additionally uses information about the states unreachable from the initial state to modify the combinational logic further. This can consist of (a) using the unreached states as don't cares, (b) using the fact that two states are equivalent, (c) using a different encoding for the reachable states, (d) retiming the FFs, or (e) merging two signals in the circuit if in all reachable states the signals have the same values.

These two types of synthesis lead to different kinds of problems when comparing the original circuit with the synthesized circuit. While combinational equivalence checking (CEC) is NP-complete, sequential equivalence is PSPACE-complete. This paper gives methods for reducing the SEC problem to a CEC problem in some cases which include clock-gating to minimize power.

The paper is structured as follows. Section 2 reviews several theorems which will be used in SEC for clock-gated circuits. Section 3 illustrates some sequential synthesis used in commercial clock-gating software and demonstrates how the theorems can be applied. Section 4 shows experimental results formally verifying the outcome of sequential clock-gating on industrial circuits. The experiments illustrate how the new methods make sequential equivalence checking (SEC) more efficient and practical on problems resulting from clock-gating. Finally, Section 5 summarizes and poses some open questions for future research.

A preliminary version of this work appeared in [6].

2 Sequential Equivalence

In this paper it is assumed that a single initial state is given for a sequential machine. We are not concerned with initializing sequences etc., but follow the philosophy articulated in [1]. Two machines are considered *sequentially equivalent* if starting at their respective initial states, they produce the same sequence of POs when identical sequences of PIs are applied to both of them.

For two sequential circuits, $A = B$ denotes that the circuits are sequentially equivalent starting from the two given initial states. If C and D are combinational circuits, then $C = D$ denotes that they are combinationally equivalent, i.e. for any input, their outputs are identical.

Let A be a sequential circuit and A^1 denote the combinational part of A. Instead of using $(A^1)^k$ to denote A^1 cascaded k times, we just use A^k to denote the *combinational* circuit obtained by connecting k copies of A^1 at the NS (next state) inputs and CS (current state) outputs. The outputs of the combinational circuit A^k are the set of k POs of A, one set for each time-frame plus the NS signals at the output of the k^{th} frame. The inputs of A^k are the k sets of PIs of each frame, plus the initial CS signals at the start of the first frame.

Let A and B be two sequential circuits with the same PIs and POs, and the same number of FFs. (B^n, A^k) will denote the combinational circuit where the NS outputs of B^n are connected to the CS inputs of A^k. The connection is done using some user-provided 1-1 mapping between the FFs of A and B. Note that (B^n, A^k) is a combinational circuit with inputs and outputs of B^n and A^k except for the NS output of B^n and the CS input of A^k, which are connected together.

It is important that to create the related combinational circuit (B, A^{k-1}) from A^1 and B^1, a 1-1 correspondence between the FF's of A and B must exist, so that the

corresponding wires can be attached[1]. This also implies that the two circuits must have the same state encoding.

Theorem 1: *Suppose two sequential circuits A and B have the same PIs and POs. Using some 1-1 mapping between the FFs of A and B to form* (B^n, A^k)*, suppose that* $(B^n, A^k) = A^{n+k}$*. Then A = B, when initialized with the same arbitrary initial state.*

Notation: $[(B^n, A^k) = A^{n+k}]_{\hat{S}}$ denotes that combinational equivalence holds if the CS state input is restricted to $\hat{S}$, and $[A = B]_{\hat{S}}$ denotes that *A* and *B* are equivalent if they are initialized with a state in $\hat{S}$.

A variation of Theorem 1 states that SEC holds after *n* cycles of *A*.

Theorem 2: *Let* R_n^A *be the subset of states that can be reached by A after n cycles starting from any initial state. Then* $[(A^n, B^k) = A^{n+k}] \Rightarrow [A = B]_{R_n^A}$.

In some cases, instead of starting from a state in R_n^A, it is possible to achieve $A = B$ by restricting a subset of FFs to a particular initial state for both *A* and *B*. This happens in SDC clock-gating presented in Example 5. Let *J* be the set of FF's initialized by a unique value at the starting cycle. Let A_J denote machine *A* when its initial state is restricted by *J* and A_J^n denote the *n*-step unrolling A^n when a subset *J* of FFs is initialized with a particular configuration of bits at the starting cycle.

Theorem 3: $[[B_J^n = A_J^n] \wedge [(A_J^n, B_J^k) = A_J^{n+k}]] \Rightarrow A_J = B_J$.

Theorem 4: $[(B^n, A, A^{k-1}) = (B^n, B, A^{k-1})] \Rightarrow [A = B]_{R_n^B}$.

In summary, four results are proved:

1. $[(B^n, A^k) = A^{n+k}] \Rightarrow A = B$
2. $[(A^n, B^k) = A^{n+k}] \Rightarrow [A = B]_{R_n^A}$
3. $[[B_J^n = A_J^n] \wedge [(A_J^n, B_J^k) = A_J^{n+k}]] \Rightarrow A_J = B_J$
4. $[(B^n, A, A^{k-1}) = (B^n, B, A^{k-1})] \Rightarrow [A = B]_{R_n^B}$

A conjecture, suggested by 1, 2, and 4 ,

$$[(A^n, B, A^{k-1}) = (A^n, A, A^{k-1})] \Rightarrow [A = B]_{R_n^A},$$

has been shown to be invalid.

3 Sequential Clock-Gating

Clock-Gating Model. In clock-gating (for power or other purposes), one finds an enable signal, *E*, and a group of flops such that when the enable is false, the clock to the flops is disabled, ensuring that they keep their old values. This can be modeled with a feedback loop through a multiplexor, so that only regular flops are required in the model (as shown in Figure 1).

[1] In some applications the number of FF's in the two circuits may not be equal; this can be addressed by inserting dummy FFs in one of the circuits as illustrated by some of the examples in Section 3.

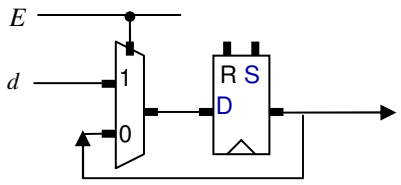
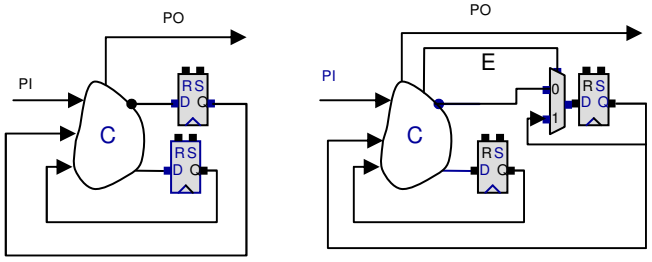


Figure 1: Equivalent model for clock-gating a flip-flop.

If the validity of the enable signal is derived by detecting when the next flop input is the same as its current state [2], then verification only requires showing that the original and final circuits are *combinationally* equivalent because the NS functions are not changed. More powerful enable signals can be found by sequential arguments, e.g. even though the FF input (NS) is different from the held value, the change is not observable at any of flops in the design after a few cycles and also not observable at any of the outputs of the design in any cycle. In this case, verification requires sequential equivalence checking (SEC). Using Theorem 1 of Section 2 this might be reduced to a CEC problem of the form $(B, A^k) = A^{1+k}$. The parameter *k* is the number of cycles after which a difference is not expected to be observable at any of the flops.

Example 1: Figure 2 shows two sequential circuits, *A* and *B,* to be proved equivalent. The designer (or the software synthesis system) expects that the effect of the enabling signal, *E*, in Circuit B is never observable. In fact, the designer expects that any perturbation in the circuit caused by the clock-gating will have died out after *k* steps. However, the internal states for the first *k*-1 steps may be changed. In this case, Theorem 1 can be used.[2].



Circuit A Circuit B

Figure 2: Verifying the enabling signal *E* with Theorem 1, where the circuit *A* and the clock-gated circuit *B* are shown. Only a subset of the FFs have been clock-gated.

The next example illustrates a clock-gating transform whose legitimacy is argued theoretically, but in practice we need to prove the result by equivalence checking.

Example 2: Figure 3 shows an example of a circuit (essentially a one-stage pipeline - see also Example 5) that

[2] There are some unpublished methods which obtain the enable signal by induction and these depend on the initial state. In these cases, we do not know yet how to do SEC easily, and speculate that probably induction has to be part of the SEC method.

has been clock-gated using fanin information. The original circuit is in 3a and the clock-gated circuit is in 3b.

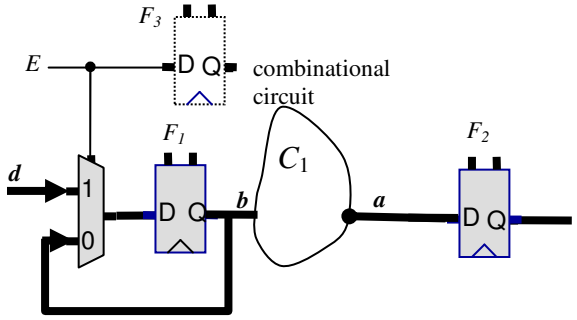


Figure 3a: Initial circuit A. When we match this with the circuit 7b, we insert dummy flop F_3 as shown.

Since the combinational circuit C_1 only has inputs from the flop output signals b, the outputs a of C_1 will not change if b does not change. Since b does not change if E is 0, we can clock-gate the subsequent flops (with inputs a) enabled with a delayed version, E', of signal E as shown.

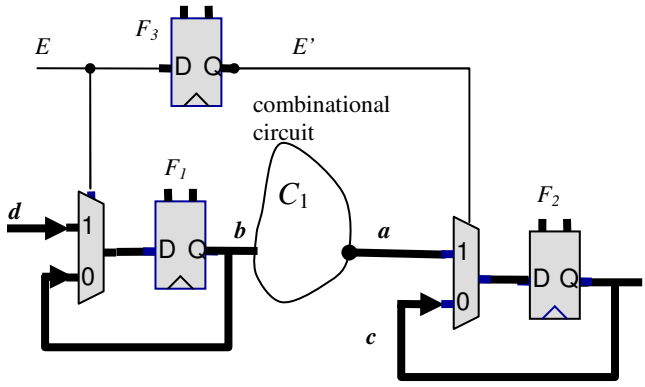


Figure 3b: New Circuit B. Delayed enable signal gating flip-flops at output of combinational circuit C_1.

Since this argument is concerned with the fanin behavior of the clocked flop, the equivalence of the original and clock-gated circuits is provable using Theorem 3 with $n = k = 1$. Note that the original and clock-gated designs differ by an additional flop in the clock-gated version. We matched this by putting in a dummy flop in the initial circuit fed by signal E but with no fanout in the original circuit. Initializing the flop E' to 1 (which is the subset J in Theorem 3) would take care of the requirement that $B_J^1 = A^1$ because this would cause the first cycle to act as if there were no clock-gating.

In Figure 3c, we initialize F_3 to 1 and check the first assumption of Theorem 3 that $B_J^1 = A^1$, where $J = \{F_3\}$.

In Figure 3d, we check the second assumption of Theorem 3 that $(A,A) = (A,B)$. Since all the checks hold, Theorem 3 implies that $A=B_J$.

Adding and removing loops. The following examples deal with adding and removing redundant loops in a circuit and display the asymmetry between these two operations.

Example 3: Figure 4a shows a sequential circuit S_1. The circuit operates as follows. When E = 1 (representing "start"), *BUS-A* provides new values that are loaded into the flops of S_1 as shown. The computation is uninterrupted as long as E = 0 and restarts with new loaded values when E = 1. Further, it is known that a is observable only when E' = 1. E' is just E delayed by 1 cycle (as shown). *BUS-B* is a cut-set representing the internal feedback to the FF of S_1.

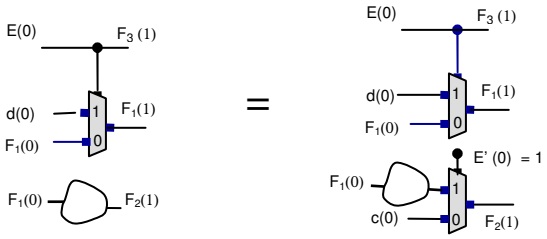


Figure 3c: A and B after one cycle are equal when F_3 is initialized to 1.

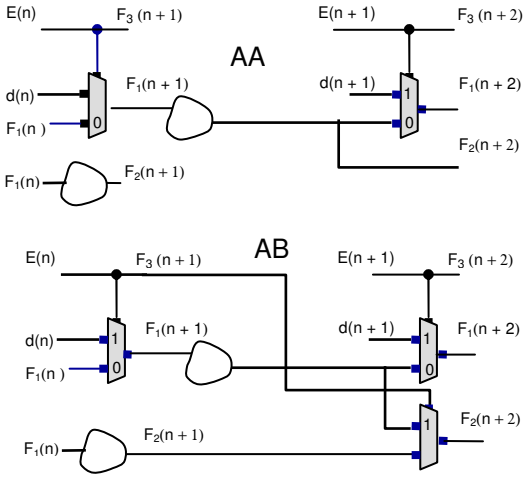


Figure 3d: Checking that $(A,A) = (A,B)$

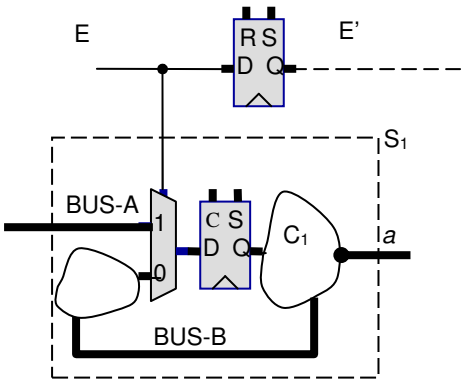


Figure 4a: Removing feedback loops within a sequential circuit S_1

Proposition 1. *BUS-B is redundant.*

Note that not only can *BUS-B* be removed, but also the logic in C_2 and any FF that does not have a combinational path to a can be removed. Simplification of the circuit in Figure 4a leads to the one in Figure 4b.

Conversely, we could clock-gate 4b to produce 4a, knowing that a is not observable when E' = 1.

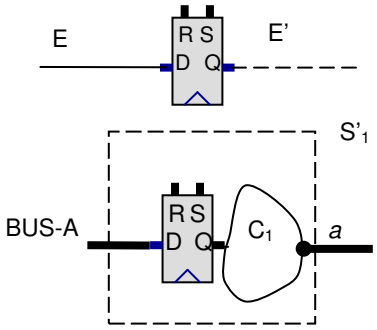


Figure 4b: A sequential circuit S'_1 that can be clock

Proposition 2. *If the output a of* S'_1 *is not observable when E' = 0, then the flops in Figure 4b can be clock-gated using E as the enable signal.*

Even though the application of Propositions 1 and 2 produces sequentially equivalent circuits, Proposition 1 represents loop removal and Proposition 2 represents loop addition. It turns out that the addition and removal of loops are not symmetrical operations with respect to checking equivalence by using Theorem 1.

Example 4: Consider Figure 5, where loops are added during synthesis (as shown by the arrow at the top), producing circuit *B* from *A*. Applying Theorem 1, where $k = 2$, we compare the combinational circuits *A,A* with *B,A* as shown at the bottom of Figure 5. Since the correctness of the *addition* of the loop can be argued using sequential ODCs [3], Theorem 1 will work in this case. (See comments in Figure 5).

However, in Figure 6, a loop is *removed* from circuit *A* to create circuit *B*. Its removal cannot be argued by ODCs, so Theorem 1 is not expected to work necessarily, and indeed it does not work here. (See comments in Figure 6).

Of course, equivalence can be proved with Theorem 1 by reversing the roles of *A* and *B* in Figure 5, but here we are illustrating the asymmetry between adding and removing loops with this example.

Theorem 1, using $(B^n, A^k) = A^{n+k}$, works when *B* has additional loops compared to *A*. However, if there are additional loops in *A*, they block observability propagation and equivalence does not hold. Theorem 2, using $[(A^n, B^k) = A^{n+k}]$, works when *B* has additional loops compared to *A*. If there are additional loops in *A*, they block controllability and equivalence does not hold. This asymmetry raises some issues; adding loops is fine for the application of Theorem 1 and 2, but removing loops is problematic.

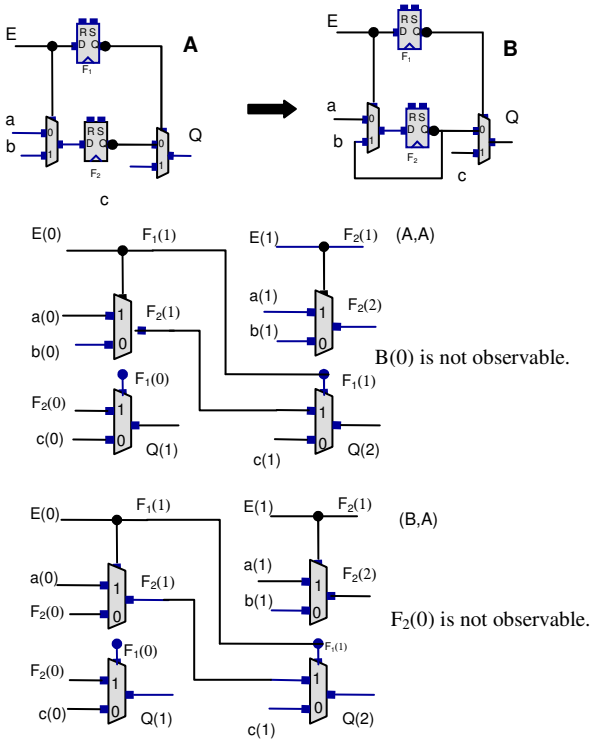


Figure 5: Loops *added* to circuit A to create B. Verification using the Theorem 1 succeeds.

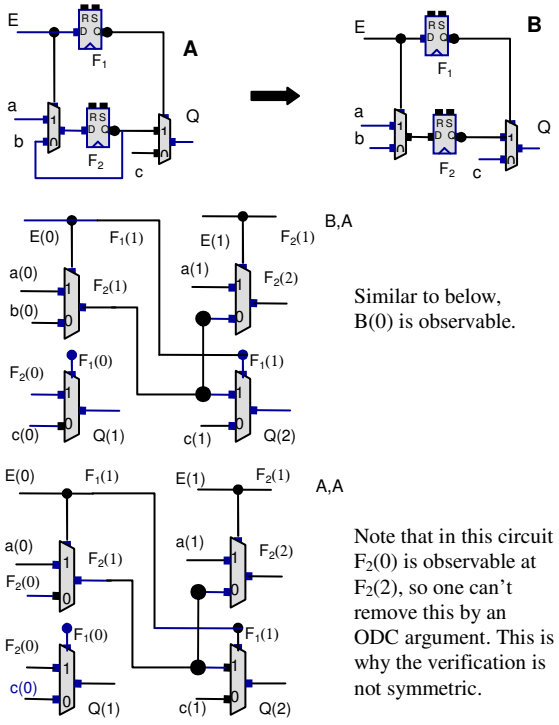


Figure 6: Loops *removed* from circuit A to create B. Verification using the Theorem 1 can't be used.

This is not a problem if loops are only added or only removed (one can simply replace the role of *A* and *B* if

loops are deleted), but it is a problem if a new circuit, A_S, is obtained from *A* by removing some loops and adding others. This problem can be solved by keeping snapshots of clock-gating results during the synthesis..

Pipelines. An important area where sequential clock-gating is used is in the context of pipelines. Figure 3 illustrates the clock-gating of a 1-stage pipeline and an application of Theorem 3.

Example 5. Suppose we have an *n*-stage pipeline and a clock-gating mechanism is given by an external FSM whose POs are used to clock-gate the FFs between the stages of the pipeline. The PIs of the FSM are external signals from the pipeline containing information about which pipe states to disable. This is illustrated with a 3-stage pipeline in Figure 7 where *A* is the original pipeline and *B* the clock-gated version. A simple example of such an FSM would be when a no-OP operation is placed into a pipe. The FSM would be the no-OP indicator delayed at each stage by a flop, the outputs being the delayed no-OP indicators. Since the no-OP information is coming from the fanin structure, Theorem 3 would be applicable.

Given a pipeline with depth *n* with the PIs entering the first stage and the POs exiting the last stage, the FSM must have the following properties:

1) The initial state (given by the *J* FFs) of the FSM is set such that none of the pipe stages are gated initially.
2) If pipe stage 1 is gated at cycle *t*, the PIs must not have changed at cycle *t*-1.
3) If pipe stage *j* >1 is gated at cycle *t*, pipe stage *j* - 1 must have been gated at cycle *t* - 1.

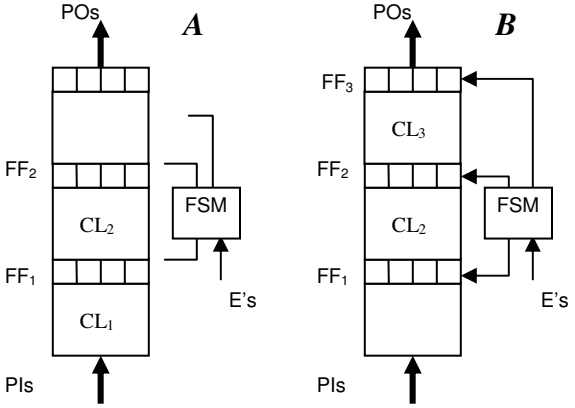


Figure 7: A 3-stage pipeline circuit before clock-gating, *A,* and after, *B.*

Proposition 3. *If the three above conditions 1)-3) are satisfied for an n-stage pipeline, then A = B_J.*

Note that even though the internal states of *A* and B_J are always the same, the *NS* functions of the two machines are not equal, because the $NS^A(t)$ depends on $PI(t)$ and $CS^A(t)$ while $NS^B(t)$ depends on $PI(t)$, $CS^B(t)$ and $CS^B(t-1)$.

Verifying equivalence of such a pipeline would entail the following. *A* and *B* are unrolled *n* times and CEC is done on these two unrolled copies. *A* could have some FFs allowed to power-up in any state. These are replaced by free variables $\boldsymbol{x}$ in the initial copies of *A* and *B*. The remaining flops *J* are given some user-specified initial state. These include the ones in the external FSM existing in both *A* and *B*, but in *A* the outputs of the FSM are not connected, as shown in Figure 7. There are two CEC problems to be checked: (1) check that there is no value of $\boldsymbol{x}$ that produces different states after *n* cycles in the unrolled combinational circuits; and (2) check that $(A^n, B^k) = A^{n+k}$ holds combinationally.

4 Experimental Results

The experimental results are shown in Tables 1 and 2. Table 1 compares the efficiency of applying the new SEC approach of this paper against the general SEC method implemented in ABC [2]. Table 2 experiments on larger problems and compares run times when equivalence checking is done on the designs after a) combinational clock-gating, b) sequential ODC clock-gating, and c) sequential SDC clock-gating.

In Table 1, six large industrial benchmarks were synthesized using sequential clock-gating transforms, based on intuitively correct sequential ODC arguments. The synthesized versions are denoted by *B* and the originals by *A*. Columns 1-5 list the sizes of the circuits. Entries in column 6-11 are the times in minutes needed to verify equivalence. The columns labeled *New* denote the use of Theorem 1 followed by the application of the CEC engine [4] in ABC to prove SEC. The column *ABC general* denotes that the SEC engine *dsec* [5] in ABC, was used. Columns *seq-j* denote experiments where (B, A^j) was compared combinationally with A^{j+1} to show how run-times scale as *j* increases. The items marked with * or **, indicate that the equivalence checking problem timed out.

Observations about Table 1.

1. In general, *New* is significantly faster than *General,* as expected (about 30 times faster for those cases when *General* could complete). The fact that *General* could actually complete on three out of the six large problems was surprising to us.
2. Except for Design 4, the new method scales approximately linearly with the size of the problem.

Table 2 compares the runtime of the new SEC approach of this paper against the runtime of CEC in ABC when combinational clocking was done. Columns 2 and 3 list the size of each design. When a combinational clock-gating algorithm is applied to a design, it first removes all existing clock-gating elements and then extracts new ones. Column 4 shows the number of clock-gating elements and the total number of flip flops that are gated after combinational clock-gating is applied. The new design is verified against the original design using CEC in ABC and the runtimes are shown in column 5. Column 6 adds sequential ODC clock-

gating elements to the designs in addition to the combinational clock-gating already done. The number of additional clock-gaters and FF's affected are shown in this column. The sequential clock-gating uses ODC information from only one frame unrolling of the design. As a result, ODC verification is done using $(B, A^1) = A^2$. The runtime for ODC verification is shown in column 7. Column 8 applies SDC clock-gating to the designs in addition to the combinational clock-gating done. The nature of this clock-gating is similar to the ones explained in Example 5. The SDC clock-gating uses information from one previous frame. The runtimes are shown in Column 9.

Observations about Table 2.

1. The run times of the new sequential verfication approaches explained in this paper, when verification is done using one extra time frame, are only 3 to 5 times that of CEC for combinational clock-gating.
2. Runtimes in Columns 7 and 9 are similar.

The verification runtimes increase by another 3-5 times for every added frame although we have not presented this data in the tables.

5 Conclusions and Future Work

This paper reviewed several methods for sequential equivalence checking, particularly effective in equivalence checking for sequentially clock-gated circuits, resulting in considerable reduction of computational effort. The methods are conservative; if the checks fail no information is obtained about non-equivalence.

Experimental results were given on twelve large industrial SEC problems that had been clock-gated, comparing the sequentially synthesized design against the original design. It was demonstrated on three of the problems that the new SEC method was about 30 times faster than the general-case method. Three other examples were solved while general SEC could not complete. On the other six examples tested, the new method was able to solve them also, and on these, we compared results for purely combinational gated circuits versus that for the case when the circuits were additionally sequentially clock-gated. Run times increased only about 3x-5x for the sequential cases.

Our theorems rely on a one-to-one correspondence between the FFs of *A* and *B*. This is needed for the combinational circuits (*A,B*) or (*B,A*) to be formed where signals in the first circuit are wired to their corresponding signals in the second circuit. Some clock-gating transforms require that an enabling signal be delayed one or more time-frames, adding extra flops. Also in retiming, the number of flops changes. We saw in Section 3 that this can be addressed by introducing dummy FFs in *A* or *B* or both.

Theorem 1 legitimizes sequential synthesis based on unrolling of a sequential machine *A*, *k* times, and combinationally synthesizing the first copy of *A* to obtain a new equivalent sequential machine *B*. However, we have not done experiments on how effective this might be in terms of improved quality of the synthesis result.

Many industrial designs have multiple levels of clock-gating. Each level of clock-gating adds loops to the design and potentially can block application of the theorems given in this paper to the lower levels of clock-gating. The difficult cases of hierarchical clock-gating are left for future research.

Acknowledgements

This work was partly supported by SRC contract 1875.001 and NSA grant "Enhanced equivalence checking in crypto-analytic applications". We also thank industrial sponsors of BVSRC: Altera, Atrenta, Cadence, Calypto, IBM, Intel, Jasper, Microsemi, Real Intent, Synopsys, Tabula, and Verific for their continued support. and we especially thank Michael Bezman of Magma (currently, Synopsys) for conducting updated comparisons on industrial examples.

References

[1] J. Baumgartner, H. Mony, V. Paruthi, R. Kanzelman and G. Janssen, "Scalable sequential equivalence checking across arbitrary design transformations." *Proc. ICCD'06*.

[2] Berkeley Logic Synthesis and Verification Group. *ABC: A System for Sequential Synthesis and Verification*.

[3] A. Mehrotra, S. Qadeer, V. Singhal, R. K. Brayton, A. Aziz, and A. L. Sangiovanni-Vincentelli. "Sequential optimization without state space exploration". *Proc. ICCAD'97*, pp. 208-215.

[4] A. Mishchenko, S. Chatterjee, R. Brayton, and N. Een, "Improvements to combinational equivalence checking", *Proc. ICCAD '06*, pp. 836-843.

[5] A. Mishchenko, M. L. Case, R. K. Brayton, and S. Jang, "Scalable and scalably-verifiable sequential synthesis", *Proc. ICCAD'08*, pp. 234-241.

[6] H. Savoj, D. Berthelot, A. Mishchenko, and R. Brayton, "Combinational techniques for sequential equivalence checking". *Proc. FMCAD'10*.

Table 1: Comparing the new and the general SEC methods. All times are given in minutes.

Design	Statistics				Seq-1		Seq-2	Seq-3	Seq-4	Seq-5
	Ands	FF	PI	PO	New	General	New	New	New	New
cpu	39282	6506	51	83	0.68	15.16	0.98	1.34	1.55	1.78
ethernet	18932	10544	96	115	0.51	18.88	0.7	0.88	1.06	1.25
receiver	31103	7276	105	79	0.78	*60.55	1.29	1.51	1.69	1.63
dram_controller	81782	13822	394	703	1.61	*152.21	2.34	17.22	72.93	267.83
dm4ch	45241	11595	1741	301	0.94	25.63	1.26	1.63	2.37	**6.18
330HiFi	114824	15284	857	804	2.05	*112.83	3.26	4.09	4.79	6.22

Notes:

1. * General sequence equivalence in ABC timed out. Although time-out was set to 1 hour, we were curious to see if the problem could complete if more time was given. Hence the irregular time-out times.
2. ** Unresolved by ABC combinational equivalence checking
3. Seq-*j* denotes the CEC problem where *j* copies of *A* are used, i.e. (B,A^j) is compared to A^{j+1}.

Table 2: Comparing Combinational, ODC, and SDC verification.

Design	Cells	FF	Comb. Gating #CGs/#gated FF	time min.	plus ODC gating #CGs/#gated FF	time min.	plus SDC gating #CGs/#gated FF	time min.
Glue Logic	5569	2259	109/1630	.05	1/156	.25	1/48	.2
CPU Core 1	65929	12554	341/12066	.5	20/289	2.5	11/196	3
CPU Core 2	126272	22235	789/22227	1	23/241	3.2	21/413	3.5
CPU Core 3	170287	39829	1592/39765	1.5	2/19	5	60/2707	5.5
Video Decoder	418275	62037	1618/61792	4.8	38/919	11.5	338/15738	14
CPU Core 4	298500	75589	2832/75273	3	108/1988	9.2	NA	NA

Notes:

1. All runtimes are on an Intel(R) Xeon(R) CPU X5570 @ 2.93GHz,
2. NA means no new clock-gating of the type is found.