Sequential Analysis Driven Reset Optimization to Improve Power, Area and Routability

Srihari Yechangunja, Raj Shekhar, Mohit Kumar, Nikhil Tripathi, Abhishek Mittal, Abhishek Ranjan
Calypto Systems Division, Mentor Graphics Corporation
Noida, India
{ syechan, rshekhar, mohitk, ntripathi, amittal, aranjan }@calypto.com

Jianfeng Liu, Minyoung Mo, Kyungtae Do, Jung Yun Choi, SungHo Park
S.LSI, Samsung Electronics Co. Ltd
Hwasong-Si, Korea
{jf.liu, my0911.mo, kyungtae.do, jungyun74.choi, sh603.park}@samsung.com

Abstract— **Resets are required in the design to initialize the hardware for system operation and to force it into a known state for simulation or to recover from an error. Given the increasing design complexity and time-to-market pressures, figuring out the registers which do not require resets is extremely challenging. In this paper, we present a novel algorithm which uses observability based sequential analysis to identify the registers in design which do not require resets. With the proposed algorithm, we have seen that in some cases 70% registers in the design can have redundant resets. Further, with removal of the redundant resets on registers up to 22% sequential power savings and up to 3% area reduction post-layout can be obtained.**

Keywords— ***Resets; Observability; Routability; Power Optimization; Sequential Analysis; ASIC***

I. Introduction

Semiconductor chip design and manufacturing in recent years have become increasingly complex with lots of conflicting parameters to be traded off including area, timing, power, time-to-market, quality etc. Market pressure of packing more and more functionality into the chip and additional challenges of going to smaller technology nodes are only adding to the complexity of the design and manufacturing processes. Verification of such complex systems has become very challenging and designers are expected to provide for effective safeguards to account for unanticipated errors.

Resets have long been used in the designs [1] to initialize the state of memory elements, typically registers, at the system startup or for simulation. Resets also provide a convenient way to force the system into a known state in case of an unexpected error. Typically, only a subset of registers in the design requires resets. However, finding optimal set of registers to reset is a very complex problem [2][3] and can require significant manual effort. Since uninitialized registers can result in erroneous design behavior, designers tend to apply resets on significantly more number of registers than is actually required. Often RTL coding guidelines also mandate designers to add resets for all registers in the design, like, ISO 26262 standard for automotive safety [4]. However, this indiscriminate addition of resets on registers has its own penalty. Registers with resets require bigger technology cells to be implemented and hence increase area and power [3][8] of the design. Large reset network can adversely impact routability of the design and lead to timing and congestion issues.

In the subsequent sections, we will discuss the existing methods of reset optimization. We will then describe the observability based sequential analysis and how it can be used to determine whether a register's reset is redundant. We will share the results of applying our redundant-reset-detection algorithm on several industrial designs. Finally, we present the scope for future work.

II. PRIOR WORK

Designers have long known the need for minimizing resets in their designs [2][3]. However, the process of identifying registers which do not require resets is heavily manual and cumbersome. Registers without resets have uninitialized state, X, at the system startup or at the start of simulation and this creates serious verification challenges. X-aware simulation [5][6] has been proposed as an effective method to verify the impact of unknown states during simulation. RTL simulation is known to do optimistic interpretation of Xs [5][6] and on the other hand, even gate-level simulations are known to be X-pessimistic [6]. Of late, several X-propagation techniques [7][8] have been developed to verify the correctness of design functionality in presence of Xs. These techniques evaluate whether an X in the design makes its way to the design output or not.

One popular way of removing redundant resets on registers is to iteratively remove the resets and then make use of X-propagation techniques [7][8] to ensure that design still behaves correctly. However, this method suffers from long runtime and is impractical for modern SoCs which can have millions of registers. Typically a very small number of registers can be optimized using this process and benefits are at best marginal. Designers would benefit from knowing upfront which registers in the design do not require reset and then verifying the correctness of removing resets from them in one pass. One other technique for reset removal has been described in [3]. This method involves forcing reset-value of a register and identifying registers in the resulting redundant regions. The method requires designers to try out multiple permutations of registers on which the reset-value is forced and hence can take a lot of time in obtaining a meaningful set of registers from which resets can be safely removed. Clearly there is a need for more automated methods to help identify significant enough number of registers with redundant resets whose optimization can provide meaningful impact on design metrics (area, power, routing congestion).

In recent years, there has been a growing use of sequential techniques to identify redundancies in design [9][10][11]. The successful adoption of such sequential techniques by design community is aided by the availability of Sequential Equivalence Checkers [12] which are adept at verifying the correctness of design changes which are sequential in nature (flop cloning, pipelining, retiming, memory gating etc.). Of these, sequential clock gating [10][11] has seen a very wide adoption in low power chip design flows and is known to optimize a large percentage of registers in the design thereby providing significant power savings with a short turn-around time. Sequential clock gating technique uses multi-cycle analysis of the design to identify writes to registers that are either unobservable down-stream or the same value is written in consecutive cycles.

III. Redundant Reset Detection using Observability don't Cares

Observability of any signal in RTL can be defined as the boolean expression under which the corresponding signal's value becomes 'observable' or influences the value of one or more design outputs. *Observability Don't Care* or *ODC* expression of a signal is thus the condition under which the signal's value has no impact on any of the design's outputs [9][13]. For example, in Fig. 1, the *ODC* condition for signal *A*, is given by the expression, $ODC(A) \equiv (B == 0)$, since if *B* assumes a value *0*, then value of *A* is completely unobservable. Note that the ODC expressions are boolean conditions, expressed in terms of design signals, i.e. the supports of the ODC expressions correspond to signals existing in the design.

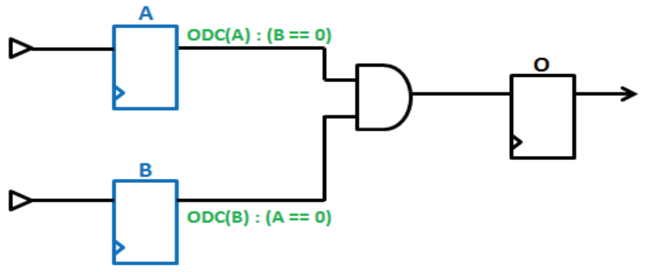

Fig. 1. ODC Expression

Here we introduce the generalized concept of 'sequentially' safe ODC expressions for any register in the design [13]. Sequentially safe ODC expression for any register, *ff(t)* over time *t,* can be defined as the condition which guarantees that the below property holds:

$$(\ SafeODC(ff,\ T) = 1\ ) \Rightarrow \forall_{t \geq T}\ ff(T) \text{ is unobservable at all Primary Outputs of the Design} \quad (1)$$

where *SafeODC(ff)* is the sequentially 'safe' ODC expression for register, *ff*, and *SafeODC(ff, t)* is the value of the ODC expression at time *t*. 'Sequentially safe' property stated in (1) thus implies that at any time *T*, if *SafeODC(ff, T)* evaluates to *true* (or *1*), i.e. the value *ff(T)* is unobservable, then, the value *ff(T)* will be forever unobservable at the design's outputs for all future time $t \geq T$ and the value would instead become obsolete due to a future overwrite.

Let us now consider an additional condition with respect to all time instances when the reset of the particular register is active, i.e.:

$$\forall_{t \in Time(Reset(ff)=active)}\ SafeODC(ff,\ t) = 1 \quad (2)$$

where *Time(Reset(ff)=active)* is the set of all time instances when *Reset(ff)* is active for corresponding register output signal, *ff.*

The condition mentioned in (2) implies that for all time instances when the reset of register *ff* is active, the *SafeODC* condition at these times also evaluates to *true*, i.e. the register output is unobservable at all these time events. Given that $SafeODC(ff,\ \tau) = 1,\ \tau \in Time(Reset(ff)=active)$, at these time events, using (1), it follows that the value, *ff(τ)* is also unobservable at the design outputs for all future time $t \geq \tau$. In case of asynchronous resets, we define the reset value of a register, *ff*, say *ResetValue(ff)*, as the value assumed by the register whenever its reset, *Reset(ff),* is triggered. Thus it can be easily seen that *ResetValue(ff)* is never observable at any design output, if the property stated in (2) is known to hold. Equivalently, it can be said that the reset of the register, *ff*, is itself redundant, since the *ResetValue(ff)* is never observable at any design output. Equation (2) can be rephrased as following:

$$SafeODC(ff,\ t)\ |_{Reset(ff)=active} == \top,\ \text{i.e. is a tautology} \quad (3)$$

Equation (3) can be verbosely stated as, 'whenever *Reset(ff)* is active, i.e. reset signal is triggered, the *SafeODC(ff, t)* evaluates to a *tautology*'. The 2 properties, stated in (1) and (3) together provide the set of registers which have resets deemed as redundant, since their reset-value themselves are never observable. Note that the above analysis can be readily extended to handle synchronous resets. For the sake of brevity, we are leaving the details from this work. In the subsequent sections, all references to *reset* imply use of *asynchronous* resets.

IV. Proposed Algorithm

Here we propose a novel algorithm based on Observability based Sequential Analysis to identify the registers in the design which have redundant resets. The steps in the proposed algorithm can be listed as follows:

A. Collection of SafeODC for all registers

In the first step, we calculate the *SafeODC* for all registers in the design as defined in (1). There are multiple methods to achieve the same. Sequential Analysis as outlined in [9][10][11][13] for clock-gating, is one of the ways of calculating the *SafeODC* for all design registers in one shot, such that the ODC expressions for all registers are *functionally safe* with respect to the design outputs.

B. Evaluating SafeODC of each register during reset

In order to evaluate the expression in (3), substitute every register driven support of the *SafeODC(ff)* expression, say *suppff*, having same reset as the target flop *ff*, by their respective reset values, *ResetValue(suppff)*. For all other supports, we replace them by independent symbols, uncorrelated to rest of the expression. Assuming the *SafeODC(ff)* expression to be optimally represented, one may trivially do a structural constant propagation on this substituted expression, in order to evaluate the value of the *SafeODC(ff)* when the reset of *ff* is active.

C. *Identification of redundant resets*

Once we have evaluated the *SafeODC(ff)* expression during the reset state of register, *ff*, based on properties stated in (3), we mark the register *ff* to be having a redundant reset, if and only if, the evaluated expression equates to a *tautology*, i.e. reduces to a constant-1 (*true*) boolean expression. Thus all registers which evaluate to constant-1 boolean expressions constitute our resultant set of registers which have redundant resets and can be optimized by simply replacing their RTL definitions by the definition of a non-resettable register.

Based on the above description, the algorithm can be more formally described as below:

Inputs:

- Design RTL, *D*.
- Register Set, *FF(D)* = registers in D having resets.
- *ResetDomain(ff)* = Reset domain for *ff*.
- *ResetValue(Sig, RD)* = Reset value for signal, *Sig*, under reset domain, *RD*.

Output:

- Resultant set of registers deemed to have redundant resets, *RFF(D)*.

Pseudo-code:

```
Procedure IDENTIFY_REDUNDANT_RESETS(D, FF(D))
begin
    RFF(D) = {} // Initialize output to empty set.
    // Compute SafeODC expressions for all registers using deep sequential
analysis in sections II and III
    SafeODC = ObservabilityBasedSequentialAnalysis(D, FF(D));
    foreach ff in FF(D) and SafeODC(ff) != 0:
        ffODC = SafeODC(ff);
        ffRD = ResetDomain(ff);
        ValueMap = {}
        foreach support in Supports(ffODC):
            if (ResetDomain(support) == ffRD):
                ValueMap[support] = ResetValue(support, ffRD);

        subODC = EvaluateBySubstitution(ffODC, ValueMap);
        if (subODC == $\top$)
            Insert ff into RFF(D);
    return RFF(D);
end
```

An illustration of the algorithm is provided in Fig. 2. Observability analysis would yield the *sequentially safe* ODCs:

$$SafeODC(F2) \equiv (S2 = 0) \text{ and } SafeODC(G2) \equiv (S2 = 1)$$
$$SafeODC(F1) \equiv (S1 = 0) \text{ and } SafeODC(G1) \equiv (S1 = 1)$$

Applying the algorithm for register *G2*, i.e. evaluation of (3) would yield:

$SafeODC(G2, @R{=}1) == \top$

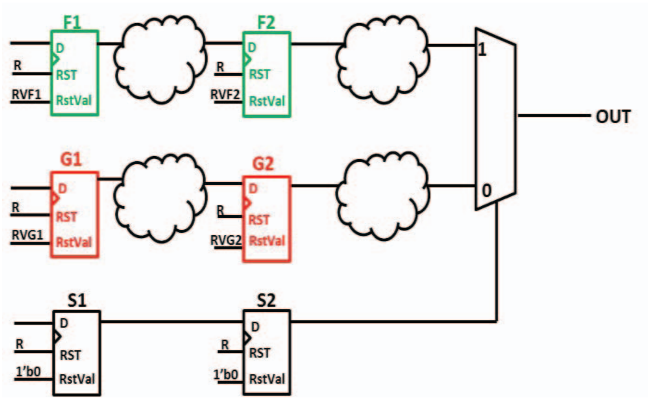

Fig. 2. Simple algorithm illustration

i.e. $\textbf{LHS} = (S2(@R{=}1) == 1)$
$= (ResetValue(S2, R) == 1)$
$= (0 == 1)$, *which is always false*, $\perp$.

Thus register *G2* cannot be considered to have its reset redundant. Applying similarly the properties for register *F2*, (3) would yield:

$SafeODC(F2, @R{=}1) == \top$

Thus register *F2* can be considered to have its reset redundant. Similar analysis would yield *F1* to also satisfy equations (1) and (3) and thus resets may be optimized away from registers *F1* and *F2,* but not from *G1* or *G2*.

V. Experimental Results

The proposed algorithm was applied on six industrial designs from variety of applications: multi-media, video processing, storage devices and smart-phone. Details of the designs are available in first three columns of Table 1 where we list out the total gate count (in KG) and percentage of registers which had asynchronous resets originally in the design. To assess the impact of the proposed algorithm on design's sequential power, area and reset network, we used an RTL Power Analysis tool [14]. The physical impact of removing a reset from register is that it will be bound to a smaller size technology cell. A smaller technology cell will consume less power (both leakage and dynamic) and will also occupy less area. We also picked technology libraries from three different ASIC process nodes (55nm, 28nm and 14nm from TSMC) to estimate the impact of our proposed algorithm.

Table 1 contains the results on the aforementioned six designs for three different process technologies. The observed results show a maximum of 70% reduction in load of reset network, 22% sequential power saving and 2% design area reduction. The sequential power savings are more pronounced for 55nm process technology, they are still very substantial for the other two technology nodes (28nm and 14nm). The runtime of the algorithm was under 10 minutes for 0.4M gate design.

One key impact of reset optimization is the reduction in load of reset network. This leads to improved routability of the overall design and results in better layout. To measure the impact of proposed reset optimization on layout, we took these six designs through the complete implementation flow (synthesis, placement, clock-tree-synthesis and routing) shown

TABLE I. RTL POWER ANALYSIS RESULTS WITH RESET OPTIMIZATION FOR VARIOUS TECHNOLOGY LIBRARIES

Design	Gate Count (KG)	Reg. with Resets (%)	Run time (secs)	Reset Load Reduction (%)	Lib. 55nm		Lib. 28nm		Lib. 14nm	
					Seq. Power Savings %	Des. Area Savings %	Seq. Power Savings %	Des. Area Savings %	Seq. Power Savings %	Des. Area Savings %
DESIGN A	94.26	97.11	114	72.5	22.0	1.4	5.6	1.0	7.8	1.1
DESIGN B	150.52	100	311	48.2	16.0	2.0	3.3	1.5	3.2	1.8
DESIGN C	211.14	100	549	18.8	11.2	0.7	4.0	0.5	3.9	0.5
DESIGN D	354.99	100	221	10.6	8.1	0.5	1.3	0.4	1.5	0.6
DESIGN E	143.42	100	37	11.9	9.2	0.6	1.5	0.5	0.9	0.7
DESIGN F	42.56	28.89	10	19.5	5.1	0.3	2.7	0.3	3.0	0.4
AVG ; MAX	116; 355	87.7; 100	207; 549	30.3 ; 72.5	11.9; 22.0	0.9; 2.0	3.1; 5.6	0.7; 1.5	3.4; 7.8	0.9; 1.8

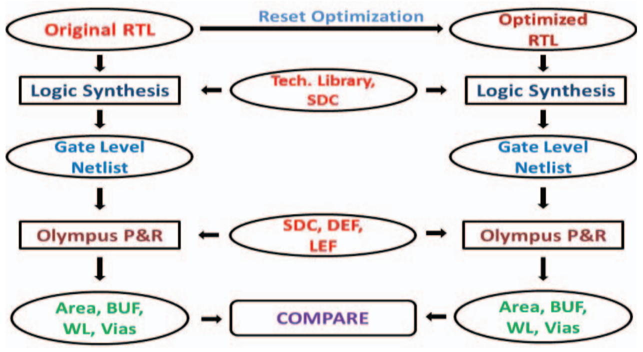

Fig. 3. Flow chart for Place-and-Route experiment.

in Fig 3. An industrial strength logic synthesis tool was used for synthesizing the RTL design. Olympus P&R tool [15] was used for the purpose of placement, clock-tree-synthesis and routing. To compare the impact on layout, we chose to measure impact on: utilized die area, number of buffers and inverters, post-routing wire-length and total number of vias used during routing. Table 2 has the results showing impact of our proposed reset optimization algorithm on the six industrial designs. There is a maximum of 3% reduction in post-layout utilized area, up to 7% reduction in buffer/inverter count, up to 2% reduction in post-routing wire-length and up to 3% reduction in via count. Thus the layout has benefitted substantially due to the smaller reset network as well as registers with smaller sizes.

TABLE II. POST LAYOUT EXPERIMENT RESULTS

Design	Metrics			
	% buffer / inverter reduction	*% utilized die area reduction*	*% wire length reduction*	*% via reduction*
DESIGN A	6.43	2.86	0.30	0.86
DESIGN B	4.18	2.41	1.91	2.77
DESIGN C	0.94	1.24	1.45	1.63
DESIGN D	1.19	0.75	1.58	0.97
DESIGN E	1.40	0.78	1.08	0.84
DESIGN F	2.86	0.59	0.59	1.66
AVG ; MAX	2.8; 6.4	1.4; 2.9	1.2; 1.9	1.5; 2.8

VI. FUTURE WORK

In future, we intend to augment our proposed algorithm to use "stability" property of registers to figure out reset requirement. As of now, the designers have to make manual or scripted changes to RTL based on suggestions from our proposed algorithm. We intend to work on methods to make such RTL changes automatically and additionally improve Sequential Equivalence Checkers to provide an automated way of verifying the correctness of this optimization.

REFERENCES

[1] Clifford E. Cummings, Don Mills, and Steve Golson, "Asynchornous and Synchronous Reset Design Techniques – Part Deux," *SNUG 2003*

[2] H. Chou, K. Chang and S. Kuo, "Handling don't-care conditions in high-level synthesis and application for reducing initialized registers," *Design Automation Conference (DAC)*, pp. 412-415, 2009.

[3] Valavan Manohararajah, Altera Corp. "Method and Apparatus for Performing Asynchronous and Synchronous Reset Removal During Synthesis," US8578306 B2.

[4] "ISO 26262: Functional Safety Draft Standard for Road Vehicles," http://sesamo-project.eu/sites/default/files/downloads/publications/iso-26262-dis-tutorial-2010-final.pdf.

[5] Greene, Salz and Booth, "X-Optimism Elimination during RTL Verification", *SNUG*, San Jose, 2012.

[6] Stuart Sutherland, "I'm Still in Love with My X!," *Design and Verification Conference (DVCon)*, 2013, http://www.sutherland-hdl.com/papers/2013-DVCon_In-love-with-my-X_paper.pdf.

[7] Evans, Yam and Forward, "X-Propagation: An Alternative to Gate Level Simulation", *SNUG*, San Jose, 2012.

[8] Lisa Piper, "X-Verification Methodology for Both Designers and Verification Engineers," Real Intent Inc., Sunnyvale, CA, http://www.eejournal.com/archives/articles/20131205-realintent/

[9] H. Kapadia, L. Benini, and G. De Micheli, "Reducing Switching Activity on Dathpath Buses with Control-Signal Gating", *IEEE Journal of Solid-State Circuits*, 34(3), 405- 414 (Mar., 1999).

[10] N. Vyagrheswarudu, S. Das, A. Ranjan, "PowerAdviser: An RTL Power Platform for Interactive Sequntial Optimizations", *DATE*, pp. 550-553 , 12-16 March, 2012.

[11] J. F. Liu, M. S. Hong, Mohit Kumar, A. Ranjan et. al., "Clock Domain Crossing Aware Sequential Clock Gating", *DATE*, 9-13 March, 2015.

[12] Pascal Urard, A. Maalej, N. Chawla, "Levereging Sequential Equivalence Checking to Enable System Level to RTL flows,", *Design Automation Conference (DAC)*, pp. 816-821, 2008.

[13] P. Babighian, L. Benini and E. Macii, "A scalable algorithm for RTL insertion of gated clocks based on ODCs computation", *IEEE Trans. on Comput.-Aided Des Integr. Circuits Syst.*, vol. 24, pp. 29–42, Jan. 2005

[14] PowerPro, Calypto Design Systems. (http://www.calypto.com/)

[15] Olympus SoC, Mentor Graphics Inc. (http://www.mentor.com/)