32

# Power Reduction Through Clock Gating by Symbolic Manipulation. *

Frans Theeuwen+, Eric Seelen ++

+ Eindhoven University of Technology, P.O. Box 513 5600MB Eindhoven, The Netherlands, email: J.F.M.Theeuwen@ele.tue.nl

+ +Philips Research Laboratories, Prof Holstlaan 4, 5656 AA Eindhoven, The Netherlands, email: seelen@natlab.research.philips.com

**Abstract**

A method to reduce power dissipation by automatically synthesizing gated–clocks in synchronous static CMOS circuits is presented. This synthesis is performed on the gate level description of the circuit. The boolean behavior of the inputs of the flip–flops is determined by examining the network. This behavior is represented in ROBDD's. Analysis of these equations results in the condition for which flip–flops do not need to be clocked. Flip–flops are grouped in so called hold domains, and clocked by a gated–clock signal. Power reductions of up to 29% are found. There is only a small area overhead (less than 8%). Testability of the resulting design is taken care of.

**Keywords**

Integrated digital circuit design, low–power design

## 1 INTRODUCTION

Due to the continuously decreasing feature sizes and the increasing clock frequencies on integrated digital circuits, power dissipation is growing to be one of the major concerns during the design of an integrated circuit. Examples of this phenomenon are for instance the DEC Alpha chip (dissipating 30 Watts at 3.3V, 200 MHz) and the SUN Viking (dissipating 8 Watts at 5V, 50 MHz).

Currently many circuits are designed by describing them in a behavioral description language like VHDL or Verilog. By using a synthesizer, this description is synthesized into a gate level netlist. This way of designing saves a lot of design time compared to traditional design methodologies like schematics entry. Most synthesizers are currently targeted towards fully synchronous designs, suitable for scan chain insertion, to be able to test the circuit by scan testing.

One of the main contributors to power dissipation is the clock tree. The clock net is one of the nets with the highest switching density. The clock net is also a net with a

This research has been sponsored by the European Jessi AC 8 project

VLSI: Integrated Systems on Silicon R. Reis & L. Claesen (Eds.)

large fanout (all flip–flops are connected to the clock net), resulting in high power dissipation. This clocking produces power dissipation on two points:

- Dissipation in the clock drivers and the clock lines.
- Dissipation in the flip–flops (most flip–flops contain an inverter connected to the clock signal.)

In micro processor like designs there is a large number of registers that are there to hold their data most of the clock cycles. Analysis of the circuits generated by logic synthesizers dissolves that this functionality is implemented by providing a conditional loop back from the output of the flip–flop to its input. If such a loop back is active, the flip–flop needs not to be clocked, because the value of the flip–flop will not change; the flip–flop is in the so called "hold mode", however due to the implementation with a loop back unnecessary power is consumed. A promising technique to reduce the power dissipation of the clock net is selectively stopping the clock in parts of the circuit, called "clock gating". This technique is not new at all and already applied in a number of ways. In (Schutz 1994) and (Suessmith et al. 1994) this technique is applied during the design of microprocessors, however the places where the gated–clocks are inserted are determined by the designer. In (Benini et al. 1995) an automatic method to insert gated–clocks in finite state machines is presented. Although every sequential circuit can be modeled as a finite state machine, this technique only works if the symbolic transition table of the implemented finite state machine is known. For large circuits this is an impractical approach. In (Benini et al. 1997) the problem of generation the state transition table is circumvented, but still the idea of modelling the circuit as a single finite state machine is used. So only clock gating can be applied if the whole FSM (or design) performs a so called "self loop". In practice however only some parts of a design can be switch off by clock gating. The tool presented in (Benini et al. 1997) will not be able to find these situations. In (Papachristou et al. 1995) a power saving technique is shown that during architectural synthesis determines which flip–flop can be switched off during the operation of the circuit. In this paper we present a method to generate gated–clock circuits starting from a netlist resulting from for instance a logic synthesizer. The idea of the developed method is to identify the flip–flops in the design that keep their data for a large portion of the clock cycles. For these flip–flops the condition will be determined for which they keep their data and the circuit will be transformed in such a way that the clock signal will be switched off if the condition is satisfied. In section 2 definitions will be presented. Section 3 will describe how the transformation is determined. Section 4 will give some implementation details. Results and conclusions are presented in section 5 and 6.

## 2 DEFINITIONS

A mapped network $N$ of logic gates will be represented by a logic network graph $G_n = (V, E)$. $V$represents the primary input and output terminals and the local functions (i.e. the gates). The set of directed edges $E$ represents the decomposition of the

multi–terminal nets $n \in N$ into two terminal nets, directed from the output pin of a gate or a primary input to an input pin of a gate or a primary output.

We consider the behavior of a gate (so also the behavior of the corresponding vertex $v \in V$) as a completely specified function $f_v : B^n \rightarrow \{0, 1\}$. Where $B^n$ is the set of all primary inputs and all the flip–flop outputs. The behavior $f_n$ of a net $n$ is defined as the behavior of the source vertex $s$ of the net. The co–factor of a function $f(x_1, x_2, ..., x_i, ....x_n)$ with respect to a variable $x_i$ is $f_{x_i} = f(x_1, x_2, ..., 1, ....x_n)$. The co–factor with respect to $x_i'$ is $f_{x_i'} = f(x_1, x_2, ..., 0, ....x_n)$. The consensus of $f(x_1, x_2, ..., x_i, ....x_n)$ with respect to a variable $x_i$ is $CON(f(x), x_i) = f_{x_i} f_{x_i'}$ (De Micheli 1994). The consensus of a function with respect to a variable represents the component that is independent of that variable. The consensus operator can be extended to sets of variables as an iterative application of the consensus operator on the variables of the set. The equivalence on two function $f$ and $g$ is

$$\mathcal{E}(f, g) = NOT(f \otimes g). \tag{1}$$

## 3 THE CIRCUIT TRANSFORMATION

As described in the introduction the basic idea of the transformation is to switch off the clock of flip–flops that take their own data. This is only possible if there exists a path in the network graph from the output of a flip–flop to its own input. If the flip–flop has to keep its value, the output value is fed back to the input of the flip–flop. In the transformed circuit the clock of the flip–flop that has to keep its data will be switched off. This will result in the circuit transformation shown in Figure 1.

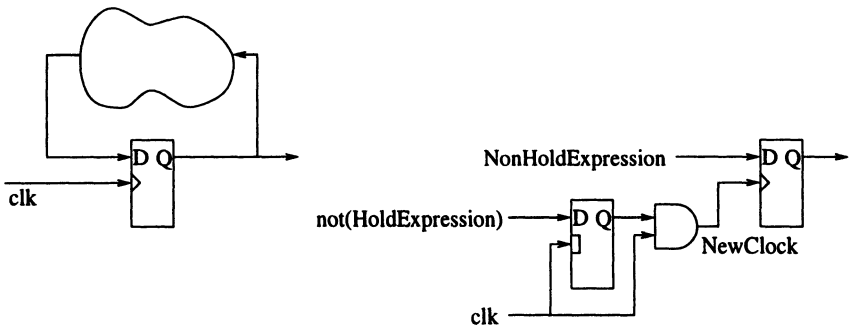


**Figure 1.** The circuit transformation.

Two new signals have to be generated:

1 The HoldExpression signal. This signal determines when the system clock will be fed to the flip–flop.

2 The NonHoldExpression signal, being the new value of the flip–flop if the flip–flop is not in hold mode.

To circumvent glitches on the gated–clock signal caused by possible glitches occurring on the signal HoldExpression, a latch that is transparent if the signal clk is low, has to be introduced.

### 3 .1 Synthesis of the hold expression

Assume there exists a flip–flop $ff \in V$ in the logic network graph $G_n = (V, E)$. The net connected to the data input of the flip–flip will be denoted $d_{ff} \in E$. The data output of the flip–flop will be denoted $q_{ff} \in E$. The expression describing the condition for which $f_{d_{ff}} \equiv f_{q_{ff}} = q_{ff}$ will be called the hold expression $h_{ff} \equiv HoldExpression$. To determine the expression $f_{d_{ff}}(x_1, x_2, \ldots, x_n)$ where $B^n$ is the set of all primary inputs and all the flip–flop outputs, a traversal of the transitive input cone of node $d_{ff}$ in a topological order has to be performed.(Cormen et al. 1989). The behavior of the output signal of a gate is determined from the behavior of its inputs and the function of the gate. After the behavior of the input of a flip–flop has been computed the hold expression can be determined by:

$$h_{ff} = \mathfrak{E}(f_{d_{ff}}, q_{ff}). \tag{2}$$

In most cases the hold expression computed by (2) appears to be a rather complex expression. This is due to the fact that expression (2) not only expresses the situation when the feedback loop around the flip–flop is active but also when the input of the flip–flop is accidentally equal to the new data that will be clocked into the flip–flop. This part of the hold expression is called the "data dependant part" of the hold expression. Experiments have shown that the data dependant part of the hold expressions has two effects:

1 The hold expression including the data dependent part is much more complex then the hold expression without the data dependant part.

2 The hold expressions for the different bits of a register in an arithmetic unit (adder, counter, subtracter etc.) are unequal to each other because of the data dependance. This will complicate the comparison of the hold expressions while composing a hold domain.

It is undesirable that the hold expression is a rather complex expression because the hold expression has to be implemented in hardware, resulting in extra silicon area and also extra power dissipation.

The control signals are in general the signals that determine when registers are in hold mode. The control signals are determined in the controlling finite state machine of the circuit or are primary input signals of the circuit. It are these signals that will enable and disable the feedback path around a flip–flop. If from the hold expression only the part that is described by the control signals is implemented it is likely that the implemented hold expressions remain simple. If the hold expression is described by only control signals the data dependant hold expressions will not be covered. As a first approximation signals that are member of a bus will be seen as data signals. All single

signals will be viewed as control signals. The control signals are defined as $C \subseteq E$ being a subset of all the signals of the circuit. The data signals are defined by $D = E - C$. The hold expression described by only control signals can be computed by:

$$hc_{ff} = CON(h_{ff}, D) \tag{3}$$

### 3.2 Determination of the hold domains

Once the hold expressions for all the individual flip–flops are determined, flip–flops are to be grouped into so called hold domains. A hold domain is a group of flip–flops whose members are connected to the same gated–clock signal. The condition for which all the flip–flops of a hold domain $D$ are in hold mode is described by the hold domain expression $H_D$. $H_D$ can be computed by

$$H_D = \prod_{ff \in D} hc_{ff} \tag{4}$$

The construction of the hold domains has to be such that the power reduction is as large as possible. The power reduction depends on:

1. The number of flip–flops in the hold domain defined as $\| D \|$
2. The relative number of clock cycles that the hold domain is in hold mode defined as $|H_D|$

Determination of $|H_D|$ is rather complex. In the general case $|H_D|$ depends on the probability of all the signals in the support set of $H_D$ and the correlation between those signals. As an approximation the signals in the support of $H_D$ will be assumed uncorrelated. The probability of these signals being "1" will be assumed to be 0.5. With these assumptions $|H_D|$ equals the part of the boolean space $B^{support\ of\ H_d}$ where $H_D$="1". This can be determined easily (Janssen).

To obtain a power reduction as large as possible $\|D\| * |H_D|$ will be maximized. This will be done by the algorithm shown below.

$V := \{hc_1, hc_2, \ldots, hc_n\}$
*While*$(V \neq \emptyset$ {
  $D := \{hc_k\} \wedge \forall x \in V : |x| \leq |hc_k|$
  $H := hc_k$; $V := V \backslash D$; *changed* $:=$ *true*
  *while (changed)* {
    $Test := hc_m \wedge hc_m \in V \wedge$
      $\forall x \in V : |x \wedge H| \leq |hc_m \wedge H|$;
    if$(\|D\| \times |H| \leq (\|D \cup \{Test\}\| \times$
      $|Test \wedge H|)$ {
      $D := D \cup \{ Test \}$
      $V := V \backslash \{ Test \}$
      $H := H \wedge Test$

```
        }
        else changed := false
    }
if( || D || ≥ Threshold )
    implement hold domain D
else
    V := V ∪ D\{hc_k}
}
```

## 4 IMPLEMENTATION OF THE HOLD DOMAIN EXPRESSIONS

The most simple way to implement the hold domain expressions is to use a logic synthesizer and technology mapper. If they are expressed in terms of primary inputs and flip–flop outputs the expressions can be rather complex. So direct implementation will result in extra area and dissipation.

### 4 .1 Optimization of the hold domain expression

In the existing circuit there are a lot of intermediate signals that can be used to optimize the hold domain expressions. In many circuits the hold domain expression is already implemented and used to control a multiplexer that constitutes the temporal feed back loop as shown in Figure 1. To optimize the hold domain expression all the signals in the input cone of the flip–flop data inputs in the hold domain are tried to simplify the hold domain expression. The algorithm checks which local net simplifies $H_d$ as much as possible. This net is used to simplify $H_d$ and a new net is searched for.

Simplification is based on strong division. The result of a strong division of a boolean function $f$ by a boolean function $g$ is:

$$f = g.a + r \tag{5}$$

The quotient $a$ and the remainder $r$ can be calculated as:

$$a = f.g + X.g' = ite(g,f,X) \tag{6}$$

$$r = f.g' + X.f.g = ite(g,X.f,f) \tag{7}$$

with $X$ representing the don't care constant. $ite(a, b, c)$ is the "if–then–else" operator defined as (Janssen)

$$ite(a,b,c) = a.b + a'.c \tag{8}$$

Not in all cases the above described algorithm results in satisfactory results, sometimes the hold domain expression is still too large. In these cases the circuit shown in Figure 2. is generated. This circuit determines when the input signals of the flip–flops equal the outputs of the flip–flops. So in that case the clock can be switched off.

### 4 .2 The NonHoldExpression

If a flip–flop is not in hold mode adequate data should be provided to the input of the flip–flop. Two approaches can be followed here:

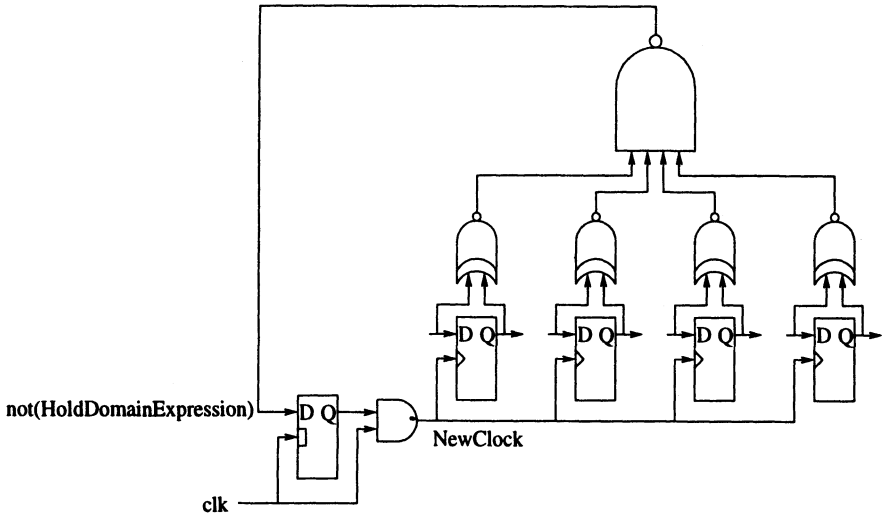


**Figure 2.** Direct implementation of the HoldDomainExpression

1 The input signal $D_{ff}$ of the flip–flop *ff* can be re–synthesized. As the don't care set for this optimization the hold domain expression $H_d$ of the hold domain $d$ to which the flip–flop belongs can be used. Also local nets can be used to simplify the resulting expression.

2 The data input of the flip–flop can be kept as it was in the original circuit (including the feedback path from the output of the flip–flop to its input).

In practice a mixture of these to approaches can be used. If method 1 yields a large expression method 2 can be used.

### 4 .3 Testability

Currently most of the sequential designs are tested using the Scan Test method. This method assumes fully synchronous circuits. The introduction of gated–clocks violates this assumption. Also current test–vector generators assume fully synchronous circuits. As shown in (Favalli et al. 1996) by application of a network transformation (Figure 3.) and by addition of some extra test control signals in the clock generation circuitry as shown in Figure 4. it is possible to generate test vectors and to test the gated–clock circuit.

## 5 RESULTS

The developed tool is tested on two designs. The first being an 8 bit micro controller called CON, the second a 16 bit general purpose signal processor called DSP. The designs the ,tool is applied to are produced by a VHDL synthesizer. To keep the computation time in the order of seconds, the described algorithm is applied to the hierarchical netlist. In this way the ROBDD's do not explode. By applying the tool to the non flat-

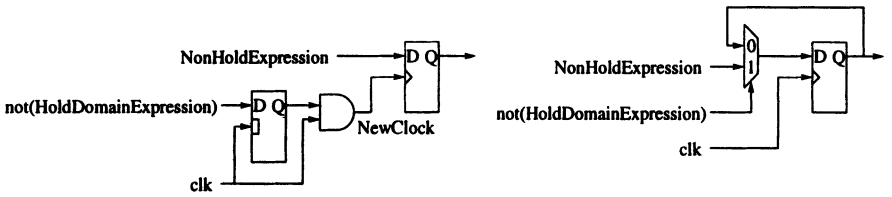


**Figure 3.** Re–modelling of the gated clock circuit for test generation

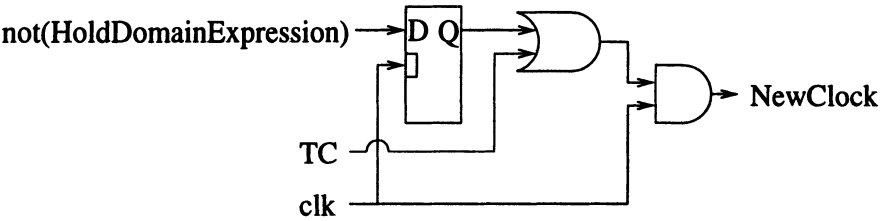


**Figure 4.** Addition of test control signals

tened netlist, hold domains that exist over hierarchy boundaries are not detected. In practice this appears not to influence the results. The results are shown in Table 1.

**Table 1.** Results of clock gating

| | CON | DSP |
|---|---|---|
| # D–ff | 732 | 2183 |
| #D –ff in loop | 528 | 2015 |
| #D–ff in hold domain | 413 | 1548 |
| #hold domains | 55 | 89 |
| #local net impl | 41 | 88 |
| #xor network impl | 14 | 1 |
| av. # d–ff/hold domain | 7.5 | 17.3 |
| size org circuit (in equiv gates) | 10812.6 | 56781.0 |
| size new circuit (in equiv gates) | 11650.2 (+7.7%) | 57214.3 (+0.7%) |
| cpu time (in sec) | 189 | 406 |

Power estimation of the circuits is done with an accurate gate level power estimator. This estimator works together with a logic simulator and counts the number of signal transitions during simulation. Each transition of a net results in a contribution to the power dissipation of the circuit. The tool takes net specific loading, slopes of the signals and type of the driving cell into account. Table 2 shows the dissipation results for design CON.

**Table 2** Power reduction for design CON

| design CON | | | | | | | |
|---|---|---|---|---|---|---|---|
| test # | original | gated–clock | | gated–clock with clcok buffer reduction | | useful gated–clock with clock buffer reduction | |
| | mW | mW | rel | mW | rel | mW | rel |
| 1 | 14.5 | 11.7 | 0.81 | 11.4 | 0.78 | 11.1 | 0.76 |

As can be seen a reduction of 19% can be obtained by application of the clock gating tool. Because of the fact that the loading on the primary clock is reduced considerably (from 732 to 2 * 55 + 732 – 413 = 429) the clock buffer can be reduced. This yields in an extra reduction of 3%. The power analysis tool gives also information about the number of clock cycles a hold domain is in hold mode. If the number of transitions of a gated–clock signal is not significant lower than the number of transitions of the original clock, the clock domain will not contribute to power reduction, so this domain is canceled. This again leads to a reduction of 2%.

Table 3 gives the power dissipation data for the design DSP. As can be seen, the power dissipation and reduction depends on the input data for simulation. An average power reduction of 27% can be obtained. (The clock buffer of this circuit was not included in the design).

**Table 3** Power reduction for design DSP

| design DSP | | | | | |
|---|---|---|---|---|---|
| test # | original | gated–clock | | usefull gated–clock | |
| | mW | mW | rel | mW | rel |
| 1 | 177 | 145 | 0.82 | 134 | 0.76 |
| 2 | 163 | 130 | 0.80 | 119 | 0.73 |
| 3 | 152 | 119 | 0.78 | 108 | 0.71 |
| 4 | 176 | 146 | 0.83 | 134 | 0.76 |

## 6 DISCUSSION AND FUTURE WORK

As has been shown it is possible to obtain a power reduction of up to 29% by applying clock gating techniques on micro controller designs at only moderate area penalty (< 8%). The size of the circuits that can be handled is much larger than has been reported so far. Testability of the transformed circuit is maintained during the transformation and the circuit transformations are kept as small as possible.

### 6 .1 Timing issues

By the application of the described tool, on two places timing problems can occur:

1 The gated clock signal NewClock (Figure 4.) is delayed one gate delay (of the AND gate). This gives rise to clock skew in the resulting circuit.

2 The signal HoldDomainExpression (Figure 4.) can possibly violate the timing constraints.

ad 1) The added clock skew is the same for all the gated clock signals. By clock tree generation extra buffers can be added to the non gated clock signal to compensate for this skew. In the layout phase the flip–flops belonging to one hold domain should be kept close to each other to reduce skew inside a clock domain because of parasitic capacitances due to wiring.

ad 2) In practice, the new circuitry added for the new HoldDomainExpression is very small. The circuits designed until now did not show problems on this point, however there is a potential problem.

### 6 .2 VHDL transformations

As discussed, in the current tool, timing problems can occur. To circumvent these problems, it seems a good idea to perform clock gating transformations before synthesis, i.e. in the VHDL description. In this way timing constraints, given to the synthesizer, will be applied to the gated clock circuit. Initial tests have shown that this approach is feasible.

### References

Schutz, "A 3.3V 0.6μm BiCMOS superscaler microprocessor", IEEE International Solid–State Circuits Conference, pp. 202–203, Feb 1994.

B. Suessmith, P. Paap III, "PowerPC 603 microprocessor power management", Communications of the ACM, nr. 6, pp. 43–46, June 1994.

L. Benini, G. De Micheli, "Transformation and synthesis of FSMs for low–power gated–clock implementation", International Symposium on Low Power Design", 1995.

C. Papachristou, Mark Spining, Mehrdad Nourani, "A Multiple Clocking Scheme for Low Power RTL Design", International Symposium on Low Power Design", 1995.

R. Bryant, "Graph–Bassed Algorithms for Boolean Function Manipulation", IEEE Transactions on Computers, Vol C–35, No. 8, pp. 79–85, August 1986.

T. Cormen, C. Leiserson, R.Rivest, "Introduction to algorithms",McGraw–Hill, New York, 1989, pp. 485–488

Documentation of BDD–package, file "bdd/dec/bdd_fns.doc", FTP:"ftp://ftp.es.ele.tue.nl/pub/geert/bdd.tar.gz", Eindhoven University of Technology, group ICS.

M. Favalli, L. Benini, G. de Micheli, "Design for testability of gated–clock FSM's", Proceedings of the Eurepean Design & Test Conference, 1996, pp. 589–596

G. De Micheli, "Synthesis and optimization of digital circuits", McGraw–Hill International Editions, 1994.

L.Benini, G. de Micheli, E.Macii, M. Poncino, R. Scarsi, "Symbolic Synthesis of clock–Gating Logic for Power Optimization of Control–Oriented Synchronous Networks", Proceedings of the European Design & Test Conference, 1997, pp 514–520.

## 7 BYOGRAPHY

Frans Theeuwen was born in Geleen, The Netherlands in 1954. He received the M.Sc degree in Electrical Engineering in 1979 and his Ph.D. degree in 1985 from the Eindhoven University of Technology. Currently he is an assistant professor at the Eindhoven University of Technology in the CAD group. In 1995 he stayed at the Philips Research laboratories as an advisor for low power design techniques. His main interests are architectural and logic synthesis, testing, module generation and low power design techniques.