EVALUATION ON POWER REDUCTION APPLYING GATED CLOCK APPROACHES

G. Palumbo - F. Pappalardo** - S. Sannella**

*DEES
(Dipartimento Elettrico Elettronico e Sistemistico)
UNIVERSITA' DI CATANIA
Viale Andrea Doria 6,
I-95125 CATANIA – ITALY

**ST-Microelectronics
Stradale Primosole, 50
I-95121 CATANIA – ITALY

ABSTRACT

In this paper the use of the gated clock approach to reduce power consumption is analyzed and compared. The approach has been implemented following three different strategies that allow the approach to be efficiently used under different design condition.

To verify the strength of the approach it has been implemented during the design of a Programmable Interrupt Controller (PIC). The results found show a 2X factor reduction in the average power consumption through the use of the three stategies. Moreover, the results have been also compared with those obtained through an automatic implementation of one of the gated clock strategies allowed by Synopsys's Power Compiler. In this second case only about 25% of power consumption is saved. It is worth noting that implementation of the three gated clock strategies leads also a design with the smallest gate count.

1. INTRODUCTION

The reduction of power consumption has become one of the dominant design aspects for many VLSI systems. This happens both for the remarkable success and growth of the portable consumer electronics and for the continuous growing of the power per unit area reaching very high level of power density [1]-[4].

In order to reduce power consumption in a digital system it is often used a set of strategies of a class named Dynamic Power Management. The common denomination to all the strategies consists in disabling the logic circuits that are not performing functional operations during a particular time frame, thus reducing its power consumption. The strategies of this class work at different design level. They range from system level with a variety of sleep modes to circuit level with the Gated Clock applied to disable the clock of flip-flops [5]-[6].

The main source of power consumption in digital CMOS circuits and systems is due to the dynamic power consumption that for each node can be written

$$P_{dyn} = \alpha f C V_{DD}^2 \qquad (1)$$

where α is called Switching Activity and represent the probability of transitions per clock cycle, f is the clock frequency, C is the capacitance at the node and VDD is the power supply voltage. Hence, the term αf represents the transition frequency at the node. In order to reduce power consumption during design at gate or circuit level, assuming as usually the power supply voltage and the clock frequency defined, we can try to reduce the term αC named "switched capacitance", which represents an effective cost-function to minimize the total dynamic power.

Clock signal represents a large part of system power that can reach percentage up to 45%. These data must be supported by exact data and references. The reasons that determine this high percentage are two:

- the switching activity is equal to one, it is the highest of the system, in fact in a period the clocked node makes a fully transition cycle (i.e., 0->1->0);
- the total node capacitance is high due to the large number of clocked nodes.

As a consequence, reduction of signal clock power consumption can determine a huge reduction on the system power consumption.

In this paper we focus our attention on the gated clock approach that allows to mitigate the switching activity both for the clock tree and for its leaf registers and flip-flops. Since implementation of gated clock needs the use of some extra control logic, one of our goal aims to verify the efficiency of the approach. Moreover, since three different strategies for different design conditions can be applied, the aim is to verify the strength of each strategy. The study has been carried out on an Interrupt Programmable Controller (PIC), fully compliant with the x86 processors family.

TABLE I

```
always @(posedge clk)
begin:ICW1
   if(ireset_ == 1'b0)
       icw1 <= 8'b00010001;
   else
       if(Soft_Reset_ == 1'b1)
               icw1 <= din;
end
```

2. GATED CLOCK

The gated clock approach is a design strategy implemented to avoid burning energy whenever the flip-flops' output does not change. It is based on the conditioning of the clock signal obtained (an activation signal represented) either through an enabling signal or through the flip-flop output itself. [7]-[10].

The digital system is usually described using a high level language such as VHDL or Verilog. For example, an eight bits register composed of edge triggered flip-flops with an enabling signal can be described according to the Verilog code reported in Table I. A synchronous reset is implemented by the first "if" statement, while the second "if" allows both the sample of the new incoming input data and the holding of the previous output value. Synthesis of the register in Table I leads to the circuit scheme depicted in Fig. 1.

In order to reduce the activity on clock node we can enable it only when the register has to sample the new input. It is clear that the energy saving is bigger for low activity input signal. We can gate the clock signal through the enable signal, as shown in Fig. 2. This is exactly the way in which

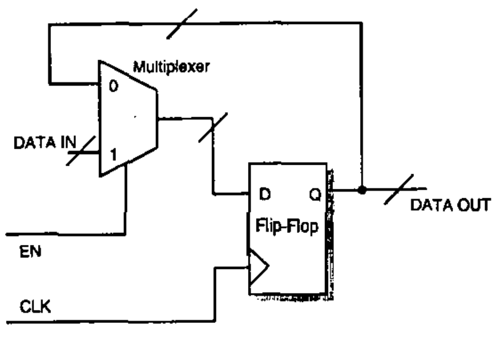

Fig. 1.

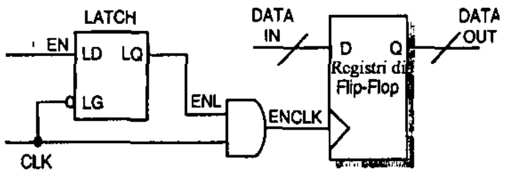

Fig. 2.

TABLE II

```
Module gated_clock (D, clock, c);
input D, clock;
output c;
wire clock_not = ~clock;
reg Q, c;
always @(clock_not or D)
 begin
   if (clock_not)  Q = D;
   c = clock & Q;
 end
endmodule
```

It is worth noting that, in order to prevent glitches on the clock network, for each enable signals we are introducing a latch, which certainly contributes to clock energy consumption. Thus this methodology is convenient at least for registers with a number of bits equal or higher than three.

To implement the gated clock we can define a module we named gated_clock as shown in Table II, we define clk_icw1 a wire variable, then we modify the register description according to Table III.

TABLE III

```
Always @(posedge clk_icw1)
Begin:ICW1
  if(ireset_ == 1'b0)
     icw1 <= 8'b00010001;
  else
     icw1 <= din;
end
```

When we have registers with a number of bits lower than three, or we have registers with different enable signals for set of bits, we can still save power adopting a different strategy. In particular, we directly control the clock through the enable signal as shown in Fig. 3.

Unlike the previous case, in order to avoid glitch on the gated clock signal this configuration requires that the enable signal must be stable before the lower clock edge. For this reasons the strategy should be used with care and avoided in the critical paths.

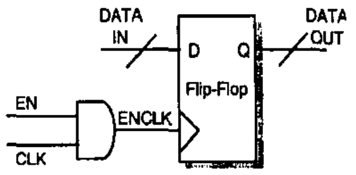

Fig. 3.

A typical Verilog code is reported in Table IV, where an eight bits register with eight different enable signals is considered.

TABLE IV

```
Always @(posedge clk)
Begin
   For (j = 0; j < 8; j = j + 1)
      Begin
         if (IsrReset[j] == 1'b1)
            isr[j] <= 1'b0;
         else
            if (IsrSet[j] == 1'b1)
               isr[j] <= 1'b1;
      end
end
```

It is apparent that implementation of the gated clock strategy in Fig. 3 for the register of Table IV leads to eight flip-flops with different clock signal. Two solutions can be identified to implement the strategy. We can leave asynchronous the reset signal applying an "and" operation between the clock and the enable signal, as shown in Table V for the first flip-flop. Otherwise we can keep synchronous the reset, including it on the gated clock signal, as shown for the first flip-flop in Table VI.

This second solution is more robust because it is able to prevent any unwanted glitch on the gated clock network.

Finally, for flip-flops and registers without an enable signal, we can properly modify the strategy proposed in [9], as shown in Fig. 4. It is based on the synchronization of the flip-flop only when the input value is changed with respect to the actual output value; otherwise the clock of the flip-flop is kept disabled. The introduction of the XOR and NOR gates modify the propagation delay of the flip-flop. In this way, designer has to verify the time behavior of the final circuit, and the strategy should be avoided in the critical paths.

TABLE V

```
wire clk_IsrSet_0 = clk & IsrSet[0];
always @(posedge clk_IsrSet_0 or
         posedge Isr_Reset[0])
begin
   if (IsrReset[0] == 1'b1)
      isr[0] <= 1'b0;
   else
      isr[0] <= 1'b1;
end
```

TABLE VI

```
wire clk_Isr_0 = clk & (IsrSet[0] |
                        IsrReset[0]);

always @(posedge clk_Isr_0)
begin
   if (IsrReset[0] == 1'b1)
      isr[0] <= 1'b0;
   else  if (IsrSet[0] == 1'b1)
            isr[0] <= 1'b1;
end
```

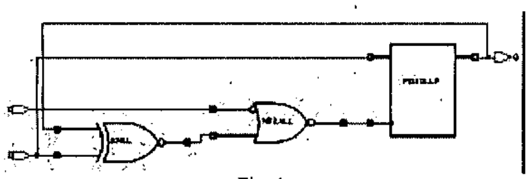
Fig. 4.

3. VALIDATION AND COMPARISON

To verify the strength of the gated clock strategies we implement them on a PIC [11]. It has the purpose of managing the interrupt signals in the x86 microprocessors. The PIC receives the interrupt signal from peripherals and choices the request with the higher priority, then it interrupt the CPU. The CPU, after the end of the current instruction, sends to the PIC the acknowledgment of interrupt. The PIC is able to manage eight levels of interrupts, which can be expanded to 64 adopting a master/slave configuration. A block diagram with input and output signals is shown in Fig. 5, where all the input and output signals have the "i" and "o" prefix, the bi-directional signals do not have prefix, and all the active low signals have the suffix "_".

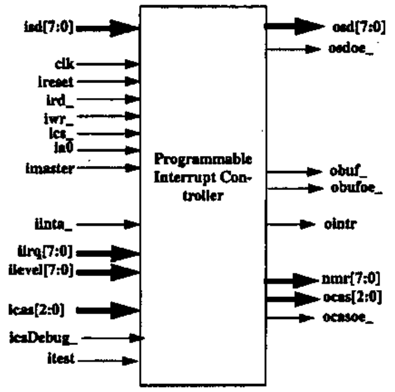

Fig. 5.

The compatibility with the standard block, Intel 8259, and the connection with the other device are represented in Fig. 6 [11].

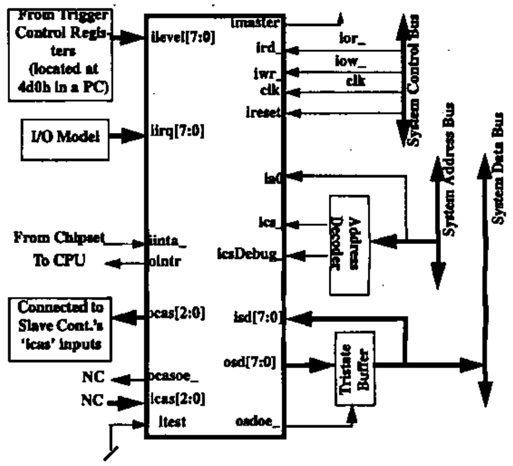

Fig. 6.

The gated clock strategies were implemented both manually following all the three strategies discussed and through the CAD, which implement only the one regarding register with an enable signal and more than two flip-flops. Then the original PIC and the other two adopting the gated clock approach have been implemented with the Synopsys Design Compiler and the HCMOS8D CORELIB cell library [12].

The fully implementation of the three gated clock strategies leads to modify 139 flip-flops on the whole of the 177 flip-flops instantiated in the original PIC (i.e., the 78% of flip-flops). In particular,

- 88 using the strategy in Fig. 2;
- 10 using the strategy in Fig. 3;
- 41 using the strategy in Fig. 4.

To verify the functionality of the designed PICs and the associated power figures, an extensive set of logic simulations have been run [13] at the gate level.

The following test-benches inside the microprocessor environment are used:

- ipc1_1.c: control PIC's registers during read and write operations;
- ipc2_1.c: verify the 0 interrupt (assigned to the system timer);
- ipc3_1.c: verify one of the interrupts, except for the zero, assigned to the ISA bus;
- ipc3_4.c: verify one of the A, B, C, D interrupts assigned to the PCI bus;
- ipc3_5.c: verify the functionality of AEOI options;
- irqRoute_1.c: verify the rotation of interrupt requests.

TABLE VII

	Original	Gated Clock Aut. Impl.	Gated Clock Manually impl.
AREA	**26767 μm^2**	**29024 μm^2**	**25436 μm^2**
Ipc1_1	360 µW	262 µW (27%)	201 µW (44%)
Ipc2_1	361 µW	279 µW (23%)	199 µW (45%)
Ipc3_1	361 µW	279 µW (23%)	201 µW (44%)
Ipc3_4	361 µW	264 µW (23%)	183 µW (49%)
Ipc3_5	358 µW	282 µW (21%)	200 µW (44%)
IrqRoute_	359 µW	283 µW (21%)	205 µW (43%)

The results, summarized in Table VII, show that the PIC with gated clock implemented by CAD save a power consumption between 21% and 27%, while the PIC in which all the three gated clock strategies have been implemented save an amount of power consumption between 43% and 49%. Hence, it means that implementation of all the three strategies, discussed so far, allows to double the power saving we can achieve. Moreover, the silicon area, also reported in Table VII, is smaller for the custom implementation and larger for designs automatically implementing the gated clock. The reduction of silicon area has a further benefit in term of stand-by power consumption. Indeed, it gives power consumption, due to leakage current, of about 279nW, a 10% lower than that in the other two implementations.

It is worth noting that in the original PIC all the tests show almost the same power consumption, while for the other two designs the tests have a 10% variation of power consumption since the gated clock signal activity changes for each test.

4. CONCLUSION

Efficiency in terms of power reduction of the gated clock approach was verified and compared. The block used to test the approach is a PIC designed using the HCMOS8D CORELIB cell library.

Gated clock has been implemented both through a custom design and through an automatic application with the tool Synopsys Design Compiler. In the custom design, in order to minimize the power consumption of the gated clock approach, three different implementation strategies were combined on three different classes of circuits:

- register with more than three flip-flops with a common enable signal;
- register with a number of flip-flops lower than three or with flip-flops enabled by different signals;
- register without an enable signal.

The Synopsys Power Compiler tool implements the gated clock only in the first register class exactly as done through the custom design.

Simulations results show that automatic implementation leads to a power consumption reduction of about 25% while custom design allows a power consumption reduction up to 45 % together with a lower silicon area.

In conclusion we have shown that gated clock is an efficient approach to reduce power consumption, provided that it is applied in all the three register classes already considered. Moreover, a further development is required in the Design Compiler in order to have a fully exploitation of the strength of the gated power approach. In fact, application of the gated clock only on the register with more than three flip-flops, having an enable signal, allows a power reduction lower of a factor two of the amount that can be saved through the application of all the strategies considered.

5. REFERENCES

[1] A. Chandrakasan, R. Brodersen, *Low Power Digital CMOS Design*, Kluwer Academic Publisher, 1995.

[2] J. Rabaey, M. Pedram, *Low Power Design Methodologies*, Kluwer Academic Publisher, 1996.

[3] K. Roy, S. Prasad, *Low-Power CMOS VLSI Circuit Design*, Wiley Interscience, 2000.

[4] A. Chandrakasan, W. Bowhill, F. Fox, (Eds.), *Design of High-Performance Microprocessor Circuits*, IEEE Press, 2001.

[5] L. Benini, A. Bogliolo, G. De Micheli, "A Survey of Design Techniques for System-Level Dynamic Power Management", *IEEE Transactions on VLSI Systems*, Vol. 8, No. 3, pp. 299-316, June 2000.

[6] L. Benini, G. De Micheli e E. Macii, "Designing Low-Power Circuits: Practical Recipes," *IEEE Circuits and Systems Magazine*, Vol. 1, No. 1, pp. 6-25, April 2001.

[7] L. Benini, P. Siegel, G. De Micheli, "Automatic Synthesis of Gated Clocks for Power Reduction in Sequential Circuits," *IEEE Design and Test of Computers*, Vol. 11, No. 4, pp. 32-40, December 1994.

[8] L. Benini, G. De Micheli, "Automatic Synthesis of Low-Power Gated-Clocks Finite-State Machines," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 15, No. 6, pp. 630-643, June 1996.

[9] T. Lang, E. Musoll e J. Cortadella, "Individual Flip-Flops with Gated Clocks for Low Power Datapaths," *IEEE Trans. on Circuits and Systems – II: Analog and Digital Signal Processing*, Vol. 44, No. 6, pp. 507-516, June 1997.

[10] Q. Wu, M. Pedram, X. Wu, "Clock Gating and Its application to Low Power Design of sequential Circuits," *IEEE Trans. on Circuits and Systems – I: Fund. Theory and Appl.*, Vol. 47, No. 103, pp. 415-420, March 2000.

[11] Programmable Interrupt Controller Intel 8259, *Microsystems Components Handbook, Peripherals Volume I*, Intel, 1996.

[12] *Design Compiler Reference Manual*, Synopsys, V.11, 2000.

[13] *Power Compiler Reference Manual*, Synopsys, V. 11, 2000.

[14] B. Chen e I. Nedelchev, "Power Compiler: A Gate Level Power Optimization and Synthesis System," *IEEE International Conference on Computer Design*, Austin, Texas, October 1997.