Buletinul Ştiinţific al Universităţii "Politehnica" din Timişoara

Seria ELECTRONICĂ şi TELECOMUNICAŢII
TRANSACTIONS on ELECTRONICS and COMMUNICATIONS

Tom 47(61), Fascicola 1-2, 2002

A Study Regarding the Implementation with VHDL of a Multiple Clock Gating Scheme for Low Power RTL Design

Alin Tisan[1], Stefan Oniga[2], Daniel Mic[1], Ciprian Gavrincea[1]

Abstract – In this paper we propose an algorithm, implemented with VHDL language in RTL design, capable of reorganizing the flip-flops from within a circuit in order to reduce the power consumption through optimal clock distribution. Practically in the end, starting from this algorithm, we will model the clock behavior in a sequential circuit. Experimental results show that these designs have ideal logic functionality with lower power dissipation compared to traditional designs

Keywords: VHDL, FPGA, power optimization

I. INTRODUCTION

With the increasing size and complexity of Field Programmable Gate Arrays (FPGA), clock distribution has become an important issue. It has become increasingly difficult to maintain high clock rates as the complexity and size of circuits implemented constantly grow. Clock managers are being incorporated in FPGAs [1, 2, 3] to solve clock distribution problems, and to improve their flexibility and functionality. Clock managers can be used in many FPGA applications. Using the frequency multiplication and division features, designers are able to manipulate the clock rates on- and off-chip, and use it for clock alignment among subsystems that use different clock rates.

An important problem of the modern digital systems is the power dissipation growing due to the increasing of clock frequencies and the decreasing of the feature sizes. For the reason that, the clock distribution network consumes large percentage (20% - 50%) of the power consumed by these systems, we will focus on trying to reduce the dynamic power consumed by the module clocked, like flip-flops, through selectively stopping the clock at these flip-flops that do not need it for changing his output value. Basically with the algorithm proposed we will group these flip-flops that have the maximum common transitions, the flip-flops with the very similar activity pattern.

The power consumed by complementary metal–oxide–semi-conductor (CMOS) circuits consists [2] of two components: dynamic and static power. The static power is largely determined by the technology. In this paper, we only consider minimizing the dynamic power. The dynamic power consumed by a module clocked at a frequency, φ is given by, $\varphi V^2 C_T$ where V is the supply voltage and C_T is the total load capacitance on the circuit. If a circuit switches A times per clock cycle, then its power consumption is given by, $P = A\varphi V^2 C_T$ where A is called the *circuit activity*. To minimize the power consumed by a CMOS synchronous system, we would in turn like to minimize its total activity. In a normal clock tree, the clock signal arrives regularly at all of the clock sinks, which means $A = 1$. Suppose that we know the times at which the clocked sinks must be active. We refer to the set of active/idle times for the module as *activity patterns*. They can be obtained by simulation of the design at the behavioral level. The clock signal must be supplied to the modules only during their active times. If the clock signal is gated such that it is only delivered during these times we can reduce the total power consumed by the clock and by the modules them-selves. We call a clock tree thus constructed an *activity-driven clock tree*. In this paper, we address the problem of minimizing the power consumption of a synchronous system by minimizing its activity through the use of an activity-driven clock tree.

The circuits designed by describing them in a behavioral language like VHDL, are synthesized (using a synthesizer like Leonardo Spectrum) into a gate level VHDL netlist. The activity pattern of all flip-flops from the RTL design was got from a vector file format obtained after a simulation of the VHDL netlist in the most common functionally conditions of the circuit. The choosing of the input's test vectors is very important because it will persuade the grouping into the unique best mode of the flip-flops.

1.Teaching Assistent
2. Lecturer Eng.
Universitatea de Nord Baia Mare Facultatea de Inginerie, DepartamentulElectrotehnica, str. V. Babes nr 62A
e-mail atisan@ubm.ro

II. THE CIRCUIT TRANSFORMATION

As described in the introduction the basic idea of the transformation is to switch the clock of flip-flops that take their own data. With this algorithm we will try to give to the flip-flop a "clock" only when it need too, to be precise only when the input data is changing. I have named this special clock: ideal clock.

Clock gating is a transformation that is performed on a synchronous digital circuit. For each flip flop in the circuit, the hold condition is determined, i.e. the condition under which the value that is clocked into the flip flop is identical to its current value. Under this condition, a transition on the clock input of the flip flop can be suppressed without changing the circuit's behavior. Such a modified clock is called a gated clock.
Since gating a clock involves a latch, and thus area overhead and extra power dissipation, flip-flops with similar hold conditions are grouped to be clocked by the same gated clock. Power reduction is achieved if a gated clock governs enough flip flops with a combined hold condition that is often true.

The tested project has 12 flip-flops (flip-flops) and was choused because of the simplicity of pursuing the results.
After functional simulation we have got the waveform (test vectors) that we have exported them as ASCII format. Using a program written in Pascal we turned the previous file into another one that describes each flip-flops activity pattern, means idle (0) or active period (1) of flip-flops (FF), fig.1.

	Transitions									
Ck	FF1	FF2	FF3	FF4	FF5	FF6	FF7	FF8	FF9	FF10
1	0	1	1	0	1	1	1	0	0	1
2	1	1	1	1	0	0	1	1	1	1
3	1	0	0	0	0	0	1	0	1	1
4	0	0	1	1	1	1	0	0	1	0
5	1	1	0	1	1	1	0	1	1	1
6	0	1	0	1	1	1	1	1	0	1
7	1	0	1	0	0	1	1	0	1	0
8	1	1	0	0	1	0	1	1	1	0
9	1	1	1	1	0	0	1	1	0	1
10	1	1	0	0	1	0	0	0	1	0

Fig.1. Activity pattern of the flip-flops

Also we have tried to estimate the activity of the flip-flops (A_G) depending on the total number of flip-flops (N_{FG}), the number of flip-flops without activity (N_{F0}), the number of the flip-flops from the group considered (g) and the number of the given clocks (no_clk), and the activity before the grouping (A_T).

$$A_T = no\_clk + or\_col \times N_{FT} \qquad (1)$$

$$A_G = no\_clk + \sum_{i=1}^{\frac{N_{FG}}{g}} \left[no\_clk + g \times or\_col\left(\frac{N_{FF}}{g}\right)i \right] \qquad (2)$$

$$A_G = \left(\frac{N_{FG}}{g} + 1\right) no\_clk + \sum_{i=1}^{\frac{N_{FG}}{g}} \left(g \times or\_col\left(\frac{N_{FF}}{g}\right)i \right) \quad (3),$$

where $or\_col(FF)$ mean OR function applied to those columns that corresponds to FF.
Based on the last file the flip-flops will be grouped taking in account the activity patterns similarities between flip-flops (the number of common transitions and idle states too). The grouping technique is binary tree algorithm based. This algorithm starts by grouping the first most alike two flip-flops, then the next two, from the left ones and so on.

So, that we get the first level of the tree. In the next stage we have to put together each two flip-flops group from the first level and, so on until there are no possible groups to do. For this stage we have made a soft in Pascal that compare and make the groups.

After we have created the tree we have to interfere in the netlist file for stopping unnecessary clocks (that is called clock gating). Starting from the EDIF format of the netlist we turned it into a VHDL format and we inserted the clock gating circuits that for the first level looks as is shown in the next figure:

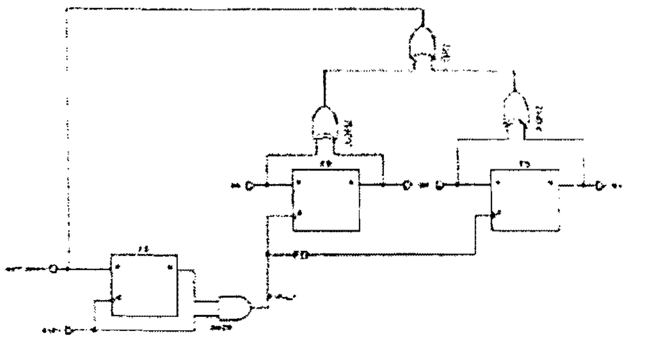

Fig.2 Clock gating circuits module

This module is composed of two XOR gates (each one being applied to input and output of one flip-flop), one OR gate applied to the outputs of the XOR gates, one flip-flop, and one AND gate. The XOR gate will decides if the flip-flop needs or no a clock to change it output data. Practically the XOR gate will compare the data from output of flip-flop with the input data. When those data are different then the XOR gate will point this to the OR gate that will totalize all this "clocks needs" and together with AND gate will give to the flip-flop considered an "ideal clock". One of the input of the AND gate will be the "initial clock" and the other will be the "command" for clock needing. The role of the flip-flop from the module is to assurance the synchronization of the new clock with the old one.
With the VHDL format for the module of binary tree:

L1G1_OR : work.components.OR2 port map(O => L1G1_OR_OUT, I0 => flip-flops 6_AND_OUT, I1 => flip-flops 9_AND_OUT);

```
L1G1_FD : work.components.FD port map(
D => L1G1_OR_OUT, C => IN_CLK_BUF, Q =>
L1G1_FD_OUT);
    L1G1_AND : work.components.AND2 port map(
I0 => L1G1_FD_OUT, I1 => IN_CLK_BUF, O =>
L1G1_CLK_OUT);
    FF6_XOR : work.components.XOR2 port map(
O => FF6_XOR_OUT, I0 => IR0_COMP_nx144, I1
=> RID_2);
    FF6_AND : work.components.AND2 port map(
O => FF6_AND_OUT, I0 => FF6_XOR_OUT, I1 =>
clock_id);
    FF9_XOR : work.components.XOR2 port map(
O => FF9_XOR_OUT, I0 => IFD8_Q_7, I1 =>
IN_PORT_7_int);
```

For providing the clocks to the upper levels we have inserted circuits as those shown in the next figure:

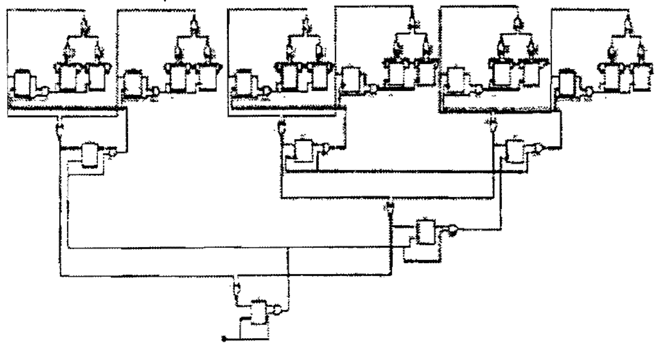

Fig.3 Clock gating circuit module applied to three levels of the binary tree

We also tried to find a grouping algorithm, a possible one is to analyze all possible the flip-flops combinations and to find the group with the maximum common transitions. In this group could be possible to include others flip-flops s that have at least the same transitions eventually some more 10% this way some flip-flops s will get unused clocks. But this seems to be a very tedious method because of the large combinations number:

$$\sum_{k=2}^{n-1} \frac{n!}{k!(n-k)!} \qquad (4)$$

In our case for 12 flip-flops, there are 4082 possible combinations, but for 100 flip-flops there are $1.5 \cdot 10^{30}$ possible combinations.

Even if, each flip-flops will work getting the ideal clock, other flip-flops are used to share the clock signal for each pair of flip-flops and the clock saving disappear.

The circuits are described in VHDL.

After we introduced the clock gating circuits we have checked again the tested circuit using a test bench made after functional simulation. The circuit behavior was the same.

Another soft was developed for automate insertion of the clock gating circuits.

The power optimizer software resolves the power dissipation in the flip–flops by rerouting the clock signal to the flip-flops in order to give them the clock only when needed (i.e. when the input data of the flip-flop changes).

Before you start the software, you need the following two files: the VHDL file containing the structural description of your circuit and the activity pattern file, which contains the activity patterns of your flip-flops.

As you can see in fig. 4, the program has three main areas

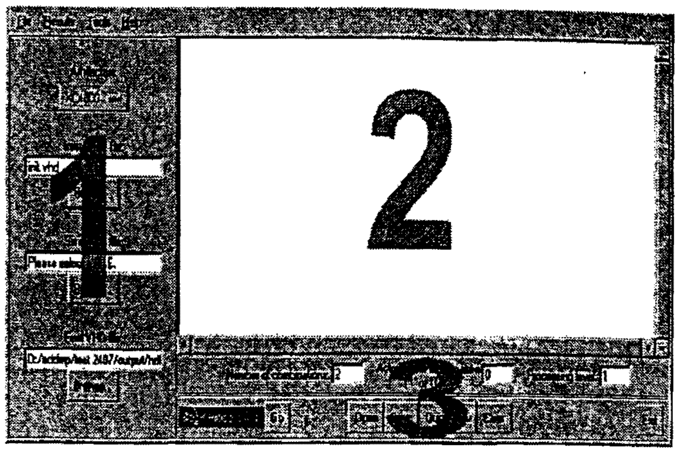

Fig.4 GUI of the program

No. 1 is the main entry area where you can specify:

a) the technology, by clicking on the technology selection button;
b) the initial VHDL file;
c) the activity pattern file (obtained as a result of the simulation, see previous explanations);
d) the name and location of the output (optimized VHDL file)

The area no. 2 is the editor window where you can see the input files and the results.

After the simulation has ended, beside the optimized VHDL file, you can see as results of the optimization the following report files (Reports Menu):

- Combination method: you can see how the flip-flops were grouped;
- Flip – flop entries: the flip-flops that appear in the initial VHDL file;
- Signals: the signals from the optimized clock tree;
- Components: the optimized clock tree components;
- Final VHDL file: the final optimized VHDL file;
- Final VHDL file: the final optimized VHDL file;
- Activity: misc. reports concerning the optimization;
- All Final Results: displays all the report files one after the other (takes all the result files from the working directory).

As for the other buttons you can see in area no. 3, you will find the explanation of their functions in the following picture:

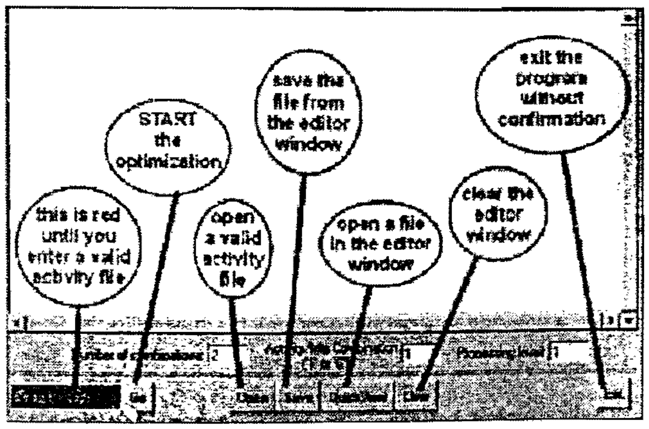
this is red until you enter a valid activity file
START the optimization
open a valid activity file
save the file from the editor window
open a file in the editor window
clear the editor window
exit the program without confirmation

Fig.5 Buttons explanation of the GUI

Charts provided in Xilinx data sheets provide dynamic power consumption values for typical design elements (for example, the power consumption of one XC4000E CLB flip-flop driving its neighbor and nine lines of interconnect is 0.2mW per million transitions per second); these can be used to derive useful power consumption estimates, but also means that we try to reduce 20%-50% that is 0.04-0.1 mW for one flip-flop. That explain the result that we get for our experiment made up from 12 flip-flops before optimization and 23 flip-flops after that. Here is the power consumption report for the original project at 10 MHz and 100 MHz:

III. CONCLUSION AND FURTHER WORK

In this paper we have proposed an approach for the construction of activity driven clock tree, with the objective of minimizing power consumption. We have developed algorithms that solve the problems of clock tree construction and gate insertion into the clock tree, while minimizing power consumption.

If in the initial circuit the flip-flops (12) carried out 8088 transitions, in the new circuit where we applied clock gating circuits, the flip-flops (now 23 flip-flops, more than the first time) carried out 4186 transitions that mean to save almost 50%. The initial 12 flip-flops have got the ideal clocks (1373) the rest of the transitions are carried out by the additional flip-flops (11) used for clock gating.
In the next stage we applied the same principle but this time each group will contain three flip-flops. From the functional point of view the results was the same but, "the saved clock" was grater than before: it was used 18 flip-flops with 2934 transitions. During next stage we will reapply our algorithms over a bigger project. We have to find which is the optimum flip-flops number that has to be grouped. We also have to finish the soft that automates the grouping process with respect to the activity pattern obtained from test vectors indifferently by the flip-fiops number and groups' size.

REFERENCES

[1] E. Tellez, A. Farrah and M. Sarrafzadeh, *"Activity-driven clock design for low power circuits,"* in *Proc IEEE ICCAD*, San Jose, pp.62-65, Nov. 1995.
[2] Amir H. Farrahi, Chunhong Chen, Ankur Srivastava, Gustavo Téllez, and Majid Sarrafzadeh, *"Activity-Driven Clock Design"*, *IEEE Trans. Computer-Aided Design*, vol. 20. no. 6, pp 705, june 2001
[3] Dennis J.-H. Huang, Andrew B. Kahng and Chung-Wen Albert Tsao, *"On the Bounded-Skew Clock and Steiner Routing Problems"*, 32nd ACM/IEEE Design Automation Conference 1995.
[4] Ian Brynjolfson and Zeljko Zilic, *"FPGA Clock Management for Low Power"*, Proc. Of Int. Symposium on FPGAs, FPGA 2000, Monterrey, CA, Feb. 2000.
[5] E. G. Friedman, *"Clock Distribution Design in VLSI Circuits – an Overview"* Proceedings of IEEE International Symposium on Circuits and Systems. pp. 1475-1478, May 1993.
[6] E. G. Friedman (Ed.), *Clock Distribution Networks in VLSI Circuits and Systems,* 525 pp., Piscataway, New Jersey:IEEE Press, 1995.