AUTOMATED CONSTRAINTS GENERATION FOR HARDWARE SYNTHESIS IN FUNCTIONAL-THERMAL SIMULATIONS

Vlad MOLEAVIN[1], Ovidiu-George PROFIRESCU[2], Adrian JOITA[3], Marcel PROFIRESCU[4]

Lucrarea prezinta un algoritm pentru ajustarea automata a constrangerilor de sinteza bazat pe simularea functional-termica a circuitelor digitale. ***Sinteza*** *este procesul de creare al portilor logice pornind de la limbaje de descriere de nivel inalt cum ar fi Verilog, VHDL, SystemC, SystemVerilog.De asemenea numit si proces de compilare deoarece este similar cu procesul software de compilare.* ***Simularea functional-termica*** *a circuitelor digitale este un proces dual de simulare atat al comportarii termice cat si al celei logice.*

Keywords: models, thermal, simulation, synthesis, Verilog.

1. Introduction

This paper presents a mechanism to adjust and optimize timing and area constraints for hardware synthesis, based on results of functional-thermal mixed simulation [1], [2]. This is a useful instrument for making circuits more robust by adding extra timing constraints for areas with high temperature; alternatively, it can generate components replacement constraints so power devices to spread across wider area. The paper also gives an overview of two important operations performed in the process of integrated circuits design: hardware synthesis and functional simulation of resulted netlists. A commercial synthesis application is presented together with its mechanisms to define timing and area constraints.

The synthesis[5] applications have appeared as a response to the exponential grown of complexity in digital integrated circuits, as a try to automate schematics creation and analyze. Such applications read VHDL or Verilog[6] code [4], [5] and deliver schematic description ready to be physically implemented in hardware.

[1] Eng., Infineon Technologies, Bucharest, Romania, e-mail: vlad_moleavin@yahoo.com

[2] Eng., ON Semiconductor, Bucharest, Romania

[3] Eng., ROHM Semiconductor European Design Centre, Germany

[4] Eng., EDIL - Micro and Nanoelectronics R&D Centre of Excellence, University POLITEHNICA of Bucharest, Romania

[5] *Synopsys Design Compiler*, Synopsys Inc. [3]

[6] VHDL and Verilog are standardized hardware description languages used to describe functionality of digital circuits or modules. They offer programming languages features that make easy for humans to handle.

Following synthesis, “Place and Route” process [6] completes the design activity by producing geometrical description of components and connections between them. The hardware synthesis is an iterative process starting from a set of descriptions and constraints. Then, an automated process attempts to translate functional code into functional primitives then into the corresponding logic structures that are functionally equivalent to the input. Physical implementation always relies on a digital technology[7] as well as on constraints which target either time or area. The iterative process continues until all constraints are met. Usually, design engineers analyze and refine constraints to smooth the synthesis process or to tune constraints on some design parts. The functional-thermal simulation results provide useful information for synthesis and “Place and Route” closing another iterative loop to adjust timing/area constraints based on thermal behavior[8].

2. Hardware synthesis commands and functional-thermal simulation environment description

This section lists some useful synthesis commands for *Synopsys Design Compiler* and presents the functional-thermal simulation environment.

Synopsys Design Compiler commands:

- **Set_dont_touch**: this command preserves given objects implementation from further optimization during synthesis process. The given object is still analyzed but no modifications are performed on its current structure.
- **Set_input_delay**: this command describes signal timing behavior on a particular input or group of inputs as coming from outside the circuit. Both rising and falling edges delays are modeled.
- **Set_output_delay**: similar to s*et_input_delay* but targets outputs and refers to timing reserve eventually needed by connected input of another circuit to this port.
- **Set_false_path:** this command inhibits the given path or object from timing analysis but still is subject of synthesis process. It may be uses to relax timing constraints or to disable analysis on some known paths.
- **Set_clock_uncertainty:** this command models the digital clock behavior besides setting the clock period. It sets the clock rise and/or fall jitter due to different clock paths lengths.
- **Set_max_delay:** this command constraints a particular combinational path to be not longer than provided value.

[7] Hardware synthesis targets an implementation technology represented as a library of components and rules. This is similar to target a hardware platform when compiling software applications.

[8] Timing constraints targets delays on particular logical paths inside design. Constrains may attempt to shorten or extend the delays.

Following, this section presents the functional-thermal simulation environment and simulation process in the context of synthesis constraints generation. This environment is a pure digital simulation environment with annotated[9] netlist as DUT[10]. Additionally, another "circuit" called thermal net is simulated, modeling the thermal behavior, interacting with the DUT by exchanging thermal information. Fig. 1 shows the thermal net being modeled by a 2D thermal cell array. The place & route information is also necessary to link digital components to appropriate thermal net nodes (in Fig. 1 red arrows indicate positions connected with digital components while grayed arrows indicate nodes with no components linked). In this digital implementation, the thermal net is modeled as a set of heat discrete equations [2].

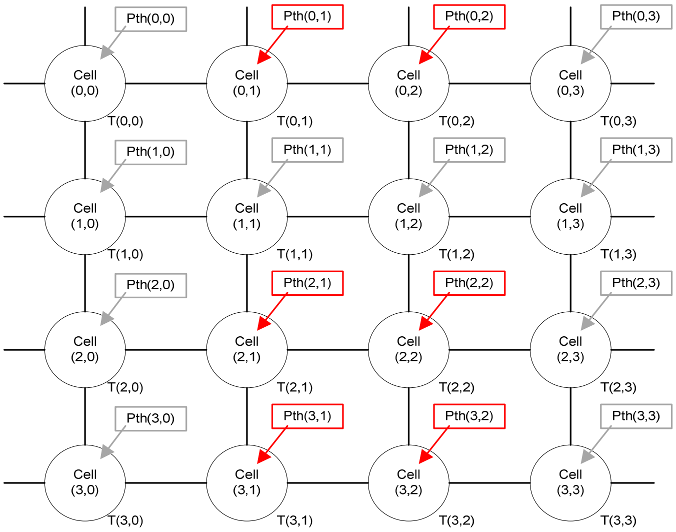

Fig. 1 - Functional view of the thermal net

Each digital component instantiated by the digital circuit has a functional model including thermal behavior; it has a thermal interface with temperature input and generated heat flow output. Fig. 2 describes the entire loop between the digital functionality and thermal behavior as modeled by this implementation [2]:

1. A digital component output toggle triggers heat quantum generation in thermal model (blue 'Transition' arrow).
2. From component thermal model, heat quantum flows into appropriate thermal net node, according to the Place & Route information (blue 'Heat flow' arrow).

[9] This is the netlist with Place and Route information.
[10] Device Under Test

3. The thermal net computes the temperature in every node as solution of heat equations set.
4. Each component’s thermal model sees the corresponding node temperature (green ‘’Temperature’ arrow) and adjusts the propagation delay functional parameter.
5. The loop is closed in the digital component behavioral model by using the adjusted propagation delay further in simulation (Green ‘Propagation delay’ arrow).

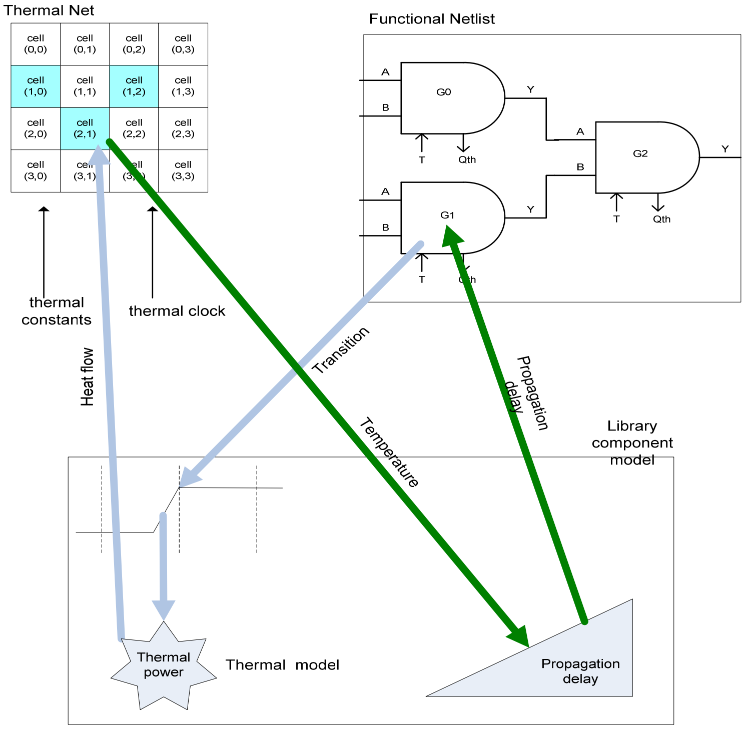

Fig. 2 - Thermal model of digital components

The main objectives for this environment are:

- Individual thermal model for each digital component; each component has two views, one functional and one thermal, as described above.
- Continuous interaction between functional and thermal behaviors by exchanging temperature and heat flow; local temperature value is provided by the thermal net (from geometrical position) to the functional model of the component instance which adjusts its propagation delay; simultaneously, the component pumps heat quanta on each outputs transition into the thermal net which, in turn, recalculates the temperature

distribution over the circuit area. This closes a behavioral loop between the two views so bringing the overall simulation closer to real function.

Other requirements for the mixed simulation environment are: usage of already standardized models, as few as possible modifications to the existing Digital environment, netlist DUT and methodology. Also, memory usage and computing power overhead inserted due to added thermal behavior should be kept low. Fig. 3 shows the process flow during functional-thermal simulation.

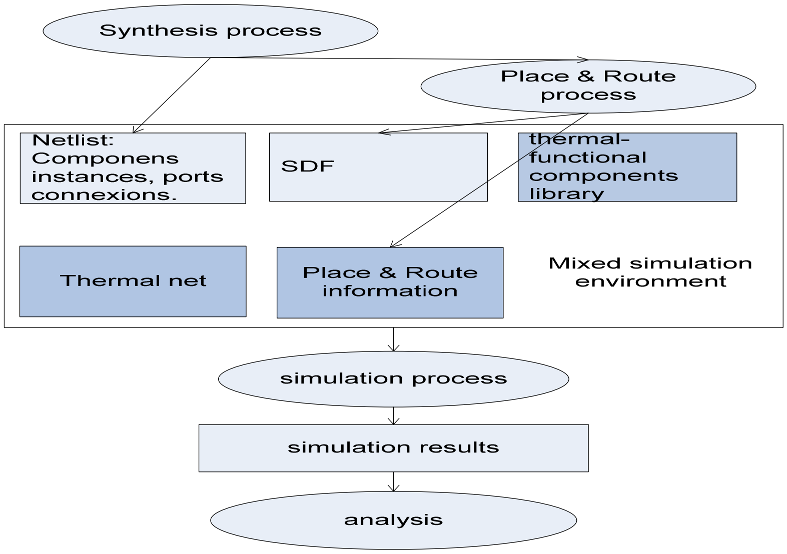
Synthesis process
Place & Route process
Netlist: Componens instances, ports connexions.
SDF
thermal-functional components library
Thermal net
Place & Route information
Mixed simulation environment
simulation process
simulation results
analysis

Fig. 3 – Functional-thermal simulation process flow

The simulation process of the unified functional-thermal behavior is similar to a regular annotated netlist simulation, with DUT from synthesis process and SDF from Place and Route process. Additionally, a VHDL model of the thermal net is loaded and connections for temperature and heat flow inserted as digital signals mapping assignments. Compilation process includes the DUT, functional-thermal components library, VHDL implementation for thermal net (previously generated and including geometrical information). Simulation step loads and run the DUT and the thermal net together. During the simulation the two models interacts and permanently exchange thermal information.

One of the main improvements by the functional-thermal simulation method is the modeling over time and space of thermal effects inside a digital circuit. During classical approach, temperature is a fixed constant, during the whole simulated time and across the entire circuit. Using the functional-thermal mixed approach, each component evolves independently from others; also, time dimension is added, allowing the thermal net and components to evolve as the real circuit behaves. High temperature areas contain many power components while low blue areas do not contain power devices. Heat flow is modeled by the thermal net, also visible in the referenced figure. Digital components models contain functional behavior. AND, OR, NOR gates as well as Flip-Flops or Latches are modeled with their logical interface and behavior. Additionally, thermal interface is present and the dependence of propagation delay on temperature is implemented; each digital transition generates a heat pulse routed into the thermal net via thermal interface. This approach keeps the simulated behavior more close to the real circuit. It is possible to catch thermal runaways or other functional glitches, as the components heat-up and combinational paths delays may become too long.

Following, the paragraph presents a simulated circuit and thermal results as base for last section conclusions.

Fig. 4 describes the placement of the components of the digital DUT circuit. The circuit contains two identical NOT gate chains (top and bottom) with the same signal as data input. A group of high current Flip-Flops is placed around the bottom gates chain. During mixed simulation the high frequency toggling FFs heat up the area around making bottom combinational path “longer” than the top one due to the increase of individual propagation delays. Fig. 5 shows the temperature distribution of the simulated circuit in functional-thermal approach. As a consequence, the two combinational paths will not evolve constant, as in a pure digital simulation; due to temperature increase, bottom chain digital functionality will break.

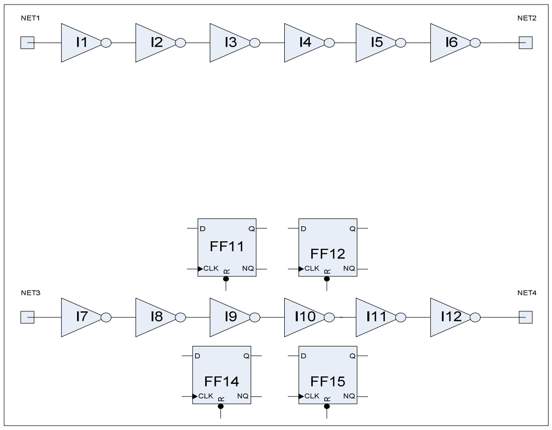

Fig. 4 - Placement of components

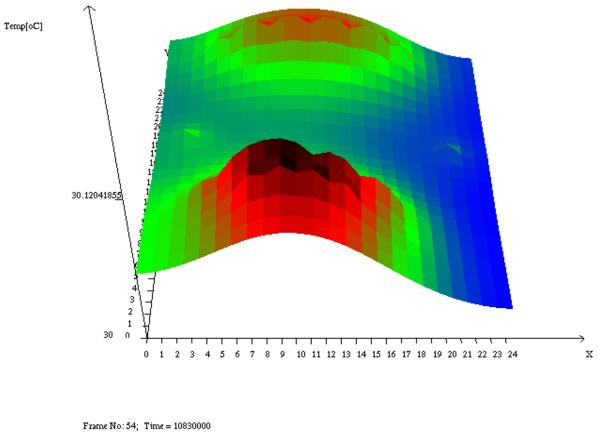

Fig. 5 - 3D temperature distribution snapshot during functional –thermal simulation of the presented circuit.

Such a simulation reveals bad placement for components or modules that have high current consumption. By grouping together in same small area many power components, leads the functional-thermal simulations to a fail or to show high temperature values in that area.

3. Functional-thermal simulation results analysis

The mixed simulation saves the thermal behavior information (both temperatures and heat flows) in form of log messages. It also saves the propagation delays of each component at each sample moment. This allows a further mechanism to find and sum the delays of a certain path at every moment and identify a potential fail by comparing with constrained value from synthesis. Analysis of the simulation results can be useful to identify modules or components with this problem and this information can be derived into constraints to an iterative synthesis process. The generated constraints can lower the maximum allowed delays in modules or in direction of re-placing the components. This is the proposed mechanism to automatically adjust the constraints and rerun the synthesis and *Place and Run* process; the target is to optimize the area and timing performance of resulting circuit in real operation. This mechanism should start with an existing set of synthesis constraints. Using **Synopsys** commands, the components list in combinational paths can be obtained; then, from simulation log the components delays can be extracted and summed at every time resulting into a vector of values per path, over time. Each value above a threshold issues a warning for the given path and generates a synthesis timing constraint. Having computing resources consumed reasonably, any number of paths can be analyzed, even all from a **Synopsys** generated “longest paths” list.

However, weak or contradicting requirements generation should be avoided as they can affect synthesis process completion.

4. Process Automation

The success of such algorithm resides in its automation capabilities. Iterative synthesis mixed with simulation steps over multiple scenarios may be required to obtain a correct and complete constraints list and ultimately a more efficient and robust design.

An efficient algorithm implementation should contain software tools for executing all the above described steps. Process running, log analyzing, information parsing, actual constraints generation and exception handling should be automated in such manner.

The algorithm may be first implemented into one of the programming languages supporting regular expressions. Perl, Python, C# can be good choices.
Special attention should be paid when analyzing and filtering constraints; this is to avoid unnecessary, contradicting constraints and infinite constraint loops. Good knowledge of tool usage is mandatory [7], [8].

5. Conclusions

In the context of exponential complexity grown of digital circuits, the presented algorithm helps to increase the efficiency of circuits resource allocation (timing, area); by identifying potential thermal problems, it helps to avoid them and so eliminates bugs and increase design robustness. The method supports redesign to relax thermal stress over circuit areas and provides better temperature estimation from early simulation phases.

REFERENCES

[1] K.*Hofmann, J.M.Karam, T.Niculiu, B.Courtois, M.Glesner, Vlad Moleavin*, “Modelling and simulation of thermal effects on the system level”, Therminic Conference, Grenoble, 1995.

[2] *Vlad Moleavin*, “Simularea avansata a sistemelor digitale integrate, PhD Thesis”, ETTI Faculty, UPB, Bucharest, Romania, 2007.

[3] Synopsys, “Synthesis Quick Reference”, Version X, USA, 2005.

[4] *P. J. Ashenden*, “The Designer’s Guide to VHDL”, pp.633-667, 3rd Edition, ELSEVIER, USA, 2008.

[5] *Z. Navabi*, “Verilog Digital System Design”, pp. 25-282, McGraw Hill, USA, 1999.

[6] *S. Kilts*, “Advanced FPGA Design: Architecture, Implementation and Optimization”, Wiley, 2006.

[7] *Himanshu Bhatnagar*, Advanced ASIC Chip Synthesis using Synopsys, Second Edition, Kluwer Academic Publishers, 2002.

[8] *E. Brunvand*, “Digital VLSI Chip Design with Cadence and Synopsys CAS Tools”, Addison Wesley, 2009.