





Confidential

	
<b>ICT-2009.3.2-248603-IP</b>	
<b>Modelling ,Control and Management of Thermal Effects in Circuits of the Future</b>	
	

	WP no.	Deliverable no.	Lead participant
	<b>WP3</b>	<b>D3.1.1.</b>	<b>BME</b>
<b>Specification on thermal models and structural design of the logi-thermal simulator</b>			
Prepared by	<b>G. Nagy, A. Timar</b>		
Issued by	<b>THERMINATOR Project Office</b>		
Document Number	<b>THERMINATOR/D3.1.1/v1</b>		
Dissemination Level	<i>Confidential</i>		
Date	<b>31/12/2010</b>		

© Copyright 2010-2013 STMicroelectronics, Infineon Technologies, NXP Semiconductors, ChipVision Design Systems , GRADIENT DESIGN AUTOMATION , MUNEDA, SYNOPSIS , BUDAPESTI MUSZAKI ES GAZDASAGTUDOMANYI EGYETEM , CSEM, FRAUNHOFER , IMEC, CEA-LETI, OFFIS, Politecnico di Torino, ALMA MATER STUDIORUM -Universita' DI Bologna

This document and the information contained herein may not be copied, used or disclosed in whole or in part outside of the consortium except with prior written permission of the partners listed above.

## **Document**

Title	Specification on thermal models and structural design of the logi-thermal simulator
Type	Deliverable CO
Ref	D3.1.1
Target version	V1_1
Current issue	V0_1
Status	Proposal
File	
Author(s)	Gergely Nagy (BME) András Timár (BME)
Reviewer(s)	
Approver(s)	
Approval date	
Release date	

## Distribution of the release

Dissemination level	CO
Distribution list	

## History

Rev.	DATE	Comment
0.1	31-12-2010	Initial version
1.0		Revised version, approved by all partners
1.1		Check and ship out

## References

- [1] V. Székely, A. Poppe, M. Rencz, M. Rosental, and T. Teszére, “Therman: a thermal simulation tool for ic chips, microstructures and pw boards,” *Microelectronics Reliability*, vol. 40, no. 3, pp. 517–524, 2000.
- [2] (2010, 19th April, 15:57) FireBolt and CircuitFire. [Online]. Available: <http://www.gradient-da.com>

*This page was intentionally left blank.*

## Contents

Document .....	2
Distribution of the release .....	3
References .....	3
1 Introduction .....	6
2 Description of the CellTherm logi-thermal simulator application .....	7
Introduction .....	7
RTL level simulation.....	7
Hot-spot detection and temperature map determination .....	7
Design flow .....	7
Power characterization .....	9
Preliminary timings.....	11
Logic simulation with SDF data.....	11
Analog simulation results.....	12
Combining temperature map and delay functions.....	13
Summary .....	13
3 The Logitherm simulator.....	15
The logic engine .....	15
The thermal engine.....	15

## 1 Introduction

Task 3.1. aims to create a new simulator engine that is capable of determining the thermal behaviour of digital systems at the gate level.

Due to the uneven event density over the chip surface, digital blocks normally experience time-variable temperature gradients, and due to the fact that digital gates show a dependence on temperature, a merely digital, gate-level simulation may strongly deviate from the actual simulation results when the surface temperature profile of the chip is considered in calculating the actual delays of the individual gates. Self-consistency between the thermal behaviour and the digital behaviour of the chip is maintained: Logic and thermal operation are traced together, hence the name: Logi-thermal simulation. The major application of such a simulation is to make sure that during timing analysis thermal effects are considered (thermal-aware signal integrity check). In other words, to make sure that the digital circuit will properly function under all allowed thermal conditions.

Two paths are being followed and thus two simulator engines are under development. One uses existing engines and glues them together to enable them to perform a logi-thermal simulation (this is called the *CellTherm* simulator engine). The other is a completely new engine that incorporates a digital and a thermal engine and performs the simulation internally (*Logitherm*).

Both of the simulators are worth creating for the following reasons. The first type is easier to integrate into any design-flow as it makes use of already familiar tools. Designers can adapt to it with ease and without any great effort. The second type on the other hand has the potential of becoming a very fast and efficient companion of the digital designer as the complete integration of the two types of simulators allow for a very efficient and fast realization.

The engines are operable and already yield results. They are still under constant improvement: the algorithms and the cooperation of the collaborating parts need to be fine-tuned. Both simulators are written with easy extendibility in mind. Their structure is modular and the interfaces between the elements are well defined. New blocks can easily be written and added to the system to complement or replace certain parts.

## 2 Description of the CellTherm logi-thermal simulator application

### Introduction

As minimum feature size is continuously shrinking and power density growing, it is inevitable to take thermal effects into account when designing digital integrated circuits and manufacturing them. We provide a novel methodology to determine the hot-spots on the digital integrated circuit's surface. Furthermore, our approach is able to calculate temperature map of a packaged semiconductor circuit excited by a logical simulator virtually. The thermal simulator can admit compact models of thermal devices, such as a heat-sink combined with a fan.

### RTL level simulation

The methodology presented takes the standard cells of a digital ASIC design as basic primitives rather than the contained transistors and wiring. This property clearly distinguishes it from today's electro-thermal simulators. The logic simulator drives these primitive cells with a user-defined stimuli and logic waveform results can be observed. By standard cells being the building blocks of the simulation, it is apparent that the thermal resolution is coarser than that of an electro-thermal simulator but it is also faster. However, with reasonable and deliberate cell simplification an optimal trade-off between simulation accuracy and duration can be achieved. For example, if the initial cell was further partitioned into smaller blocks, transistor-level electro-thermal accuracy could be approximated.

The presented method is based on the so called relaxation method, where two simulator engines—an electric and a thermal—are coupled together. The input of each simulator comes from the other simulator and they together produce the final simulation result.

### Hot-spot detection and temperature map determination

By calculating the temperature distribution of the circuit, local and global hot-spots can be detected. Hot-spot detection is crucial in today's ultra large scale integrated circuits as functional operation may completely fail due to unanticipated heating of devices at certain coordinates. With our presented methodology and application bundle, circuit designers can create temperature-aware IC designs where thermal issues emerge during the design phase rather than after manufacture.

The simulation stimuli resembles the real-world operation of the circuit thus the fully packaged design can be evaluated for defects or misfunctionality far before manufacture. The evaluation is feasible by using a thermal simulator that is capable of simulating the design together with an arbitrary compact thermal model of the package and its surroundings.

### Design flow

The presented methodology is based on standard tools such as Verilog hardware description language and its Programming Language Interface (PLI), thus it can be integrated into any logical simulation flow easily. The majority of today's logic simulators (both commercial and open-source licensed) support this language and interface.

The *CellTherm* simulation software developed at the Department of Electron Devices in BME, Hungary acts as a glue logic between the standard logic simulator and the thermal



simulator. The Therman solver engine [1] was used as the thermal simulator that was also developed in the Department. The Therman solver engine can be augmented with the compact thermal model of the IC package where the compact model is generated from thermal transient testing measurements.

The glue logic interface application controls communication between the logical and the thermal simulator engines and displays the results in real-time. Real-time means that the simulation results get displayed right after each simulation timestep, thus providing the user immediate graphical feedback of the result. The design flow of the logi-thermal simulation is shown in **Figure 1**.

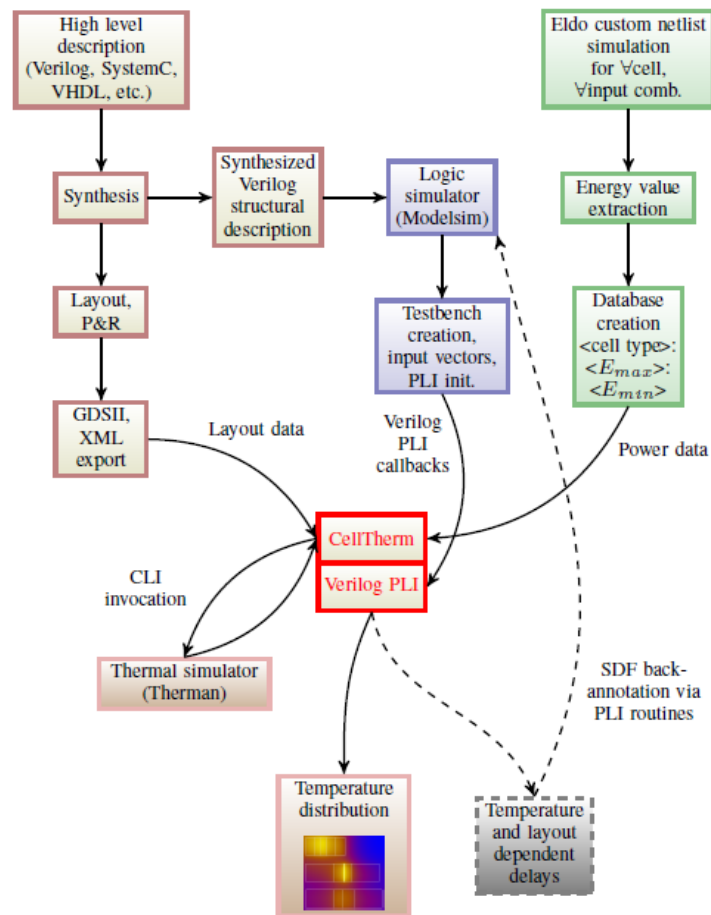


Figure 1 Logi-thermal design flow

The logi-thermal design flow starts with two concurrent processes. On one hand, the IC designer team describes the logic behaviour of the application in a high level language like SystemC, SystemVerilog, SystemVHDL, etc. Then the process continues with an iterative simulation cycle where the majority of the logic and timing errors can be detected and the design can be corrected. After successful logic and timing simulation, synthesis takes place, where logical RTL gates get mapped to the physical gates of the chosen process design kit (PDK). This synthesized description can then be simulated for correct timing and the floorplan can be created. Simultaneously, the cell power characterization has to be done. Often the process design kit comes with Liberty or TLF files which contain data about the cells' timing, delay and power characteristics. Unfortunately, sometimes this is not the case. The PDK may be missing these files, or just one type of them, or the necessary power

characteristics for the cells. Therefore a rather simple method has been developed to obtain the power characteristics of the cells. The characterization process can be done independently of the high level design and synthesis. The synthesized HDL description is stimulated with a logic simulator with test vectors and switching activity from the results can be determined for each cell. From the Place&Routed floorplan, the switching activity and the power characteristics of the cells the *CellTherm* application can calculate the temperature distribution in the design. This is done concurrently with the logic simulation thus immediate graphical results show the evolved hot-spots.

**Figure 2** shows a screenshot of the *CellTherm* application after the first few nanoseconds of the simulation time where steadystate temperature distribution has not yet been achieved.

Temperature-dependent delays could also be calculated and back-annotated to the simulation via Standard Delay Format (SDF) files. Delay back-annotation can take place before each timestep of the simulator.

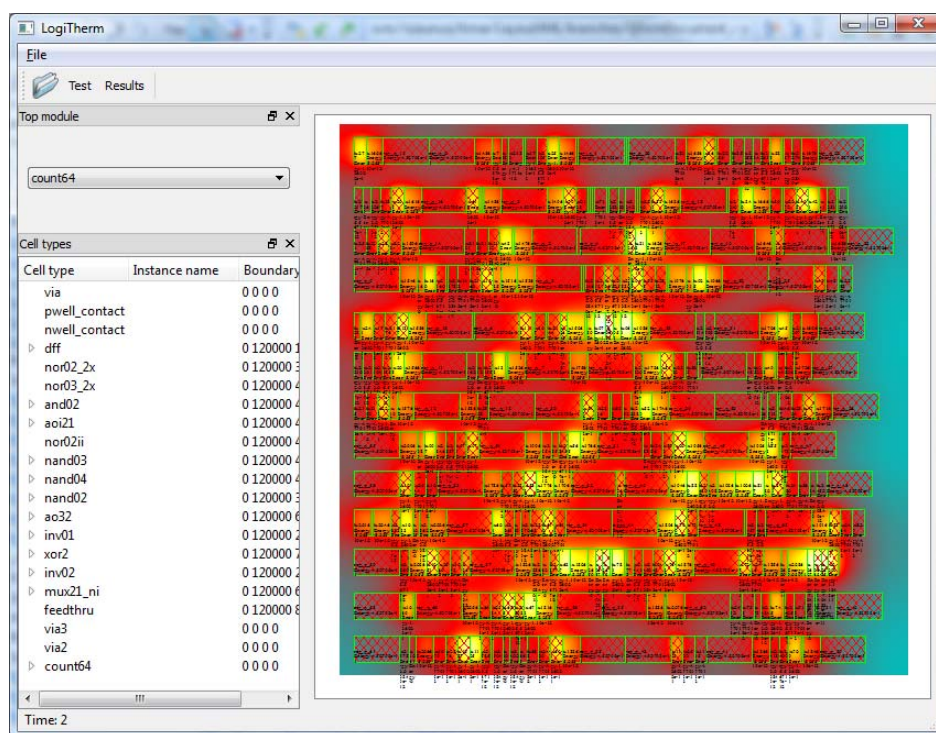


Figure 2 CellTherm application showing actual temperature map of simulated IC

## Power characterization

Power characterization is made using an analog simulator that runs transistor-level simulations of the cells in the library. The power characterization of the standard cells has to be done only once per process design kit. The methodology presented involves using the Eldo analog circuit simulator engine from Mentor Graphics. A custom Eldo netlist is generated according to the simulated cell library and all of the cells of the library are simulated with every possible input combination. As the methodology focuses mainly on simulating CMOS circuits, the great majority of the power consumption (and thus the heating) occurs when a switching activity takes place (hence the name “dynamic power”). During the initial analog simulation with Eldo, dissipated energies during every possible switching event in a cell are collected and stored for every cell type. The energies dissipated during a logic transition are calculated using:

$$\varepsilon = \int_{T_1}^{T_2} P(t) dt,$$

where  $P(t)$  is the total power consumption function of the cell after every possible logic transition. **Figure 3** shows a typical input voltage–power consumption characteristic. The energy dissipation on a logic transition is calculated with the following algorithm:

- 1) The vicinity of the logic transition has to be found where the great majority of the power consumption lies. The vicinity is found by the following steps:
  - a) Search for the maximum point of the  $P(t)$  power consumption function in the adjacent interval of  $\pm$ Period/2 around the logic transition. In CMOS circuits, this is an appropriate interval because the significant peaks of the power function  $P(t)$  are near the switching event. Far away from the switching events is only static consumption which is negligible in contrast to the consumption at logic transitions. Period here means the time between any two subsequent input logic transitions.
  - b) Find the time instances where the  $P(t)$  curve crosses 1% of the local maximum. These points are  $T_1$  and  $T_2$ .
- 2) Integrate the  $P(t)$  curve in the  $[T_1, T_2]$  interval. This value ( $\varepsilon$ ) is the average energy consumption belonging to that particular logic transition.

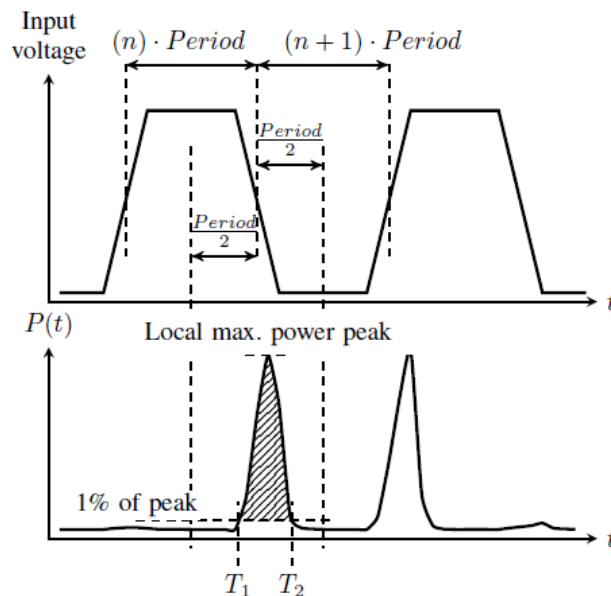


Figure 3 Calculation of dissipated energy per logic transition

The  $P(t)$  function is calculated with *Eldo*'s POW(Xinstance\_name) function which calculates real power dissipated in a cell and ignores power stored in capacitances. This behaviour is required because only real dissipated power must be taken into account during the thermal simulation. The measured power is exactly equivalent to the power dissipated by the transistors of the measured instance, thus it is the power that causes heating of the components in the cell.

A standard logic simulator (e.g. Modelsim from Mentor Graphics) can be complemented with a thermal simulator engine (e.g. Therman or FireBolt from Gradient Design Automation [2])

that calculates the temperature distribution of every cell instance in the layout according to the switching activity provided by the logic simulator. CellTherm conducts a thermal simulation on the actual layout with the actual power density based on the switching activity that is extracted from the digital simulator. This approach clearly realizes the relaxation method, where two distinct simulator engines (a logical and a thermal engine) produce results that the other utilizes.

### Preliminary timings

In the early phase of the design, when the behavioural description gets synthesized, pre-layout timing data can be approximated and the synthesizer software (e.g. LeonardoSpectrum from Mentor Graphics) can extract preliminary delay values from the design. The synthesizer software outputs the predicted post-synthesis pre-layout delay data into a Standard Delay Format (SDF) file, which can later be included in a logic simulation. The logic simulator can take these delay data as startup timing values and it is able to check against basic timing integrity issues. The SDF file contains not just delays of the individual cells of the design but setup and hold timing checks also. This way the simulator can send off an alert if timing requirements are not met. The preliminary delay and timing values are predicted values and do not take physical layout into account. Later on, after floorplanning and Place&Route, real timing data can be extracted from the design in SDF format. **Figure 4** shows a portion of a generated SDF file with the predicted delay and timing values.

```
(DELAYFILE
(SDFVERSION "2.0")
(DESIGN "count84")
(TIMESCALE 1 ns)
...
(CELL
(CELLTYPE "dffr")
(INSTANCE reg_output_0)
(DELAY
(ABSOLUTE
(PORT D (::0.00) (::0.00))
(PORT CLK (::0.00) (::0.00))
(PORT R (::0.00) (::0.00))
(IOPATH CLK Q (::0.49) (::0.53))
(IOPATH R Q (::0.00) (::0.58))
(IOPATH CLK QB (::0.39) (::0.30))
(IOPATH R QB (::0.44) (::0.00))))
(TIMINGCHECK
(SETUP D (posedge CLK) (0.47))
(HOLD D (posedge CLK) (0.06))))
```

Figure 4 Pre-layout SDF file

### Logic simulation with SDF data

The annotation of delay values takes place via the `$sdf_annotate()` Verilog construct in the simulator. A test case has been developed as a proof-of-concept to demonstrate setup and hold timing failure caused by applied SDF delay values. A basic flip-flop chain has been built in Verilog with the SDF delay data acquired from the synthesis and stimuli were intentionally corrupted in order to observe timing errors. The setup and hold timing constraints extracted from the SDF delay data are  $T_{\text{setup}} = 470$  ps and  $T_{\text{hold}} = 60$  ps, respectively. In Figure 5 the simulator sends off an error message that the setup timing requirements are not met.

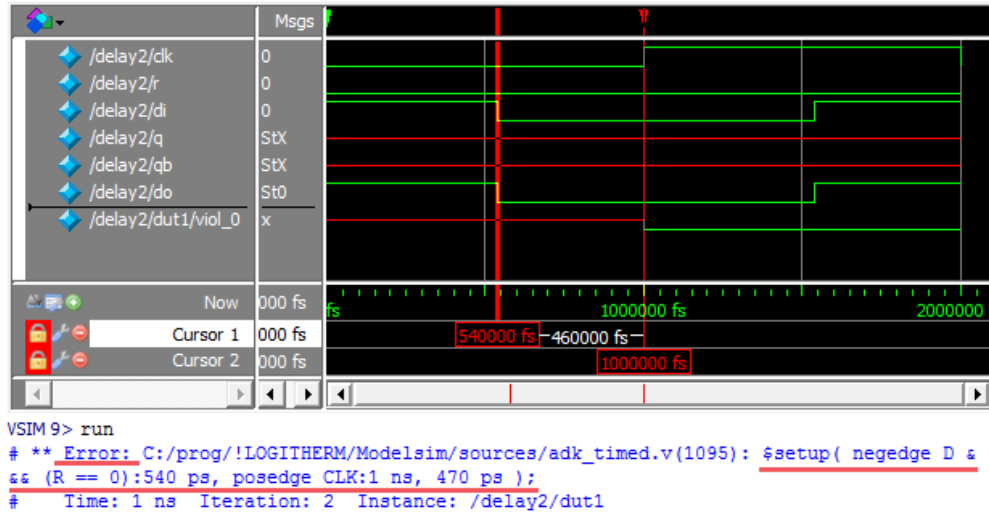


Figure 5 Setup timing error in ModelSiAnalog simulation results

Analog simulations were run with the Mentor Graphics's ELDO simulator to examine cell delays in function of temperature. Each cell of the standard cell library have been simulated in a test environment where only the cell in question was present and the ambient temperature has been swept in a wide range. The ambient temperature was swept from  $-40^{\circ}\text{C}$  -  $+90^{\circ}\text{C}$ . The temperature-dependence of delays could unequivocally be demonstrated. **Figure 6** shows the temperature-delay function of a D flip-flop circuit. The delay between *DATA* input's rising edge and the *CLK* rising edge has been set so a setup timing failure could be detected above a certain temperature. In **Figure 6** the curve runs from  $-40^{\circ}\text{C}$  to  $+40^{\circ}\text{C}$ . Over  $T > +40^{\circ}\text{C}$  delay between *CLK* and the *Q* output cannot be interpreted because the output did not change. This is clearly a setup timing error. The intrinsic delay of the cell increased so much with the temperature that the setup time requirement could not be satisfied, the input transition could not propagate to the output.

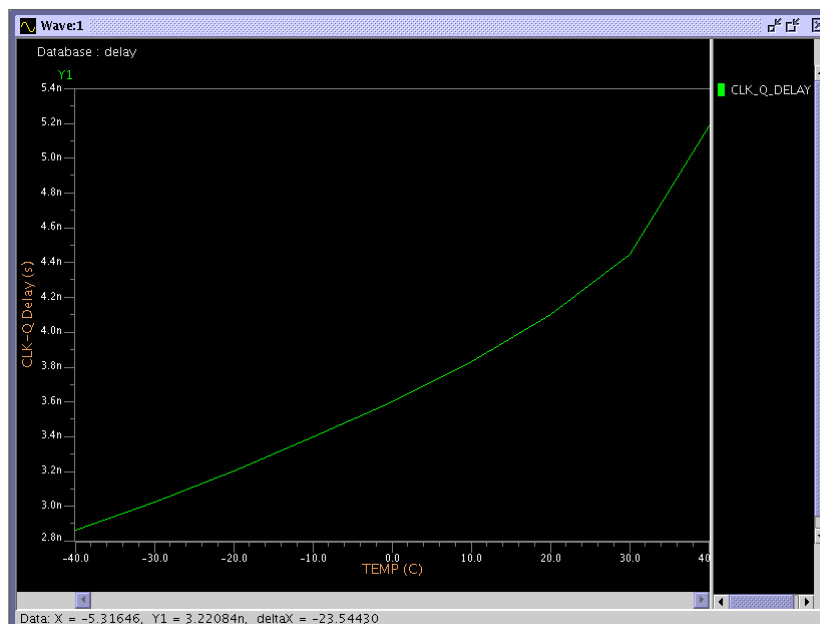


Figure 6 Delay vs. timing function of a D flip-flop cell

**Figure 7** shows the transient simulation results where the ambient temperature of the cell has been swept. With the temperature rising, the delay between *CLK* and *Q* rises up to a certain point, where setup timing requirements are no longer met. Over  $T > +40^{\circ}\text{C}$  the input signal change cannot propagate to the output thus leaving the output at a low logic level.

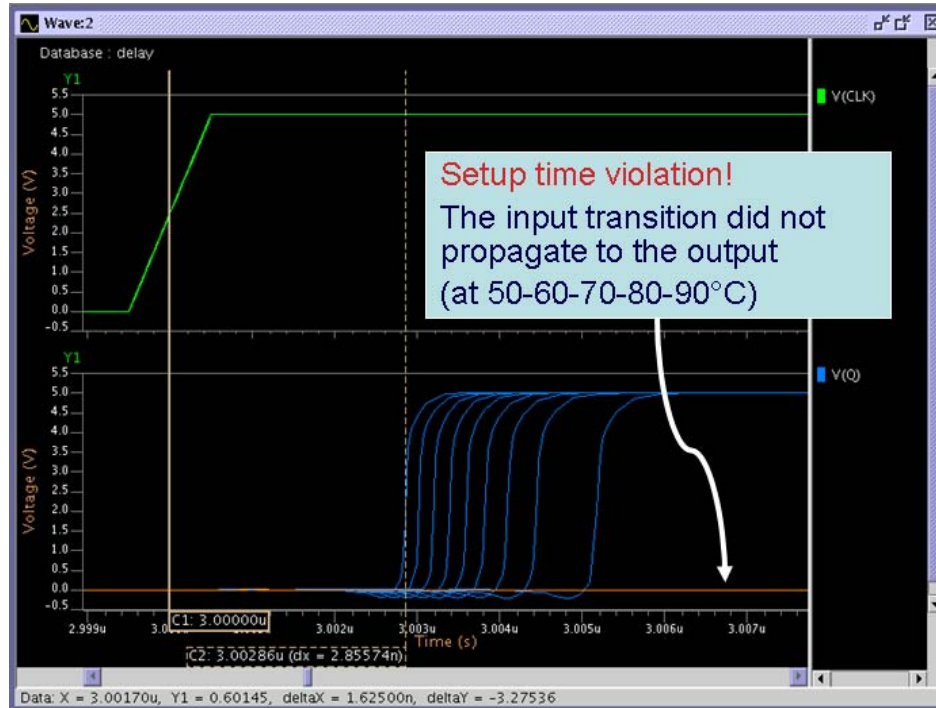


Figure 7 Output transient results when sweeping ambient temperature

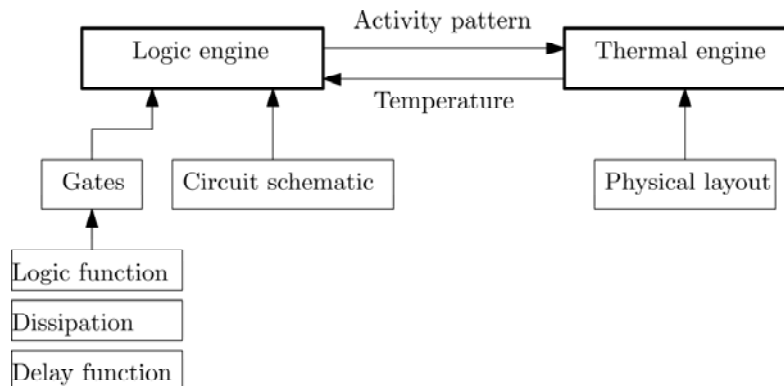
### Combining temperature map and delay functions

With the previously proposed method, temperature-delay functions of the standard cells have been acquired. Using the temperature map resulting from the *CellTherm* application actual temperatures of the cells can be determined. Interpolating the temperature values into the temperature-delay functions the delays belonging to a certain temperature can be derived. The *CellTherm* application is able to back-annotate these temperature-dependent delay values into the running logic simulation via SDF structures. After every simulation time-step, the simulator continues calculation with the altered delay values. If delays change too much with the temperature and setup or hold timing requirements cannot be satisfied any more, the simulator sends off an error message and stops simulation.

### 3 The Logitherm simulator

The Logitherm simulator is a system of two engines: a logic and a thermal one that are closely coupled to perform a digital simulation that takes into account the gate delays and their dependence on temperature. The temperature of the gates is calculated by the thermal engine based on the dissipation pattern yielded by the logic engine.

The engines are written in standard C++ and with easy extensibility in mind. The important elements of the simulator can be altered, replaced or complemented in a very simple and well-defined way.



*The structure of the Logitherm simulator*

#### The logic engine

The logic engine realizes all the ordinary features that are associated with such a tool. A signal type was defined that can handle undefined and high impedance values along with the two logic values (true and false). A logic gate can have an arbitrary number of ports that comprise an arbitrary size of bits, each of the aforementioned signal type. The direction of the ports can be set dynamically, but static directions can also be modelled. The gates will not allow their inputs to be driven by more than one source nor their outputs to be driven by an external source.

What makes this engine special is that the gates also have a temperature property that is updated constantly and a delay that is a function of the temperature.

Whenever a change occurs at any of the inputs of the gate, the function that describes its operation is called and it decides whether the given change will alter the outputs or not. If the delay of the gate is not zero then nothing actually happens at the time of the function call, the new value of the outputs is merely recorded. When the delay time is over the gates connected to the given output port are notified of the new value which triggers a call to their own descriptor function.

In every time step of the simulation the gates record whether they were active in that step or not. A gate is active in a time step if any of its inputs or any of its outputs change. Such an event means that the gate dissipates power. This information is fed to the thermal engine.

It is very easy to define a new gate. The abstract Gate class has to be extended by defining one function that gives the value of the gate's outputs as a function of its inputs. The temperature-delay function can be added to the gate also by defining a simple function that gives the delay for a temperature value.

The gates are loaded dynamically by the simulator framework. This means that a new gate can be added to the system without altering the code of the engine. The shared library containing the new gate or gates simply has to be referenced in the configuration file of the project and everything else is done automatically.

The input format for the circuit description and the specification of excitation can vary – a parsing framework was created in the frames of the project that enables the developers to define grammars and handler functions for matches with little effort. This way, the project can be enabled to receive any (preferably human readable) formats that the project partners wish.

### **The thermal engine**

The thermal engine can be the bottleneck in the logi-thermal simulation as analogue calculations always take a much longer time than the simulation of a pure digital behaviour. This means that there is a trade-off between precision and simulation time. In fact this one of the most important issues that the developers of a logi-thermal simulator face. The neglected details have to be chosen very carefully to remain at a certain level of accuracy while achieving a relatively high speed.

At the moment the thermal engine is a very simple electronic circuit solver. The solver can calculate the transient response of a circuit consisting of conductances, capacitors and current generators to any arbitrary excitation. This means that a system of linear differential equations has to be solved. The calculations are based on sparse matrices and an LU decomposition is performed.

Temperature is modelled with electric voltage, power dissipation with electric current, heat conductance with electric conductance and heat capacitance with electric capacitance. The dissipating areas (logic gates) are current generators in the electric equivalent circuit.

In our very simple model, the logic gates are represented with areas that have a homogeneous dissipation. The integrated circuit is divided into small volumes which are represented by a capacitance (calculated from the silicon's heat capacitance), conductances (based on silicon's heat conductance) and current generators if the given volume overlaps with a logic gate. In this case the power dissipated by the fraction of the gate that takes place on the volume is proportional to the ratio of the overlapping area and the surface under examination.

The value of the current generators is determined by the activity results coming from the logic engine. At this point the time steps of the two engines are equal but this is not efficient as the time constant of the two systems differ in orders of magnitude. This problem can easily be solved by placing a proxy element between the digital and the thermal engines that handles the differences in the time steps without the engines knowing about this.



## **4 Summary and future plans**

A new tool is under development in T3.1 of the Therminator project at BME that will enable designers to acquire information about the thermal behaviour of their digital circuits at a speed close to a pure digital simulation. This information can be vital in complex circuits operating at a high frequency as the change in gate delays caused by the self-heating of a circuit can corrupt timings and thus the proper operation of the entire system.

Two separate realizations of a logi-thermal simulator are being constructed. One forms a cooperation between existing tools to achieve the goal and the other one is an entirely new tool implemented from scratch.

A cooperation is forming between UNIBO and BME to integrate the results of the two institutions. The points of view of the two teams are at a different level: UNIBO examines the behaviour of the digital circuits at a high abstraction level, while BME stays at the gate or cell level. A very efficient tool could be created by integrating the two approaches and performing detailed calculations only on subcircuits where an increased activity and heating is detected.