Timed verification of the SPSMALL memory*

Manuel BACLET[1] and Rémy CHEVALLIER[2]
[1] LSV - CNRS UMR 8643 - ENS de Cachan
61, Av. du Président Wilson - 94235 Cachan Cedex - France
baclet@lsv.ens-cachan.fr
[2] ST Microelectronics - D.A.I.S - Central R&D
850, rue Jean Monnet - 38921 Crolles cedex - FRANCE
remy.chevallier@st.com

Abstract

The aim of the paper is to verify a small synchronous memory component with the real-time model checker Uppaal, taking into account the electrical propagation delays through gates and along wires.

1 Introduction

Timed automata Timed automata have been proposed by R. Alur and D. L. Dill in [AD90]. They are a suitable model for describing and verifying real-time systems. They have been widely studied and efficient algorithms have been designed and implemented in real-time model-checker such as HyTech[1], Kronos[2] or Uppaal[3] and the verification of many real cases has been successfully carried out.

System on Chips The micro-electronics firms have two main focuses on the digital products: firstly the improvement of the product capabilities, and secondly the decrease of the product cost. In order to conciliate these two antagonistic objectives, the only way is the increase of the system integration capability. One chip includes today more functionality than few chips of yesterday plugged together: a complete system, processors, peripherals and memories are gathered now in the same component. This component is called a System on Chip (SoC). The other big issue is that the improvement of the system capability induces the explosion of the data flows in the SoC and then, the explosion of the memory size needed in the component. The memories integrated in a SoC take then more and more surface on the silicon and the time spent by the system to read and write data in the memories is directly impacting the performances of the SoC. The timings of the memories must be then carefully computed and checked. The verification by timed automata is then a good solution in order to be sure that the right behavior of the memory is preserved even if the timings of the memories needed by the SoC are very aggressive.

The Spsmall memory component In this paper, we propose to model and verify the Spsmall component, which is small SRAM module manufactured by ST Microelectronics, using the real-time model checker Uppaal. The main point is that we want to take into account the propagation delay of electrical signals through transistors and along wires. The Spsmall module is a small memory with a maximum total capacity of 64 kbits (3 to 512 words of 2 to 256 bits). We chose to model the smallest memory consisting in three words of two bits, which leads to a netlist of 305 transistors.

The electrical network consists in the connection of several *logical gates* and *registers*. Each component has a characteristic *propagation delay* which corresponds to the time needed to obtain the right output value if its inputs are kept constant. Those

*Partially supported by project MEDEA+ Blueberries

1. http://www-cad.eecs.berkeley.edu/ tah/HyTech/
2. http://www-verimag.imag.fr/TEMPORISE/kronos/
3. http://www.uppaal.com/

delays leads to the definition of *setup* and *hold* time for the input signals of the memory.

Related works Timed verification of digital circuits has been studied in [BJMY02]. The authors succeded the checking of various circuits of sizes up to 22 gates. If we suppose that 6 transistors are needed for each gate, this leads to more or less 132 transistors. Hence, the SPSMALL memory, with its 305 transistors, is beyond this threshold. In our approach, we take into account several characteristics that allows the verification of much larger circuits:

- the module we want to model is a memory. Hence, we use the fact that the module is naturally divided into several subentities, namely the input latches, the address decoder, the memory array and the output stage.
- we take advantage of the *symetry* of the design, since the memory array consists in 3 banks of 2 bits memories.

Finally, we think that our approach can be adapted to deal with other memory design.

Outline The memory component and the specifications it is intended to have are detailed in section 2. In section 3, we describe how the modeling of the memory was carried out. Finally, in section 4, we detail our results.

2 Description of the memory

2.1 Functional description

The memory is a synchronous device: all operations (read, write) are synchronized with the rising edge of a unique clock. We now describe briefly the different signals involved in these operations. Address (A), input data (D), write enable not (WEN) and chip select not (CSN) signals are latched in at the rising edge of the clock (CK). If CSN is set high at the rising edge of CK, it inhibits the access to the memory for the current clock cycle, and the memory is held in stand-by mode. WEN precises the direction of a memory operation (read when set to 1 and write when set to 0). Output data are internally latched at the completion of each access, and are valid until the next positive edge of CK. During write operations, data written into the memory are copied back to the output data pins (write-through). The OEN signal is used to disable the output pins (high impedance state).

2.2 Timing description

Two timings are defined for each input signals: the setup timing, $setup$ and the hold timing, $hold$. These timings are the delays before and after the rising edge of the clock for the signals to be stable to be correctly interpreted by the system. This means that if a rising edge occurs at date t, the input signal must keep constant value in the interval $[t-setup, t+hold]$. Those definitions are motivated by the fact that the electrical signals are not instantaneous and hence delays of propagation along wires and through gates must be considered to obtain right behaviors.

2.3 Architectural description

The real implementation of the memory is described in Fig. 1.

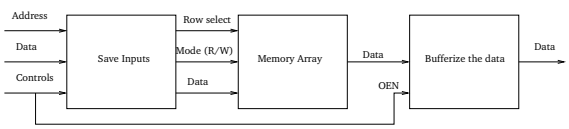

Figure 1: General design

The «save inputs» part is connected to the input signal and saves in registers the value of the data, the address, the chip select signal and the write enable signal. The «memory array» part includes the memory array and the select logic associated to it. The «bufferize the data» part manages the outputs with the output enable signal and the data saved in the memory or the data set at the input of the design (write mode).

2.4 Specifications

The properties we want to check are the following:

i. When the environment performs a write operation, the new data is effectively stored in the memory (if the address is valid) and available

on the output pin $taaw$ ps after the rising edge of the clock. (At a rising edge of the clock, if WEN is set to 0, CSN to 0,

- if A_0A_1 is not 11, the value of the input data D_0D_1 is saved into the memory array at address A_0A_1
- the output Q_0Q_1 is set to the value of D_0D_1 after $taaw$ ps.)

ii. When the environment performs a read operation, the right value stored in the memory array is available on the output pins taa ps after the rising edge of the clock. (At a rising edge of the clock, if WEN is set to 1 and CSN to 0, Q_0Q_1 is set to the value kept into memory at address A_0A_1 after taa ps. If the address is invalid, the output values remains unchanged)

3 Modeling

We propose to model this memory by a network of timed automata. We were provided a RTL netlist described in VHDL which was obtained from the transistor netlist with the tool TLL [BDPG02] from the TNI-Valiosys company which is involved in the MEDEA+ Bluberries project. This tool was also used to compute the values of the propagation delays along wires and through gates.

The main complexity issue we faced was the potentially high number of clocks if we model each gate separately, leading to untractable verification. Hence we decided to carry out several abstractions and we now detail them.

3.1 Abtracting the model

On our first tries, the model-checking was not tractable. Hence, we proposed several abstractions that have led to a quite smaller model and that can be used to model memories of any capacity.

The OEN (output enable not) signal is only used in the output stage circuit and we can suppose that it is always set to 0 since verifying timings for this signal can be carried out by modeling only the output stage.

The CSN (chip select not) signal was also abstracted and in the sequel, we suppose that it is always set to 0.

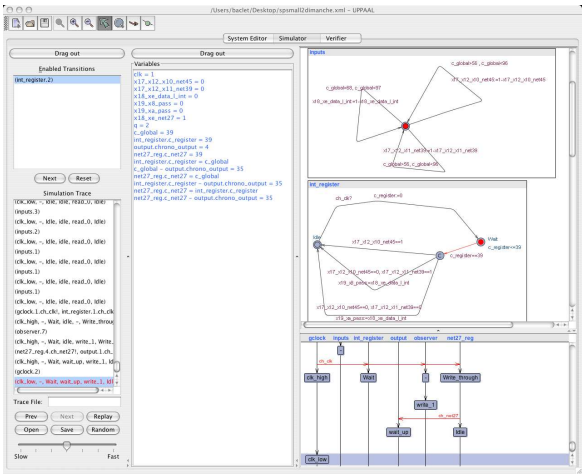

Figure 2: A snapshot of Uppaal

The following idea was considered: since we are only interested in checking timings, we can suppose that the component has correct behaviors when using non agressive cycle, setup and hold timings. Hence, we can check the memory behaviors using only one data bit and only one address bit. Hence, we set A_1 and D_1 to 0.

3.2 Modeling with Uppaal

Using these ideas, we propose an abstracted modeling which is of much smaller size and it was possible to produce easily an Uppaal model since Uppaal provides debugging facilities and a graphical interface (see Fig. 2). Descriptions of models handled by Uppaal can be found in many papers [BL96, LPY97]. We now give a short description of the automata associated to the different subentities of the memory module.

3.2.1 The clock modeling

The clock signal was modeled with a 2-states automaton: clk_low and clk_high. The rising edge of the clock event is modeled by the message ch_clk that the automaton sends periodically.

3.2.2 Inputs modeling

The three input signals (WEN, A, D) were modeled with a 1-state automaton. Those signals behave in the following way: they can change their value at

any time except within intervals of the form $[t - setup, t+hold]$ where t is the date of a rising edge of the clock and $setup$ and $hold$ are the characteristic timings of the considered input signal.

3.2.3 The memory array modeling

It is modeled by a 2-states automaton which has the following behavior: when it receives a rising edge of the clock signal (ch_clk), it enters a wait state. When it is able to leave this state, it comes back to the idle state while updating (or not) the values stored in the memory array depending on the current input values.

3.2.4 The output latch modeling

This automaton has three states: $Write\_through$, $Read$ and $Idle$. When the automaton receives a rising edge of the clock signal ($ch\_clk?$), it goes to one of the states named $Write\_Through$ or $Read$, depending on the operation performed during the cycle and then wait in this state for a duration that depends on the operation. Then it comes back to the $Idle$ state while updating, if needed, its output (and if it is the case, it sends a message ch_output_latch).

3.2.5 The output stage modeling

This automaton models the combinatorial circuit that manages the output of the output latch. When the automaton detects that the signal of the output latch changes its value (by receiving a message ch_output_latch), it sets the output signal Q to the undetermined value (usually denoted by the letter X), it enters a wait state ($Wait_up$). After some delay, it comes back to the $Idle$ state while updating the value of signal Q.

4 Model-checking and results

Thanks to Uppaal, we were able to show that our modeling has the nice property of being deadlock free and to ensure that the access time in read mode is 0.77 ns and in write-through mode 0.57 ns. More precisely, we showed that those timings are optimal since the memory model has incorrect behaviors when choosing smaller timings.

Conclusion

In this paper, we describe a modeling in Uppaal of a small SRAM component. We detail how the modeling was abtracted from the memory description (so that model-checking remains tractable) and we succeeded in verifying formally the modeling. More precisely, we were able to ensure that the read and write access timings are optimal. Future works will consist in finding a more precise modeling (e.g. by taking into account the CSN signal that we abstracted) and maybe seeing whether the timings of the memory can be computed from the transistors netlist and few reference gate timings.

References

[AD90] R. Alur and D. L. Dill. Automata for modeling real-time systems. In *Proc. 17th International Colloquium on Automata, Languages, and Programming (ICALP'90)*, volume 443 of *Lecture Notes in Computer Science*, pages 322–335. Springer, 1990.

[BDPG02] P. Bricaud, T. Delaye, F. Poirot, and A. Gorun. Do standardized embedded IP transistor view exists for SoC IP integration? In *Proc. 4th Sophia Antipolis forum on MicroElectronics (SAME'2001)*, 2002.

[BJMY02] M. Bozga, H. Jianmin, O. Maler, and S. Yovine. Verification of asynchronous circuits using timed automata. In *Proc. 1st Workshop on the Theory and Practice of Timed Systems (TPTS'2002)*, volume 65 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 2002.

[BL96] J. Bengtsson and F. Larsson. UPPAAL, a tool for automatic verification of real-time systems. Master's thesis, Department of Computer Science, Uppsala University (Sweden), 1996. ISSN 0283-0574.

[LPY97] K.G. Larsen, P. Pettersson, and W. Yi. UPPAAL in a Nutshell. *Journal of Software Tools for Technology Transfer*, 1(1–2):134–152, 1997.