Global Journal of Computer Science and Technology: A Hardware & Computation
Volume 18 Issue 1 Version 1.0 Year 2018
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Implementation of Extracted Timing Methodology on Process Monitor for Silicon Characterization

By Bhagyasri Chandaka & Dr. R. Ramana Reddy

Abstract- Process variations are playing a key role in defining the behaviour of an IP. These process variations can accurately measure using process monitor. In order to verify process variations, the process monitor should meet all timing requirements. Static Timing Analysis (STA) uses best case/ and worst analysis overly pessimistic, and could be optimistic also in some cases. Static Timing Analysis (STA) is a method for estimating yield of a circuit in terms of timing activities. Model extraction is a technique that accurately captures the characteristics of interface logic of a design in the form of a timing library model and provides a capacity improvement in timing verification by more than two orders of magnitude. Extracted timing model is an efficient timing library model to get accurate timing arcs of the circuit. This paper describes Methodology for creating timing models and also the flow to develop IP (process monitor) ETMs (.lib) using Synopsys Primetime tool, which can be used in any SOC and ETMs (Extracted timing models)with necessary time-budgeting instead of IP Netlists. Generated ETM with and without annotation delays and compared the library file. And the process monitor's ring oscillator is designed through Verilog code using cadence tool.

Keywords: *back annotation, ETM, ILM, process monitor, QTM, ring oscillator, STA.*

GJCST-A Classification: *B.7.m*

Strictly as per the compliance and regulations of:

Implementation of Extracted Timing Methodology on Process Monitor for Silicon Characterization

Bhagyasri Chandaka [α] & Dr. R. Ramana Reddy [σ]

Abstract- Process variations are playing a key role in defining the behaviour of an IP. These process variations can accurately measure using process monitor. In order to verify process variations, the process monitor should meet all timing requirements. Static Timing Analysis (STA) uses best case/ and worst analysis overly pessimistic, and could be optimistic also in some cases. Static Timing Analysis (STA) is a method for estimating yield of a circuit in terms of timing activities. Model extraction is a technique that accurately captures the characteristics of interface logic of a design in the form of a timing library model and provides a capacity improvement in timing verification by more than two orders of magnitude. Extracted timing model is an efficient timing library model to get accurate timing arcs of the circuit. This paper describes Methodology for creating timing models and also the flow to develop IP (process monitor) ETMs (.lib) using Synopsys Primetime tool, which can be used in any SOC and ETMs (Extracted timing models)with necessary time-budgeting instead of IP Netlists. Generated ETM with and without annotation delays and compared the library file. And the process monitor's ring oscillator is designed through Verilog code using cadence tool.

Keywords: back annotation, ETM, ILM, process monitor, QTM, ring oscillator, STA.

I. Introduction

SOC contain some analog and digital circuitry in the form of pre-designed blocks commonly known as intellectual property (IP). Soft IPs are processor cores, peripheral interfaces etc. and Hard IPs are analog blocks like PLLs etc. There are different challenges for SOC timing closure when multiple IPs is integrated. Soft IPs is implemented by performing synthesis, constraints and timing analysis followed by back-end process. Whereas Hard IPs requires set of views that provide appropriate information of IP at each stage that are used.SOC timing can be implemented in following ways.

1. Flat timing approach
2. Hierarchical timing approach

Flat timing approach will consume more time and becomes complicated at initial timing analysis when the designer go for larger designs with many IPs (As we read gate level netlist, SDC, SPEF). Once IPs timing is closed any how designer do Timing signoff on flat design. This paper describe the methodology for SOC timing closure using hierarchical approach, which generates IP hierarchical timing models and used for SOC level to close interface timing. This way it helps us faster timing convergence at SOC level. Process monitor includes a variety of test structures in the form of ring oscillators developed to easily integrate into SoC designs. The PVT conditions of silicon can be change from layer to layer or from lot to lot. In this paper section-2 presented the basic structures of process monitor's ring oscillator and how to interface the process monitor with the SOC. Timing extraction plays an important role in the hierarchical analysis flows by reducing the complexity of timing verification.

L.T.N.wang shows implementation of process monitor's ring oscillator sensitive frequency layout dependency nature using gate lithography [2]. Cristiano Forzan, Davide Pandini shows a solution that is the design performance with variability problem of accurately evaluating is statistical static timing analysis (SSTA). Based on the probability function the timing analysis run time is estimated in a single run[3]. R. Saleh, S. Wilton, S. Mirabbasi shows the process of creating a reusable IP block, differs from the traditional ASIC design, implementation of reusable IPs takes more time but these IPs can use in any SOC [6].Cho W. Moon, Harish Kriplani and Krishna Belkhale shows timing model extractor builds a timing model of a digital circuit for use with a static timing analyzer. And the implemented structure easy to analyse and which reduces the run time and complexity of the SoC [4].So the efficient timing methodology, Extracted Timing Methodology (ETM) implementation on process monitor is discussed in this paper. And also the implementation of process monitor's ring oscillator and delay calculations are discussed. Since run time of circuits timing analysis depends on the timing library files of the IPs, by considering the ETM libs the run will reduce.

II. Architecture of Process Monitor

Process monitor includes a variety of test structures in the form of ring oscillators developed to easily integrate into SoC designs. Each process monitor can be instantiated several times in a single chip. The IP supports JTAG and Direct Access interfaces only, for easy integration. Typical process monitor applications

Author α: e-mail: bhagi50sri@gmail.com

include silicon characterization for performance and variations of different key transistors, On-Chip Variation (OCV) measurements. The process monitor block has 2 key components.

- Ring Oscillators (RO)
- Digital block for controllability and observability

The digital block is a top level module that comprises of all ROs and a global control/observe logic. The individual ROs are connected in a daisy chain structure. This module supports up to 64 ROs. The frequency measurement works with a reference clock where the reference clock captures the incoming signal in a counter. The function of this digital control block is to generate one unique enable signal to choose the RO to be tested/characterized. Once an individual RO is enabled, the output of the RO is propagated down through the daisy chain and goes through a 32-stage ripple counter. A large divider is needed to ensure $F_{refclk} >> F_{means}$. After synchronization with reference clock and further conditioning the signal is passed to capture counter.

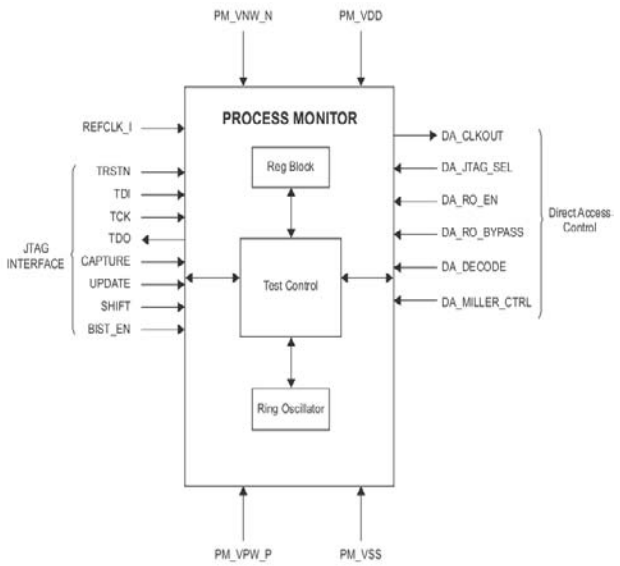

Fig. 1: Block diagram of process monitor

Silicon manufacture companies usually put considered some test structures in scribe line on selected wafers in each lot to observe and track the effects of process variation and DC characteristics of MOS devices, but these structures do not capture all the variation data. They also don't provide explicit information to the designer about the corner the chip is running. Test structures like ring oscillators are generally used to get more information on a chip's process corner. Such structures are placed at various locations across the chip to judge the effect of process variations on standard cell delays and performance. Ring Oscillator is the basic block in process monitor. Implementation of process monitor's ring oscillator is shown below.

a) Ring Oscillator

The purpose of the ring oscillator is to characterize the performance and behaviour of the key transistors across the die. Each ring oscillator (RO) makes use of classical ring oscillator structure of the core. The purpose of the ring oscillator is to characterize the performance and behaviour of the key transistors across the die. Ring oscillator structure as the core, for the process measurement along with additional peripheral circuit.

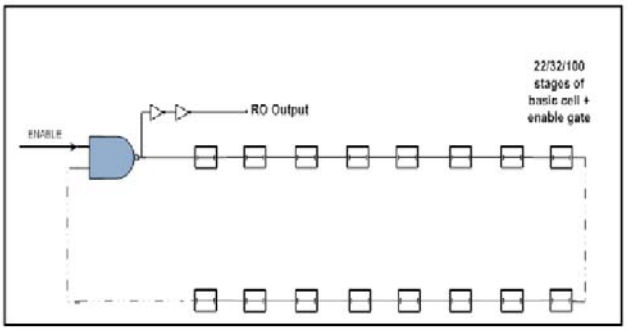

Fig. 2: Process monitor's Ring Oscillator structure

Ring Oscillators are built using basic standard cells with different RO depths. There are two types of ROs one with 23 RO depth and another with 101 RO depth. Depth indicates the number of cells (inverter) in the Ring Oscillator. The above fig 2 represents the basic structure the process monitor's Ring Oscillator. Based on the standard cells used in the SOC the particular Ring Oscillator will be selected. PVT (process, voltage and temperature) conditions are changing from lot to lot or wafer to wafer these changes can easily identified by measuring the Ring Oscillators delay. Performance of the circuit is calculated using the delay of the ring oscillator. The delay calculation of the ring oscillator is described below.

b) Delay calculations

The following are the assumptions for ROs:

- Each RO has a 4-bit ripple divider; number of divider stages "M" (4).
- Number of stages in the ring oscillator being characterized "A".
- TM has a 8-bit Ripple divider in the test module; number of divider stages "N" (8).
- 24-bit output counter – CDIV.
- The maximum ring frequency is < 5GHz.
- Choose reference frequency that is at least 10 MHz
- Start Timer Value w.r.t reference frequency, which indicates when to start counters (START_VAL).
- Stop Timer Value w.r.t reference frequency, which indicates when to stop counters (STOP_VAL).

The RO frequency in each test structure is given by:

Frequency of RO

$$F_{ring\_osc} = \frac{2^{M+N} X\ F_{ref}\ X\ CDIV}{STOP\_VAL - START\_VAL} \quad (1)$$

Average gate delay of each ring gate:

$$T_{pd} = 0.5\ X\ \frac{1}{AX F_{ring\_osc}} \quad (2)$$

The start and stop timing information is available in the registers which are in process monitor and programmed through JTAG. The counters will counts the number of clock pulses arrived between the start and stop time. By using above formula one gate delay can be calculated. The simulations results are shown in the below section, which are generated in cadence Verilog tool.

c) Simulation Results

REF_clk time period is 20ns.12-bit counter will count upto 4096. For 23 chain Ring Oscillator, DA_OUT time period is (2x23x4096) 188.416ns and for 101 chain Ring Oscillator the DA_OUT time period is (2x101x4096) 827.392ns which is shown in the fig 3 and fig 4. Delay of the gate is calculated by using the equation (1).One gate delay is 1ps and time period is 2ps. DA_CLKOUT shows the output frequency of ring oscillator which is shown in the fig 3. TDI, TDO, CAPTURE, SHIFT AND UPDATE are the interface signals.

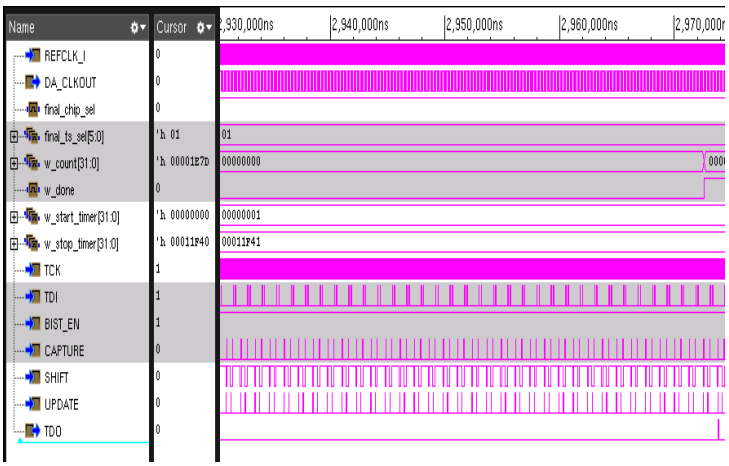

Fig. 3: Process monitor JTAG interface simulation waveforms

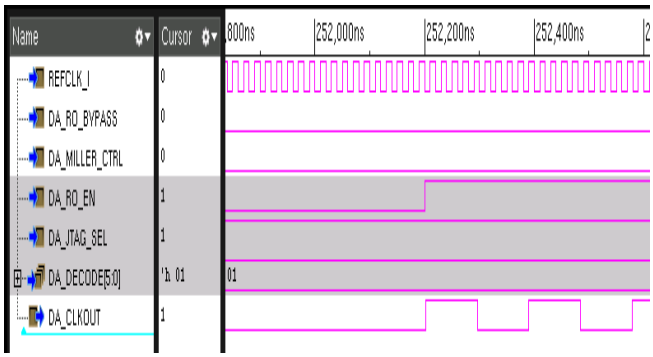

Fig. 4: Process monitor's DA (Direct Access) interface simulation waveforms

III. Static Timing Analysis (STA)

Static timing analysis is a method of validating the timing performance of a design by checking all possible paths for timing violations without having to simulate. No vector generation is required, no functionality check is done.

Timing models are to represent the timing characteristics of complex blocks in a design. A timing model of the block models the full input and output timing characteristics without the complete netlist of the block. There are some timing models like QTM and ILM, which are used based on application.

a) Quick Timing Models (QTM)

In the early stages of the design cycle, if a block does not yet have a netlist, the designer can use a quick timing model to describe its initial timing. Later in the cycle, it can replace each quick timing model with a netlist block to obtain more accurate timing. To create a quick timing model the designer can use a series of Prime Time commands to specify the model ports, the setup and hold constraints on the inputs, the clock-to-output path delays, and the input-to-output path delays. It is also possible to specify the loads on input ports and the drive strength of output ports. The generated file can save a quick timing model in the Synopsys .db format, then instantiate the quick timing model in a design just as the designer would instantiate library cells or interface timing specification models.

b) Interface Logic Models (ILM)

An interface logic model (ILM) is a partial netlist that contains only the interface logic of a block. The ILM contains.

- The combinational logic from each input port to the first stage of sequential elements of the block.

- The combinational logic from the last stage of sequential elements to each output port of the block.
- The clock paths to these sequential elements.
- Combinational paths from the input ports that do not encounter a sequential element and directly to an output port.

A generated ILM is context-independent, which means that the model is accurate for a range of operating environments. When the model is used in a design, the timing behaviour of the model is different for different operating environments.

An ILM is a partial netlist that retains the combinational logic from each input or output port to the first or last stage of sequential elements of the block. The intent of an ILM is to produce a model that closely resembles the timing of the interface to that of the block-level netlist.

IV. Extracted Timing Methodology (ETM)

The designer can generate an extracted timing model (ETM) for a block with a corresponding technology-mapped gate-level netlist. Using extracted timing models provides the following benefits:

- Reduces the runtime and memory for full-chip analysis, the designer can run chip-level analysis with extracted models in place of the gate-level netlist for some modules.
- Protects the intellectual property of a netlist-based core.

Extraction uses a technology-mapped netlist as input and generates a context-independent timing model in the Synopsys .db format or Liberty format. The generated model contains the same timing behaviour as the original netlist. The delay values of arcs in the model are within the user-defined tolerance of the original path delays. The generation of ETM flow is represented in the fig 5.

To generate ETM timing lib, pre requisites are design constraint file, command script to perform step by step analysis and library files. If the generated model meets over timing requirements, then it can be used in any SOC. fig 6 represent the timing arcs model for ETM.

The delay data in the timing arcs is accurate for a range of operating environments. The extracted delay data does not depend on the specific values from input transition times, output capacitive loads, input arrival times, output required times, and so on. When the model is used in a design, the arc delays vary with the input transition times and output capacitive loads. This is called a "context-independent" model because it works correctly in a variety of contexts.

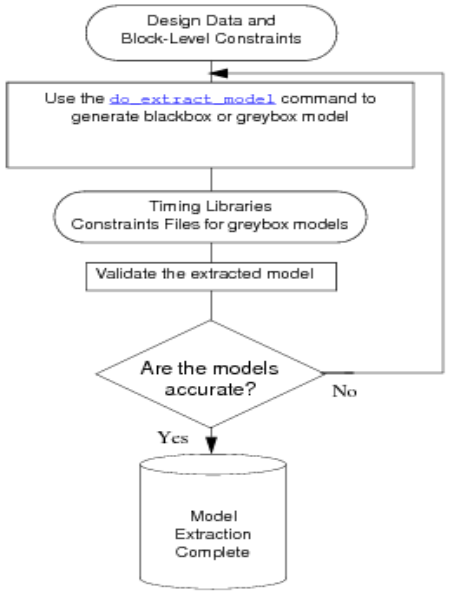

Fig. 5: ETM generation flow chart

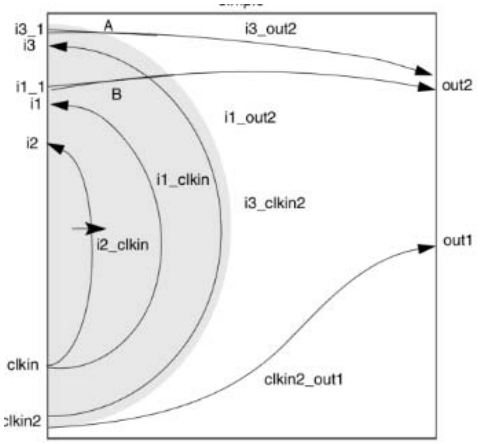

Fig. 6: ETM timing model of core cell

The characteristics of the ETM model depend on the operating conditions in effect at the time of extraction. However, clocking conditions and external constraints do not affect the model extraction process. There are some commands like create_clock, set_clock_latency, set_clock_uncertainty, set_input_delay, and set_output_delay, do not affect the model extraction process, but the extracted model, when used for timing analysis, is sensitive to those commands.

By specifying the annotation delay the timing analysis is more relevant to the practical conditions. That is after layout of the circuits resistance and capacitance values are added which increases the delay. In back annotation process these delay values also considered.

a) Back Annotation

Back-annotation is the process of reading delay, resistance, and capacitance values from an external file into the tool for timing analysis. Using back-annotation, the designer can more accurately analyze

the circuit timing in the tool after each phase of physical design. For initial static timing analysis, PrimeTime estimates net delays based on a wire load model. Actual delays depend on the physical placement and routing of the cells and nets. A floor planner or router can provide more detailed and accurate delay information, which the designer can provide to PrimeTime for a more accurate analysis. This process is known as back-annotation. And this delay information is often provided in an SDF file. The back annotation delay on the ports can be provided by specifying the set_annotation_delay command. Here the annotation delay is 50% of the time period.

```
pin("TDI") {
direction : input ;
max_transition : 50.000000 ;
capacitance : 1.885365 ;

/* Other user defined attributes. */
original_pin : TDI;
timing () {
related_pin : "TCK" ;
timing_type : setup_rising;
rise_constraint(f_dtrans_ctrans){
index_1 ( "0.000000, 3.088950, 7.836690, 19.881800, 50.000000");
index_2 ( "0.000000, 2.422370, 5.442220, 12.226800, 50.000000");
values( "498.858948, 497.792267, 496.238129, 492.614716, 476.830109",\
        "500.336609, 499.269928, 497.715790, 494.092377, 478.307770",\
        "502.833191, 501.766510, 500.212372, 496.588959, 480.804352",\
        "509.181091, 508.114410, 506.560272, 502.936859, 487.152252",\
        "520.603943, 519.537292, 517.983154, 514.359741, 498.575104");
        }
fall_constraint(f_dtrans_ctrans){
index_1 ( "0.000000, 3.088950, 7.836690, 19.881800, 50.000000");
index_2 ( "0.000000, 2.422370, 5.442220, 12.226800, 50.000000");
values( "523.261780, 522.195129, 520.640991, 517.017578, 501.232941",\
        "524.642761, 523.576111, 522.021973, 518.398560, 502.613922",\
        "526.961731, 525.895081, 524.340942, 520.717529, 504.932892",\
        "532.568298, 531.501648, 529.947510, 526.324097, 510.539459",\
        "541.497620, 540.430969, 538.876831, 535.253418, 519.468750");
        }
} /* end of arc TCK_TDI_stupr*/
```

Fig. 7: Timing arcs on TDI pin without annotation delay

```
pin("TDI") {
direction : input ;
max_transition : 50.000000 ;
capacitance : 1.885365 ;
/* Other user defined attributes. */
original_pin : TDI;
timing () {
related_pin : "TCK" ;
timing_type : setup_rising;
rise_constraint(f_dtrans_ctrans){
index_1 ( "0.000000, 3.088950, 7.836690, 19.881800, 50.000000");
index_2 ( "0.000000, 2.422370, 5.442220, 12.226800, 50.000000");
values( "548.755676, 547.689026, 546.134888, 542.511475, 526.726807",\
        "550.233337, 549.166687, 547.612549, 543.989136, 528.204468",\
        "552.729919, 551.663269, 550.109131, 546.485718, 530.701050",\
        "559.077820, 558.011169, 556.457031, 552.833618, 537.048950",\
        "570.500671, 569.434021, 567.879883, 564.256470, 548.471802");
        }
fall_constraint(f_dtrans_ctrans){
index_1 ( "0.000000, 3.088950, 7.836690, 19.881800, 50.000000");
index_2 ( "0.000000, 2.422370, 5.442220, 12.226800, 50.000000");
values( "573.159424, 572.092773, 570.538635, 566.915161, 551.130615",\
        "574.540405, 573.473755, 571.919617, 568.296143, 552.511597",\
        "576.859375, 575.792725, 574.238586, 570.615112, 554.830566",\
        "582.465942, 581.399292, 579.845154, 576.221680, 560.437134",\
        "591.395264, 590.328613, 588.774475, 585.151001, 569.366455");
        }
} /* end of arc TCK_TDI_stupr*/
```

Fig. 8: Timing arcs on TDI pin with annotation delay

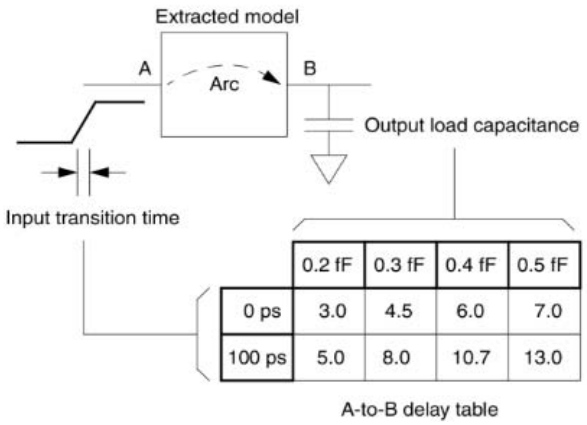

Fig. 9: Delay calculation from the generated ETM timing arcs

The delay of the cell or macro is calculated by considering the input transition time and output load capacitance which is specified in the generated ETM library. Index 1 indicates the input transition time and index 2 indicates the output capacitance. And these values can be adjusted to meet the timing budget at SOC level.

V. Conclusion

The timing analysis solutions in use today have been extended to more complex circuit analysis and variation.ETM (Extracted timing models) with necessary time-budgeting instead of IP Netlists reduces run-time and memory usage and protects the IP. By considering the annotation delays the generated library is more relevant to the real time environment. The timing models in hierarchical analysis saves the runtime. And by using process monitor the gate is 2ps is measured, which is an accurate process to measure the delay of the gate.

References Références Referencias

1. AugusliKifli, K C Wu (2015). "SOC test integration platform". IEEE transaction on Very Large Scale Integration (VLSI). Volume: 15.
2. L .T. N. wang, N. Xu, S. O. Toh, A. R. Neureuther, T. J. kingLiu, B. Nikolic. "parameter specific ring oscillator for process monitor at the 45nm node". Custom integration circuit conference, 2010, IEEE.
3. Cristiano Forzan, Davide Pandini (2009). "Statistical static timing Analysis". The VLSI journal, Integration, volume 42.
4. Cho W. Moon, "Timing Model Extraction of Hierarchical Blocks by Graph Reduction",Harish Kriplani and Krishna Belkhale, pp. 152-157, 2002. (Proc. Design Automation Conference).
5. M. Bhushan, A. Gattiker, M. B. Ketchen and K.K. Das, "Ring Oscillators for CMOS Process Tuning and Variability Control". In IEEE Trans. on Semiconductor Manufacturing, vol. 1, no. 1, pp. 10-19, Feb. 2006.

6. R. Saleh, S. Wilton, S. Mirabbasi (2006). "System-on-chip Reuse and Integration". Proceedings of IEEE, volume: 94.
7. Seok-Yoon Kim, N. Gopal, L. T. Pillage (1994). "Time-domain macro models for VLSI interconnect analysis". IEEE transactions on computer-Aided design of Integrated circuits and systems. Volume: 13.
8. EvgeniKrimer, Isaac Keslassey, Avinoamkoladny, Lsaskharwalter, MattanErez. "Static timing analysis for modelling QoS in Network-on-chip". Journal of parallel and distributed computing. Volume 71, Elsevier.
9. CristianGrecu, ParthaPratinPande, Andre Ivanov, Res saleh (2005). "Timing analysis of network on chip architectures for MP-SOC platforms". Microelectronics journal. Volume 36, Elsevier.