Model Transformation Method of Embedded System Hardware for Improving Design Process Performance

Arif Sasongko[1], Maman Abdurohman[2], Kuspriyanto[3], and Sarwono Sutikno[4]

[1,3,4]School of Electrical Engineering and Informatics,
Bandung Institute of Technology, Indonesia
[2]Informatics Faculty,
Telkom Institute of Technology, Indonesia

Abstract*:* Gap between designer productivity and IC technology is a challenge for improving embedded system design methodology. Register Transfer Level (RTL) is a current design abstraction that no longer adequate as the starting abstraction level. It needs a new methodology for facing the lack of RTL. Transaction Level Modeling (TLM) is a candidate design methodology to improve designer productivity. It offers many advantages especially on process time reduction. In this paper, a new model transformation method is proposed for transforming TLM model to RTL model design incrementally. TMC (TLM-RTL Model Communicator) is used as a key component in this method. TMC communicates between TLM and RTL model. GCD (Greatest Common Divisor) processor is designed to show transformation process from TLM to RTL model. The experiment results show that model transformation method using TMC is effective for transforming TLM model to RTL model incrementally. These results show the design process improvement using this method.

Keywords: Design Methodology, Transaction Level Modeling (TLM), Register Transfer level (RTL), TLM-RTL model communicator (TMC), Greatest Common Divisor (GCD).

1. Introduction

The growing of consumer demands for more functionality tools has lead to an increase in complexity of the embedded system designs. One solution supporting those demands is the ability of semiconductor industry to reduce the minimum feature sizes of chip. However, although current IC technology is overing the possibility to satisfy the growing consumer demands, the effort needed in modeling, simulating, and validating such designs is adversely affected. This problem emerges because the current modeling method and frameworks cannot catch the rising complexity of that kind of design.

The utilization of register transfer level (RTL) as a new abstraction layer over gate level design was a revolution step to face this kind of challenge several years ago. The RTL abstraction layer is accepted as current abstraction layer for describing hardware designs. Unfortunately, for current and future product, this abstraction is not enough. In the last decade, there are many design methodologies that proposed to improve RTL capabilities and performance. These design methodologies include Transaction Level Modeling (TLM), hardware-software codesign, system level design and Electronic System Level (ESL) modeling.

Embedded system design

The design flow of embedded system begins with design specification, including system constraints, both cost and processing time. System functionality is defined in behavioral description, hardware software partitioning is done to optimize design result and still satisfy the requirement. Hardware and software integration is done after hardware/software detail design.

Received: November 7th, 2010. Accepted: June 21th, 2011

Register transfer level design is carried out by means hardware programming language such as, Verilog, VHDL and Esterel. Verification and testing process is done to ensure embedded system design is fit to specification [1].

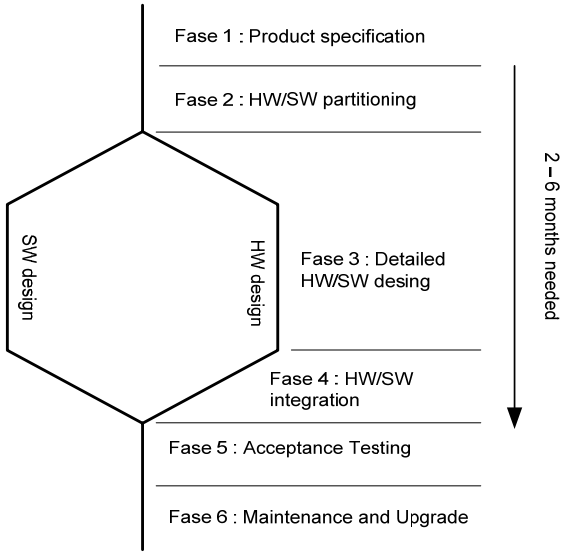

Figure 1. Embedded system design flow [1]

This method is not adequate to catch the current and future complexity as predicted by the Moore Law.

Moore's Law and Productivity gap

Moore's Law said that silicon capacity has been steadily doubling every 18-24 months. This increment allows companies to build more complex systems on a single silicon chip. However, designer ability to develop such systems in a reasonable amount of time is not fit with the increase in complexity. This is referred to as the productivity gap, which is based on the ITRS (International Technology Roadmap for Semiconductors).

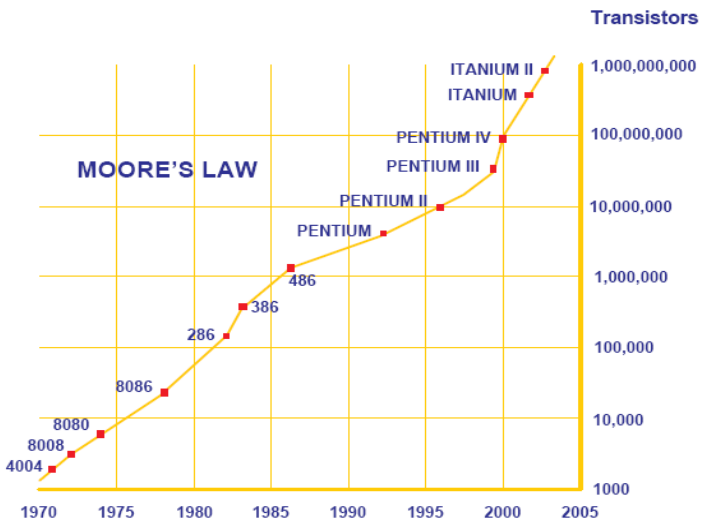

Figure 2. ITRS shows the Moore's law [11]

Increasing the complexity and functionality of electronics systems causes the increase of the possible design choices and the alternatives to explore for optimization purposes. Therefore, design space exploration is vital when constructing a system in order to choose the optimal alternative with respect to performance, cost, etc. The reduction of time to develop these system-level models for optimization purposes can improve design acceleration with acceptable performance. A possible way to reduce this time is to raise the abstraction layer of design.

Register Transfer Level design

One of the past design revolutions in hardware design was the introduction of RTL design layer as the entry point of the design flow. At RT level, registers and a data-flow description of the transfers between them replace the gate-level instantiation of independent flip-flops and logical gates. Some hardware description languages such as VHDL, Verilog and Esterel are used for writing models at this RT level. The translation of RTL model to gate level is called synthesis. Examples of component at this level are adder, multiplexer, decoder, and memory.
The complexity of hardware design combined with the lack of a revolution design approach similar to the RTL introduction has induced very slow simulations and cause productivity gap. The peak problem for system-on-chips is software development requirement, co-simulating embedded software with the RTL model is possible, but too slow to allow an effective development. Designers are forced to wait the final chip to start writing the software of the system. This results wasted time in the development cycle and increased time-to-market. Another approach to address the problem is to try to raise the abstraction level: by creating models with fewer details before the RTL one, it should be possible to achieve better simulation speeds while at the same time less accurate.

There are many efforts to increase embedded system design process performance. One of these is electronic system level. Integrated IP RTL – Extended Finite State Machine (EFSM) is one design to combine some Intelectual Property (IP) with the new design[3]. This concept lies on electronic system level. Another method is Assertion Based Verification (ABV). ABV verifies models to sure that model fullfil these requirements [2].

2. Transaction Level Modeling (TLM) Design Methodology

A. Transaction Level Modeling (TLM)

Transaction-level Modeling fills the gap between purely functional descriptions and RTL model. They are created after hardware/software partitioning. The main application of TLM is to serve as a virtual chip (or virtual platform) on which the embedded software can be run.

The main idea of TLM is to abstract away the physical communication on the buses by so-called transactions: instead of modeling all the bus wires and their state change, only the logical operations (reading, writing etc) carried out by the busses are considered in the model. As contrast to the RTL, where everything is synchronized on one or more clocks (synchronous description), TLM models do not use clocks. They are asynchronous by nature, with synchronization is occurring during the communication between components. These abstractions allow simulations multiple orders of magnitude faster than RTL. Figure 3 shows intermediate models that reside between algorithm model and RTL model.

The other advantage of TLM models is that they require far less modeling effort than RTL or than Cycle Accurate model. This modeling effort is further reduced when the designer has already the C/C++ functional code for the processing done by the hardware block to model. For instance, one can reuse the reference code for a video decoder or for a digital signal processing chain to produce a TL model. It is different from cycle accurate, which is no longer the reference after RTL model is created, TLM is by this means an executable, “golden model” for the hardware. Various definitions of TLM exist; some of them even rely on clocks for synchronization, which looks more like Cycle Accurate level. A transaction term is an atomic data exchange between an initiator and target. The initiator has the initiative to do the

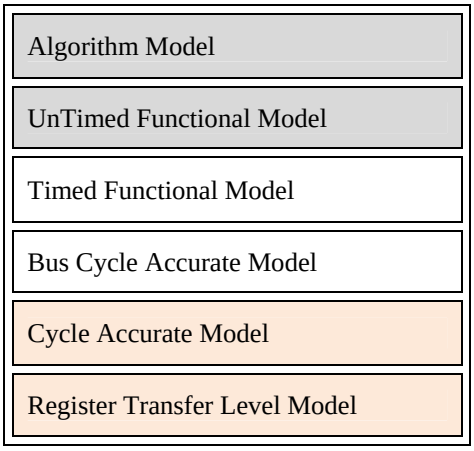

Figure 3. TLM Model Stack

transaction where as the target is considered as always able to receive it (at least, to indicate to the initiator that it is busy). This corresponds to classical concepts in bus protocols. The initiator issues transactions through an initiator port, respectively, the target receives them by a target port. Some components only have initiator ports some have only targets ports. Also, some components contain both initiator and target ports.

The information exchanged via a transaction depends on the bus protocol. However, some of the parameters are generally common to all protocols:

- The *type* of transaction determinates the direction of the data exchange, it is generally read or write.
- The *address* which is an integer determining the target component and the register or internal component memory address.
- The *data* that is sent to received.
- Some additional *meta-data* including : a return status (error, success, etc), duration of the transaction, bus attributes (priority, etc).

The most basic functionality shared by all buses or more generally interconnection networks is to route the transactions to their destination depending on their address. The destination is determined by the global memory address map which associates a memory range to each target port.

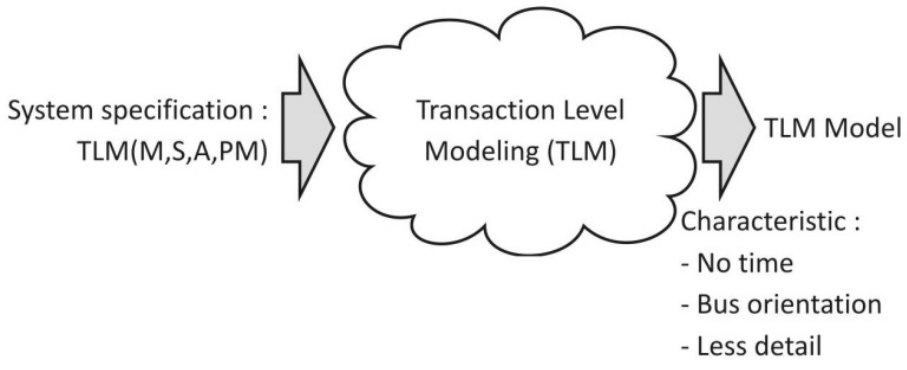

Figure 4. TLM process model

In order for the embedded software to be executed correctly, the address map, the offset for each register must be the same as in the final chip (register accuracy). Additionally, the data produced and exchanged by the components must also be identical (data accuracy). Finally, the interrupts have to correspond logically to the final ones. One can view these requirements as a contract between the embedded software and the hardware. This contract guarantees that if the embedded software runs flawlessly on the virtual platform, then it will run in the same way on the final chip.

B. TLM-RTL Hardware Embedded System Design

The new design flow for modeling hardware embedded system is developed by using transaction level modeling (TLM) method for early verification purpose. Verification process done before the detail design. Transaction level modeling is one of new trends on embedded system design after the development of register transfer level modeling.

There are three stages in detailed design:

1. Hardware part definition: hardware embedded system definition that will be implemented.
2. TLM modeling: Model construction with transaction modeling approach and perform early verification. Model refinement process can be generate by performing 4-tuple correction: M, S, A, PM.
3. RTL modeling: RTL model construction is the final process of all hardware designs of embedded system. In this process, transformation from TLM model into RTL model is conducted.

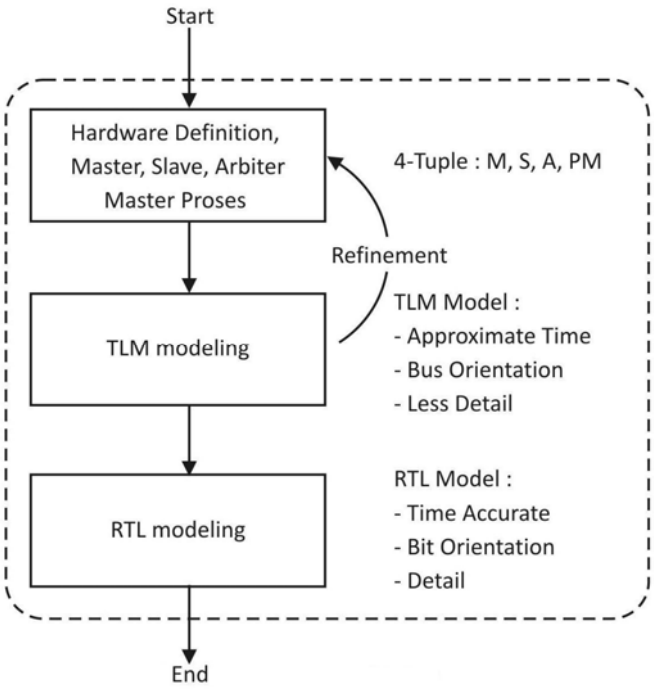

Figure 5. TLM-RTL design methodology

C. Procedure of Modeling Diagram Block

Basic procedure of modeling is designed as standard process on hardware modeling. Modeling steps of the new design methodology are:

1. Define: 4-tuple input (M, S, A, PM).
2. A module with port and method is made for each master.
3. A module with port and method is made for each slave.
4. An arbiter bus is made with algorithm in A.
5. Every method in master and slave is defined in PM.
6. Early verification of system requirement compliance
7. If system requirement is not satisfying, then perform tuple refinement starting from step 1.
8. Adding port and RTL process
9. Port and process removal from TLM.
10. RTL arbitrary implementation.
11. Port mapping

Stages 1 to 6 are initial stage of transaction level model creation for the purpose of early verification of hardware modeling. The first process output is a TLM model that fulfill the design requirements. Stages 8-11 are the RTL implementation.

1) Diagram Block

Diagram block is a diagram that shows inputs and outputs of the system. Inputs of diagram block of transaction level model include :

- Master: Number of master component actively perform read() and write() process as standard operation of components
- Slave: Number of slave components considered passive components and waiting for transaction of master.
- Arbiter: bus management system, namely mutual access management algorithm of one slave with one master or more.
- PM: Process taking a place in master and slave such as read() and write() process.
- Number of modules: Total the whole main components existing in a system including Arbiter.
- Specification is system requirement explanation that should be met by the system being designed.

Output of design system block is a TLM Model. Model formulation process is conducted systematically.

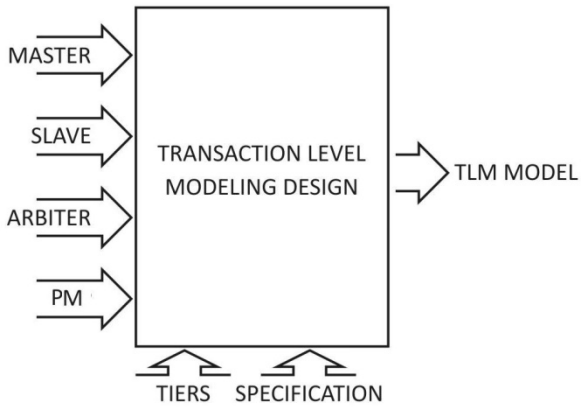

Figure 6. Diagram Block

2). *Defining Master and Slave*

- Master and slave component definition consists of three parts; name, port, and functions/method. Example of master:

Name : MicroPro

Port :
int mydata, input1, input2, input3;
int cnt = 0;
unsigned int addr = m_start_address;

Function/Method : Do_add(); Memory_Read(); Memory_write();

- Arbiter is bus management algorithm, such as: round robin, strict priority based, etc.
- PM is a process in master. PM is the more detail definition of in the form of pseudo code. Example greatest common divisor (GCD) process.

In transaction level modeling, data transfer process and control from master to slave are conducted by accessing a bus controlled by an arbiter. Each master can deliver request of bus access to send data or read data from slave. There will be several possible conditions achieved by master. These conditions are “bus condition is OK” if the bus is not being operated by other masters or “WAIT condition” when the bus is being used by other master or “ERROR condition” when the targeted slave is not around in the slave list.

3). *TLM – RTL Transformation*

After finishing early verification process and being met with given specification, then the last stage is transforming the model from TLM into RTL. The purpose of the transformation is to generate detail model available for synthesis. Phases of TLM into RTL model transformation can be divided into several general stages; those stages are:

- Port addition and deletion: in the TLM modeling process, there are ports that have to be deleted, because it is not needed in RTL model, such as port request. Meanwhile, it is necessary to add new ports in RTL model for performing detail process, as the nature of RTL modeling.
- Process addition and deletion: beside of ports addition and deletion, it is also necessary to add and delete process. Example of process that must be deleted from TLM is such process that tries to send request, while addition process that should be given in RTL model is process of accessing multiplexer (for example).
- Total Master and Slave determination: Total master and slave information is used to make the pattern of RTL bus. Total master and slave can influence total multiplexers and types of multiplexer. Multiplexer for 4 masters applies the first mux4 while 2 masters apply the first mux2.
- Determining arbitrary algorithm (according to given protocol). This algorithm manages the access when multiple masters try to access bus or slaves. Example of algorithm used is round robin, such as in Avalon bus.
- Port mapping: The last stage of transformation is connecting all ports from all components available along with additional components, such as multiplexer, detail, pin-per-pin.

4). Examples of TLM-RTL Transformation

The followings are examples of transformation from TLM to RTL by Wishbone bus.

Bus target: **Wishbone**

1. Port addition and deletion:
   - Sc_in_clk clock; (added)
   - Sc_port<sc_signal_out_if<bool> > grantM1; (deleted)
2. Process addition and deletion:
   - Sel_mux_master1 (added)
   - grantM1process (deleted)
3. Total Master and Slave determination:
   - Determining the amount of rows added and reduced in all systems.
4. Arbitrary determination
   - Wishbone protocol arbitrary is Round robin
   - Every master sends request of slave access. If there are several masters requesting access of one similar slave, then the arbiter will give an access for the master and send waiting signal for other masters.
5. Port mapping of all modules: master and slave
   - Mapping of master post to all multiplexers.
   - Mapping of multiplexer post to slave and the master.

3. Model Transformation Method

A. Transformation Method – TMC Interfacing

Modeling level transformation refers to the transformation process from a model of certain level to a model of different level. The system consists of several inter-communicated components. Each component block communicates with the other component blocks.

Transformation process of TLM level to RTL level is performed by turning TLM components and bus into RTL. In the complex system, this transformation needs a systematic method, so transformation process can be performed faster. Method applied in this paper is incremental model transformation.

Incremental model transformation is a process to transform of TLM model to RTL one, by transforming block per block component. By theses model transformation phases, it allows to find error in the initial phase so that the transformation process of TLM – RTL model can be performed faster than the entire model transformation. This transformation is performed by applying two kinds of models simultaneously. System model designed in the entire TLM model takes a place in the initial condition. In the next phase, the model will be transformed into RTL model by keeping on maintaining other parts in existing TLM model. Through such process, error investigation of RTL model can be conducted at the same time as transformation process. The transformation process then occurs to the other parts of TLM models until the models are eventually turned into RTL model.

It is necessary to prepare the communicator which communicates the data between two models with different level of abstraction. Such communicator is well known as TR-Model Communicator (TMC) – stands for TLM RTL Model Communicator – which serves to communicate data in different models.

TR-Model Communicator (TMC) refers to the module that connects the parts of TLM-RTL models. The module will be used to communicate data from TLM to RTL model and vice versa. The motivation of TMC design is to bridge TLM modeling with predefined RTL models. The utilization of predefined RTL model which has been widely used is to allows accelerating the design process of TLM level.

The simplest communication model of system comprises of two components consisting of master and slave. TR-Model Communicator is used to translate the TLM data to more detailed RTL signal. Figure 7 illustrates the process of TR-Model Communicator that connects a TLM master to RTL slave.

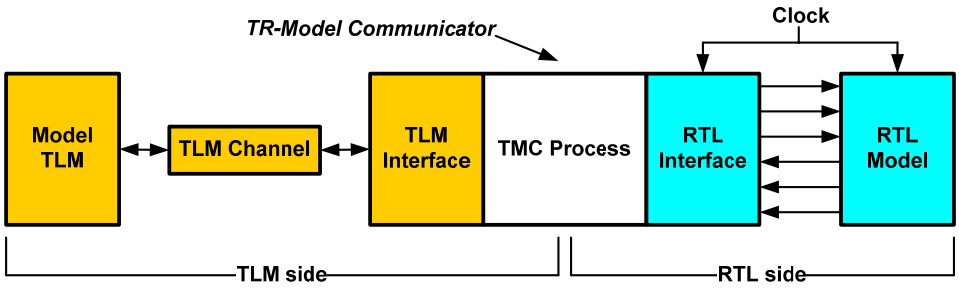

Figure 7. TMC Specification

TR-Model Communicator is built for the purpose of shared TLM and RTL modeling. This module is related to two different models. The process in TR-Model Communicator is translating TLM signal to more detailed RTL signal and vice versa.

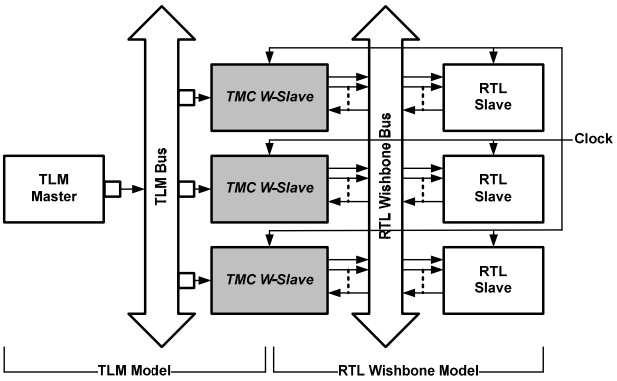

Figure 8. TMC W-Slave

It requires several TR-Model Communicators to connect master and slave complex systems consisting of multiple masters and slaves.

TR-Model Communicator serving as the connector of two models comprises of two interfaces: TLM and RTL interfaces. The function of each interface is to connect TR-Model Communicator to TLM and RTL model. Figure 7. shows both interfaces existing in TR-Model Communicator.

The parameter used to measure the success of TMC is indicated as follow:

1. The correctness of TR-Model Communicator in connecting RTL to TLM model.
2. The correctness of system functionality. The use of TR-Model Communicator in communicating TLM and RTL model must give the same functional outcome in the simulation.

There are three main functions in TMC :

1. To communicate with TLM model (TLM interface). In this process, TMC communicates and accommodates all functions existing in TLM model.
2. To communicate with RTL model (RTL interface). In this process, TMC communicates and accommodates bit per bit communication with RTL model.
3. Transaction transformation process as a core process of TMC. In this process, TMC performs transformation from transaction function existing in TLM interface that is translated into transaction bits in the interface of RTL model.

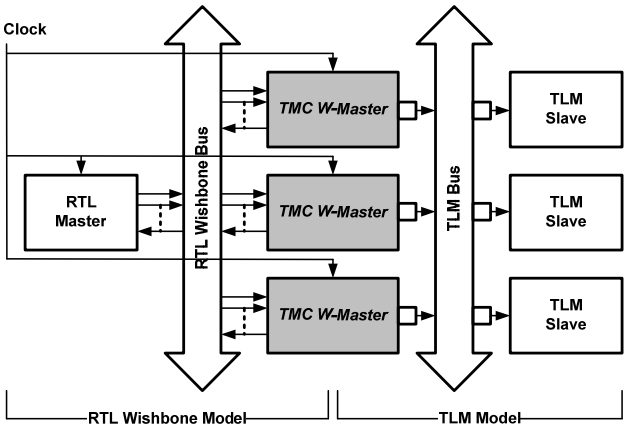

Figure 9. TMC W-Master

Based on type of bus existing in RTL, TMC can then be divided into two categorizations:

1. Standard TMC : TMC that communicates TLM model to standard RTL bus such as: Avalon, Wishbone, AMBA, OCP, etc.
2. Non-standard TMC : TMC that communicates TLM model to RTL model which does not apply standard bus. TMC can be customized as the need of built RTL model.

B. TML-RTL Prototyping Model Design (PMD)

PMD method refers to transformation process from TLM model to RTL model in incremental manner by using TMC. The emphasize on PMD method will be during model verification process performed in every phase of transformation. By using this method, the correction can be performed in every phase if problem/bug is found inappropriateness between TLM and RTL model. The prototype correction can improve the acceleration of TLM-RTL transformation process.

Verifications in each transformation stage are required in PMD TR method. The verifications will allow us to assure that both different models can perform transaction in accordance to the system specification. In the final phase, all models have been turned into RTL model.

PMD method is more beneficial because there is a model verification on every stages. It is different from brute force model transformation that needs more complex verification of all parts of model. In PMD method, all parts of the model are verified in incremental process, block-per-block verification, until all models are transformed from TLM into RTL model.

The PMD concept can provides us with modular transformation model by utilizing TMC module as communicator among the models of different levels. Basically, this process requires time more efficient than that needed by the brute force model transformation. Figure 10. shows the example of transformation stages from TLM model to RTL model.

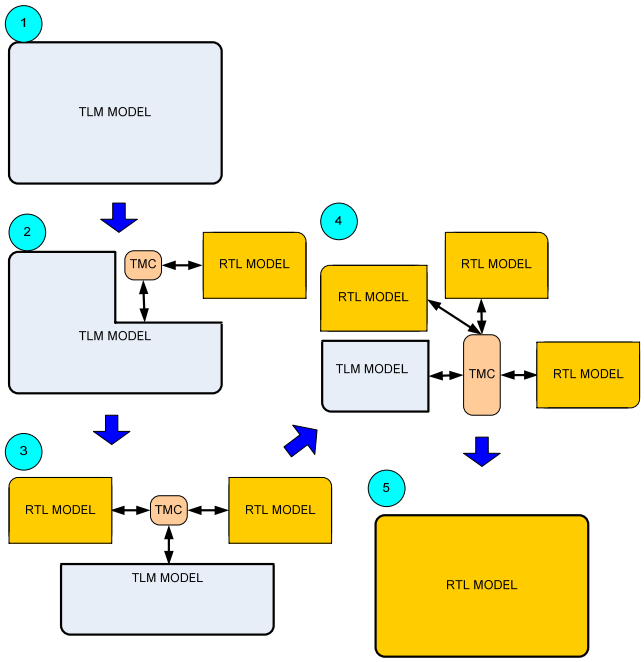

Figure 10. TLM-RTL Prototyping Model Design Steps

4. Experiment and Results Analysis

A. Wishbone SoC Bus

Wishbone bus is an SoC bus. The bus is designed for chip-based application. SoC is a compact system with three kind components, master and slave components and bus system.

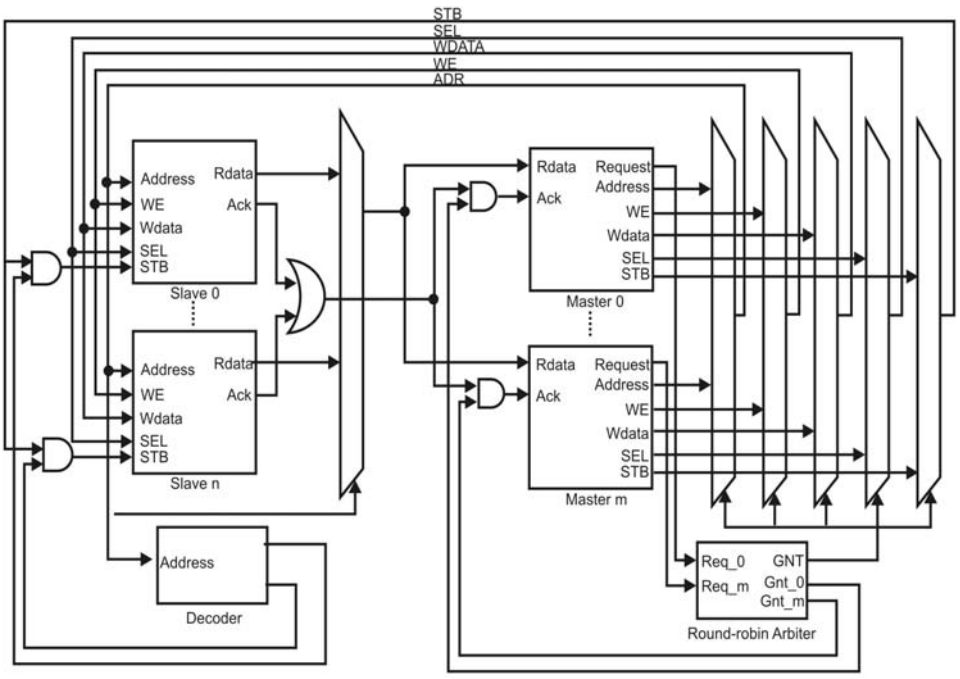

Figure 11. Wishbone bus architecture

In our experiment, Wishbone bus is used in implementation stage in the level of RTL. There are 5 main components in Wishbone bus along with each function as follows:

1. Master: active components which have initiative to perform data access either read() or write().
2. Slave: passive component waiting for data access from master.
3. Logic Request: components managing access requests from master to slave. Each component has one logic request component.
4. Logic Arbitrator: component managing access of one slave according to request of one master of more. Each slave has one Logic Arbitrator to manage the slave access.
5. Multiplexer: component for managing access of a slave according to request of Logic Arbitrator. There are 5 multiplexers for each slave; mux address, mux BE_n, mux write, mux writedata and mux read.

B. Case Study : Greatest Common Divisor Processor

In this case study, a system consisting of several master and slave components is designed. It is designed by using TLM modeling. By making use of TMC Wishbone, each sub system will be transformed into RTL model. In the last stage of modeling, the system will have been modeled in RTL model.

```
0: int  x, y;
1: while (1) {
2:   while (!go_i);
3:   x = x_i;
4:   y = y_i;
5:   while  (x != y)  {
6:       if  (x < y)
7:          y = y - x;
         else
8:          x = x - y;
       }
9:   d_o = x;
    }
```

Figure 12. GCD Algorithm

The built system is realized in form of processor. The function is to count GCD (Greatest Common Divisor). It comprises of five components. Those are three master components and two slave components. The first master will serve as input master which generate numbers that will be counted. The second master is called processor master. This component reads data from memory, counts GCD, and brings the value back into the memory. The third master is called output master. This component reads data from memory and displays it. Slave comprises of two memories; one for input and output data storage and another to be displayed to output master.

Master comprises of GCD processor, DMA controller input, DMA controller output. Slave comprises of RAM Memory, secondary Memory. Each component has its own function as provided below:

1. GCD processor is the main processor used to count GCD.
2. DMA controller input is used to enter the data from external system into secondary memory.
3. DMA controller output is used to display data taken from RAM Memory.
4. RAM Memory is used to store temporary data.
5. Secondary memory is used to store permanent data.

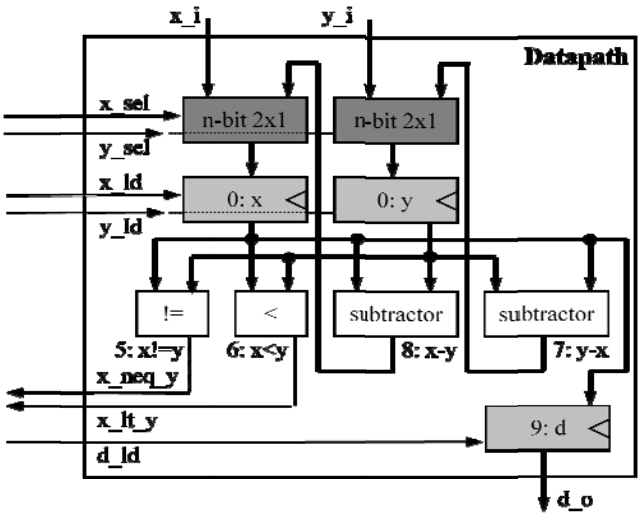

Figure 13. GCD Datapath

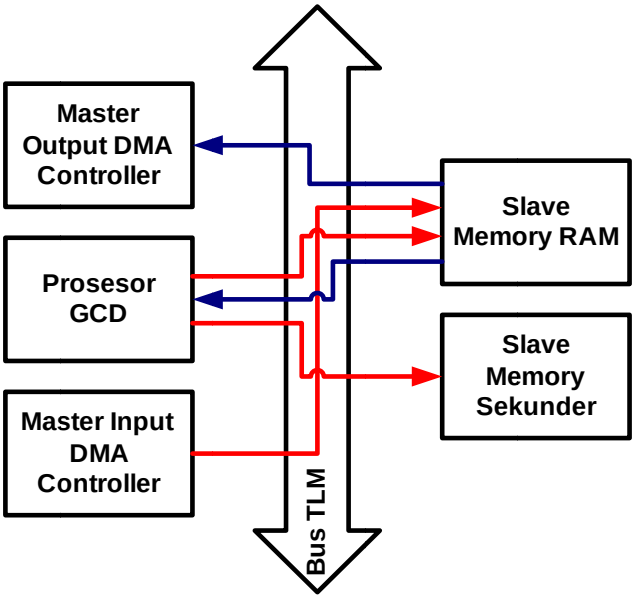

Figure 14. GCD System Model

Greatest common divisor refers to the biggest number that divides two numbers without resulting in division remaining. The input of the system will be in the form of two numbers that will be counted, whereas the output will be in the form of number which is the greatest common divisor (GCD).

State Machine Diagram: The first stage to do in designing hardware is the creation of state machine diagram. Such diagram can be obtained by translating algorithm into the states that will eventually transformed into datapath.

The datapath refers to the system which has function to process GCD in accordance with predetermined specification. Rules in component creation in datapath are shown as follow:

1. Defining register for every declared variables
2. Defining functional unit for every arithmetic operation (adder, modulo, comparator, or)
3. Connecting port, register, and functional unit
4. Defining the unique identifier

GCD processor performs numbers calculation process in order to discover greatest common divisor. The data is taken from RAM memory in form of two numbers. The numbers will periodically be sent from DMA controller (master input). GCD calculation result will furthermore be stored in Slave Memory RAM and secondary Memory. Master output will take the data form slave memory RAM to display in peripheral output.

The components existing in the experimentation are below:

1. Master Input DMA Controller refers to a master that can periodically sent two numbers to slave memory RAM. The numbers are included in system input that will be processed by GCD processor in the sequent stage.
2. GCD processor is the master that performs GCD calculation process in accordance with the input obtained from slave memory RAM. GCD calculation result will then be stored in two sites; one in slave memory RAM for the next process and another in secondary slave memory to permanently be stored.
3. Master output DMA controller is a master that periodically copy data from slave memory RAM and display the data to peripheral output device.
4. Slave memory RAM is the component to store the temporary data before being processed and displayed.
5. Secondary slave memory is the component to permanently store data to be used in the other processes.

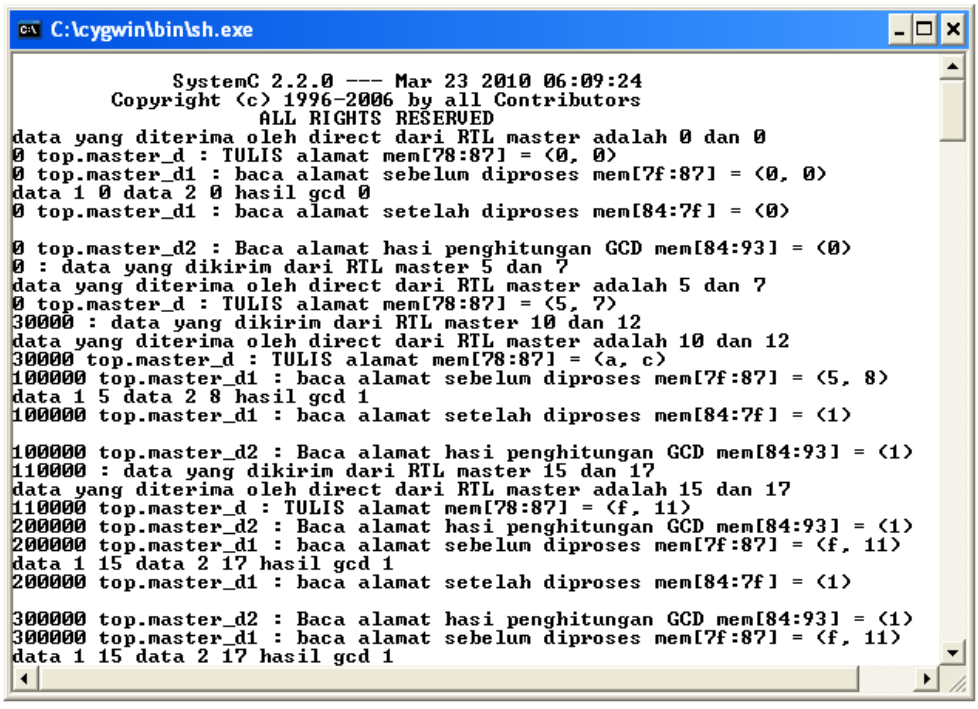

Figure 15. TLM Process

C. Model Transformation Testing

Transformation process of TLM into RTL model is shown in the following section. The testing result is achieved by comparing functional testing between system using TLM and one using RTL in incremental manner. The verification is conducted in every stage in order to facilitate error discovery in every stage. Testing process is conducted by way of generating input vector sequence by master intput DMA controller.

The data will then be stored in slave memory RAM to be processed by GCD processor. The final result of calculation process is displayed by master output DMA controller. During the testing, the similar process is also performed for the system which uses bus TLM of TLM model with TLM-RTL combination model by using TMC component as the connector of both different models.

The testing result indicates that the system has functioned in accordance with functional specification shown in calculation of GCD(5.7) and GCD(15.17) in 1 of process result.

Transformation Result

Testing result of GCD processor of post transformation process by adding TMC component that connects to TLM and RTL models has shown the functional amount similar to system specification. Experimentation result shows that GCD(5.8) is 1 and GCD (20.22) is 2.

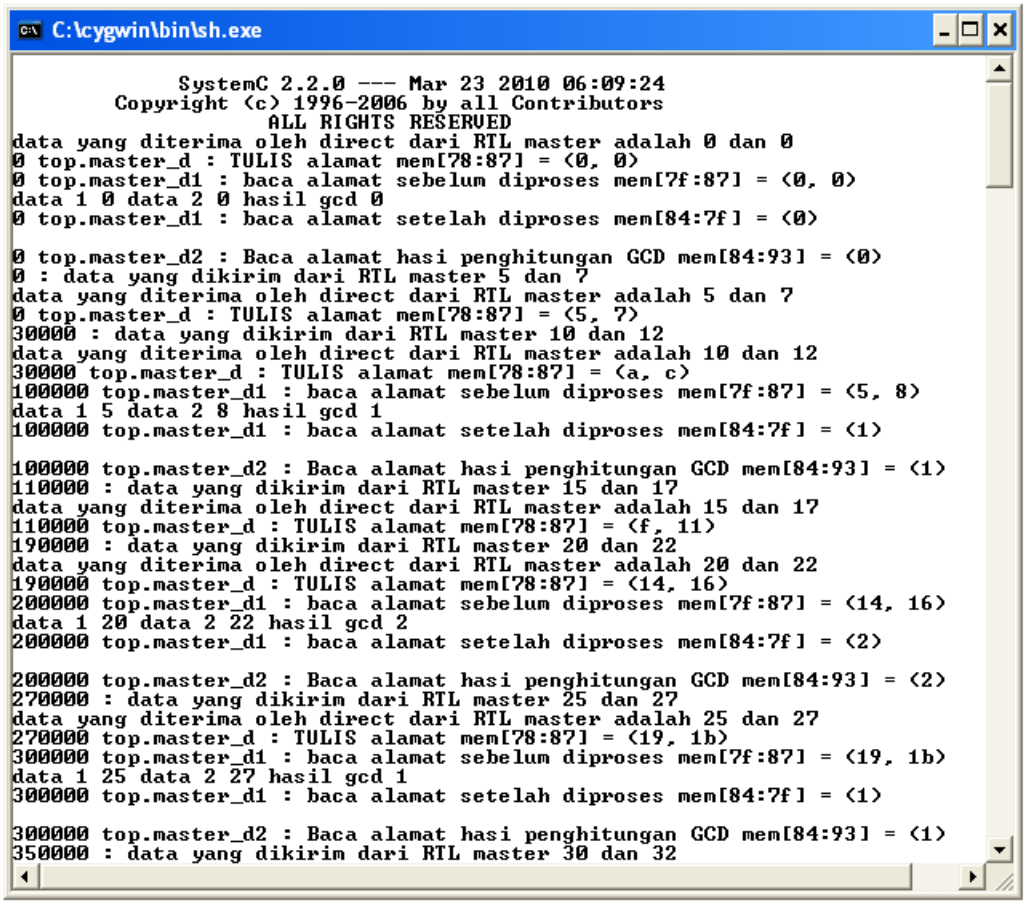

Figure 8. TLM-RTL Process

The testing result has indicated that, functionally,the transformation process has been successfully conducted by comparing both simulation result of TLM and RTL.

The testing result has also suggested that TMC usage as the connector of both different models has successfully performed in accordance with predesigned specification.

5. Conclusion

Based on the testing shown in the previous chapter, it can be concluded that the new method of transforming TLM model to RTL model can be used to increase design process performance. It means that using this method the transformation process need shorter processing time. Contribution of this paper is proposing the new method for transforming TLM model to RTL model based on rapid prototyping verification process. This method utilizes TR-Model Communicator (TMC) as a module that communicates between two different models. TMC transfers data from TLM model to RTL model and vice versa. The system can be verified on every transformation stage. This is the advantage of incremental model transformation method, so can increase the design process performance.

The future work of this research is making a tool for automatic model transformation based on incremental transformation method.

Acknowledgment

Arif Sasongko thanks to STEI of Bandung Institute of Technology for their support and research resources so this research could be advancing. Maman Abdurohman thanks to IT Telkom for their financial support.

References

[1] Berger, Arnold S. “Embedded System Design : An Introduction to Processes, Tools, and Techniques”. CMP Books. 2002.

[2] Bombieri, N. Fummi, F. dan Pravadelli, G. (2007) : Incremental ABV for Functional Validation of TL-to_RTL Design Refinement. DATE 2007.

[3] Bombieri, N., Deganello, N. dan Fummi, F. (2008) : Integrating RTL Ips into TLM Designs Through Automatic Transactor Generation. DATE 2008.

[4] Chatha, Karamvir Sigh. “System-Level Cosynthesis of Transformative Application for Heterogeneous Hardware-Software Architecture”. Dissertation at University of Cincinnati. 2001.

[5] Cornet, Jerome. “Separation of Functional and Non-Functional Aspects in Transactional Level Models of Systems-on-Chip”. Dissertation at Institut Polytechnique De Grenoble. 2008.

[6] Cummings, Clifford. “SystemVerilog’s priority & Unique – A Solution to Verilog’s full_case & parallel_case Evil Twins”. SNUG. Israel. 2005.

[7] Frank Vahid and Tony Givargis. “Embedded system A Unified Hardware/Software Introduction”. JohnWiley & Sons, Inc., New York, 2002.

[8] Genovese, Matt. ”A Quick-Start Guide for Learning SystemC”. The University of Texas. Austin. 2004. 15

[9] Gordon E. Moore. “Cramming more components onto integrated circuits”. Electronics, 38(8):114-117, 19 April 1965.

[10] Leung, Julie. Kern, Keith. Dawson, Jeremy. “Genetic Algorithms and Evolution Strategies”.

[11] Mathaikutty, D., A. (2007) : Metamodeling Driven IP Reuse for System-on-chip Integration and Microprocessor Design, Dissertation at Virginia Polytechnic Institute and State University.

[12] Mooney III, Vincent John. “Hardware/Software co-design of run-time systems”. Dissertation at Stanford University. 1998.

[13] Palnitkar, Samir. “Verilog® HDL: A Guide to Digital Design and Synthesis, Second Edition”. Sun Microsystems. Inc. California. 2003.

[14] Patel, Hiren D. “Ingredients for Successful System Level Automation & Design Methodology”. Dissertation at Virginia Polytechnic Institute and State University. 2007.
[15] _____, “Ptolemy II Project”. UC. Berkeley. 2008.
[16] _____. (2002) : Describing Synthesizable RTL in SystemC™, Synopsys, Inc., Version 1.2, November 2002. www.synopsys.com
[17] _____. (2003) : Avalon Bus Spesification : Reference Manual, Altera. : www.altera.com

Arif Sasongko is a lecturer at School of Electrical Engineering and Electronics (STEI) of Bandung Institute of Technology. His major areas of interest include cryptography, telecmomunications electronics and embedded system design. His current project is designing highspeed data link wimax system. Contact him at asasongko@gmail.com

Maman Abdurohman is a lecturer at Telkom Institute of Technology. He is working at faculty of Informatics of Telecom Institute of Techonolgy – Bandung. His primary areas of interest include embedded system design and microcontroller. Maman has an master degree from Bandung Institute of Technology. Contact him at mma@ittelkom.ac.id

Kuspriyanto is a Professor at STEI of Bandung Institute of Technology. He is a senior lecturer in computer engineering laboratory. His major areas of interest include digital system and electronic design. His current position is Head of Laboratory of Computer Engineering. Contact him at kuspriyanto@yahoo.com

Sarwono Sutikno is an Associate Profesor at STEI of Bandung Institute of Technology. His major areas of interest include cryptography and embedded system design. His current job is a director in PPATK on electronic transaction control. Contact him at ssarwono@gmail.com