ActaEnergetica
POWER ENGINEERING QUARTERLY

Designing Architecture of Embedded System Design using HDL Method

Nuria Rabanal
PhD, The University of Leon, Leon, Spain

Prof. Dharmesh Dhabliya
Professor, Computer Engineering Department,
Vishwakarma Institute of Information Technology,
Pune, India
dharmesh.dhabliya@viit.ac.in

Keywords
TLM and RTL Model, Embedded System, Design

Abstract
the vendors and researchers are under pressure to enhance the embedded system design methodology due to the time-to-market pressure and productivity gap. Register Transfer Level (RTL), the conventional design method, is inadequate for the needs of embedded system design. A different approach is required to deal with the RTL problem. In this paper, we propose a new approach to hardware embedded system modeling that makes use of Transaction Level modeling to boost the efficiency of the design process (TLM). When it comes to design abstraction levels, TLM is the next step up after RTL. Time to design and design precision are two of the metrics tracked. Avalon and Wishbone buses, both System on Chip buses, were utilized in the actualization of the RTL model.TLM and RTL model improvement performance metrics. The results of the experiments show that Avalon RTL's performance is improved by a factor of 1.03 for 3-tiers, 1.47 for 4-tiers, and 1.69 for 5-tiers when the new design methodology is used.Wishbone RTL increases performance by a factor of 1.12 on a 3-tier system, 1.17 on a 4-tier system, and 1.34 on a 5-tier system. The pattern of these outcomes demonstrates a rising standard of design practice.

Received by the editor: 29.03.2022
Received in revised form: 14.06.2022
Accepted: 30.06.2022

1. INTRODUCTION

Since the turn of the century, ICT (Information and Communication Technology) has undergone unprecedented development and growth in virtually every area. Multiple public and private sectors around the globe have launched successful e-services initiatives. Closer inspection reveals that most of these prosperous programmes are concentrated in metropolitan regions, while the rural areas fare much worse. This is because of a wide variety of commercial and technical constraints. For an e-services initiative to be successful, one of the primary technical obstacles are the creation of an effective application platform well-suited to a rural environment. The needs of a rural application platform can be taken into account when designing an embedded system.

Although the term "embedded system" can mean a number of different things, the most common usage refers to a system in which the hardware and/or software are both integral parts. It's a computer setup that serves only one programme or one product. It could be a standalone system or a subsystem of something larger. It does not require additional memories like a computer does because

its software is typically embedded in ROM (Read Only Memory). Different types of embedded systems exist, each with its own set of characteristics and uses. Limited scope (or low end) installed frameworks, medium scale inserted frameworks, and top of the line implanted frameworks are the three primary classes of installed framework. These limited scale frameworks are regularly worked around a solitary 8-or 16-bit microcontroller, and subsequently, they have smoothed out equipment and programming, depend on ready memory (ROM and Slam), and come up short on working framework and different simultaneous strings of execution. These use very little energy and may even be powered by batteries. Embedded systems on a medium scale are typically programmed on a single or multiple microcontrollers, DSPs, or microprocessors with a size of 16 or 32 bits.It's not just the software that's complicated, but the hardware as well. Most of these also feature removable storage. Depending on the system's requirements, a cluster of processors, configurable processors, or programmable logic arrays may be required to handle the hardware and software complexity of high-end embedded systems.These are employed in elaborate programmes that necessitate joint hardware/software development and integration.In addition to internal memory, these high-end computers will also feature external, high-speed RAMs (such as DDRs) that add several megabytes of memory to the system. It is normal practice to insert specific capabilities — like cryptographic calculations, realistic handling calculations, video decoders, discrete cosine change and reverse change calculations, Organization conventions, and organization driver capabilities — into equipment (through particular co-handling units) for speedier generally speaking activity. These can associate with top of the line show subsystems and run undeniable working frameworks (now and again ongoing operating system). In view of the rising computational refinement of current applications, this subset of implanted frameworks has drawn in a lot of consideration and concentrate lately.

The term "application platform" refers to a software infrastructure that is used to host and run applications. The application platform provides everything that is needed to successfully run or execute an application. Hardware, device drives, operating systems, and the like typically fall into this category. The application platform's precise set of features and functions will vary with each unique programme. E-services, for example, need features like video and graphics playback, network connectivity, and display support. Furthermore, the outcome of a framework is reliant upon the accessibility of a suitable application stage for a predefined biological system of sending. Advancement is troublesome in light of the fact that there are numerous boundaries of an application stage and there are clashing necessities.

A. Evolution of Embedded System From Past To The Present

Although there are a theoretically infinite number of embedded systems possible, they all operate according to the same basic principles of system components and design methodologies.

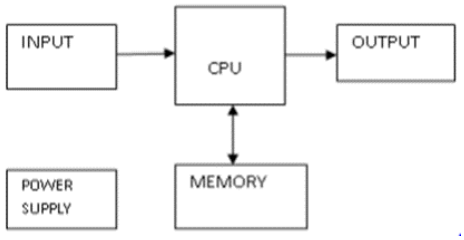


Fig.1. Generic Block Diagram of Embedded System

To give only one model, each installed framework integrates a processor and programming to work that processor. It's conceivable that a microcontroller, or even a chip, is the cerebrums of the activity.In addition, ROM and RAM are required to store the executable code and temporary storage for run-time data manipulations, respectively. If the amount of data to be stored is low, the memory modules could share a chip with the CPU. In most microcontrollers, you'll find such a setup. Without internal memory, one or both types of data will need to be stored on external memory chips. In addition, all embedded systems have inputs and outputs of some kind. Sensors, probes, signals, and buttons make up the majority of the system's inputs. In most cases, the output is a visual representation, a communication signal, or a modification of the physical environment.

Computer chip (which can be a microchip, microcontroller, DSP, or blends thereof), memory, and information/yield subsystems make up an installed framework. All embedded system parts had been upgraded at an exponential rate.The

Apollo Guidance Computer was an early example of an embedded system that received widespread attention. Designed in 1966 by Charles Stark Draper of the MIT Instrumentation Laboratory. The Intel 4004 was the first calculator microprocessor.

B. Design Challenges

Concerning embedded systems for rural application platforms, we discuss here the difficulties of design, of resolving competing requirements, and of optimizing for performance. There are significant design challenges involved in meeting the requirements. Table 1 displays a number of requirements, some of which are in direct opposition to one another. A good example would be the correlation between increased computing power and corresponding increases in energy consumption. Power usage is greater for the high-computing CPUs compared to the low-computing CPUs. Pentium 4 uses about 60W and has 9700 MIPS. An Atom D510 consumes around 13 W and can process 8400 MIPS, while an ARM Cortex A8 can process 2000 MIPS while using less than a watt of power [13,14]. There is a lack of mechanical and electrical stability in desktop computers. The hard drive spins; there are slots for additional memory, and so on. Such components and layouts are unacceptable in an embedded system. The hard drive's mechanical moving parts lessen the system's mechanical ruggedness; RAM slots are provided in desktop systems to allow for future memory upgrades; however, this design cannot be used in an embedded system because installing RAM into slots is not a mechanically strong arrangement. Given that in-field upgrades are not an option, the amount of memory needed in an embedded system must be determined in advance. According to our analysis, there is likely no feasible system that would meet all of the requirements simultaneously. Therefore, it is crucial to give proper weight to the relevant parameters / requirements. The relative importance of each factor is determined by the requirements of the given task and the nature of the product or service being developed. Power consumption may not always be the most important factor when making a purchasing decision. For this reason, there can be no universally applicable criteria for evaluation; rather, these must be determined individually for each case. The difficulty comes from having to carefully consider all of the variables and make sound engineering tradeoffs.

C. Embedded system design

Configuration stream of implanted framework starts with plan detail, its characterize framework requirement, both expense and handling time. Framework usefulness is characterized in social portrayal, equipment programming dividing is finished to streamline configuration result nevertheless fit the necessity. Equipment and programming incorporation is finished after equipment/programming point of interest plan. Register move level plan is completed by implies equipment programming language, for example, Verilog, VHDL and Esterel. Check and testing process is finished to guarantee installed framework configuration is fit to determination

2. REVIEW OF LITERATURE

Ptolemy is an endeavor made at the School California, Berkeley. The latest Ptolemy Conveyance is Ptolemy II 7.0.1 that has been shipped off u 4 April 2008. Ptolemy is a framework for multiplication, model, and mix of programming that has been committed only to electronic sign taking care of (DSP).

The principal thought of Ptolemy is the use of a pre-described pay model that will coordinate cover parts affiliations. The key issue address by the Ptolemy is the usage of the mix of various replacement models. A piece of the model spaces that have been executed are: CT (constant time showing), DDF (dynamic dataflow), DE (discrete-event showing), FSM(finite state machines and particular model), PN(process networks with odd message passing), Meeting(process), networks with synchronous message passing, SDF (facilitated dataflow), SR (concurrent responsive), Remote

Ptolemy II contains supporting bundles like charts,gives the controls of Chart hypothesis, math, givesnumerical grids and vectors and sign handling,plots, gives visual information show, information, gives type framework,information wrapping and articulation parses. Ptolemy II bundlecontains the accompanying parts:

Ptolemy II C Code Age: The essential ability is to make codes for the SDF model, FSM and HDF: the entire model could be changed over into C Codes.

Ptalon: Is an entertainer situated planning addressing the mostnormally planning technique in an implanted frameworkplanning. This framework is regularly demonstrated as block outline,where a block presents framework or; lines or between block boltsaddressing signals.

Backtracking: This offices serve the capability to save thepast framework state values. The capability is the most basicin dispersed calculations.

Nonstop space: Constant Space is a redo ofNonstop Time spaces with fastidious semantics.

3. METHODOLOGY

A. Modeling At the Transaction Level

Exchange level-the hole between a simply utilitarian portrayal and a RTL model can connect through model. Following hardware/software partitioning, in which it is decided whether a certain processing operation will be carried out by a dedicated hardware block or by software, these are then created. Embedded software can be executed with TLM since it acts as a virtual chip (or virtual platform).

TLM's central concept is to model only the logical operations (reading, writing, etc.) performed by the buses, as opposed to all the wires that makes up the buses and their state changes. TLM models do not make use of clocks, in contrast to the RTL, where everything is synchronized on one or more clocks (synchronous description). They are inherently asynchronous, with synchronization taking place during the exchange of information between parts. These simplifications make it possible to run simulations much more quickly than with RTL.

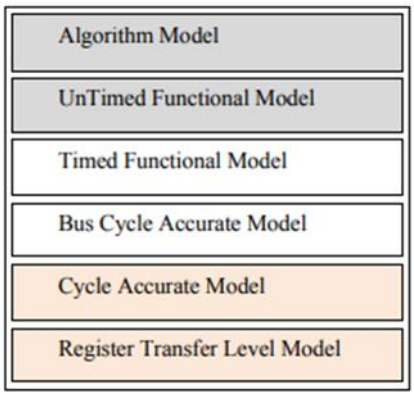


Fig.1. TLM Model Stack

TLM models likewise have the advantage of requiring essentially less demonstrating exertion than either RTL or the Cycle Exact model. While a working C/C++ code for the handling done by the equipment block to demonstrate as of now exists, how much work expected to address it is fundamentally diminished. To make a TL model, for example, you can reuse the source code from a current video decoder or computerized signal handling chain. This makes TLM the equipment's "brilliant model," as it very well might be executed, as opposed to a Cycle Exact model, which is not generally utilized once RTL has been created. A few executions of TLM even use timekeepers for synchronization, making them seem to be Cycle Exact level than the others. To convey data between an originator and a collector molecularly is the meaning of an exchange. The initiator is the one that steps up to the plate and complete the exchange, while the objective is constantly thought to be accessible to get it (or possibly to indicate to the initiator that it is occupied). This is in accordance with conventional thoughts in transport techniques. Through their respective "initiator" and "target" ports, senders and receivers of transactions send and receive data. One type of component may have just target ports, whereas another type may have only initiator ports. The initiator and the target ports are built into some components.

The bus protocol dictates the nature of the data transferred during a transaction. Some, however, are standard across the board for protocols:

- Whether information is read or written depends on the nature of the transaction.
- The address is a numeric identifier that specifies the desired part and the location of a register or other internal memory.
- The information being transmitted and received.
- Additional metadata, such as the return status (error, success, etc.), transaction time, bus properties, and so on (priority, etc).

All buses, and interconnection networks more generally, have the fundamental capability of sending transactions to their intended recipients in accordance with their addresses. The map of all the memory locations, which is used to determine where data will be sent, assigns a certain amount of

memory to each of the ports that will be the final destination.

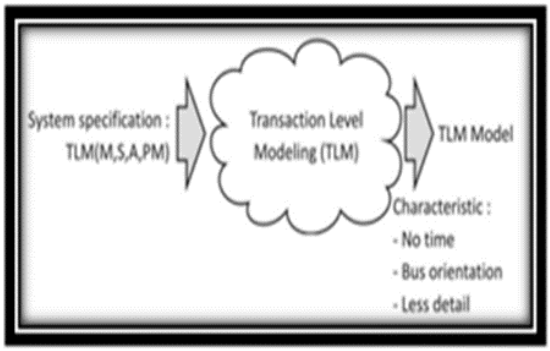


Fig.2. TLM process model

The address map and offset of each register must be identical to the finished chip for the embedded programme to run properly (register accuracy). In addition, all data generated and transmitted between the parts must conform (data accuracy). The last thing that has to happen is that the interruptions make sense with the ones that come after them. These specifications can be thought of as an agreement between the hardware and the embedded software. If the embedded software functions perfectly on the simulated platform, the final chip will function the same.

B. Hardware Embedded System Design Using A New Design Flow

In this work, we present a novel methodology for early verification based on transaction level modeling (TLM) of hardware embedded systems, which we use to develop a new design flow for modeling such systems. The verification procedure begins before the detailed design phase. Because of the success of register transfer level modeling, a new direction in embedded system design is toward transaction level modeling.

1. After dividing the hardware and software development processes, this study's focus has narrowed in on embedded hardware systems. In the third and last phase of detailed design,
2. The first step is to define the hardware itself, in this case the embedded system hardware that will be used.
3. TLM modeling, secondly, entails building models using a transaction modeling strategy and carrying out early verification. Correction of the tuple M, S, A, and PM can generate the process of model refinement.
4. Third, the RTL modelling: RTL model construction is the last step in all hardware designs of embedded systems. Here, the TLM model is converted into an RTL model.

	TLM	**RTL-WIS**	**RTL-AVL**
2-TIERS	228	362	263
3-TIERS	361	142	632
4-TIERS	290	861	267
5-TIERS	145	243	163

Table.1. TLM-RTL Model

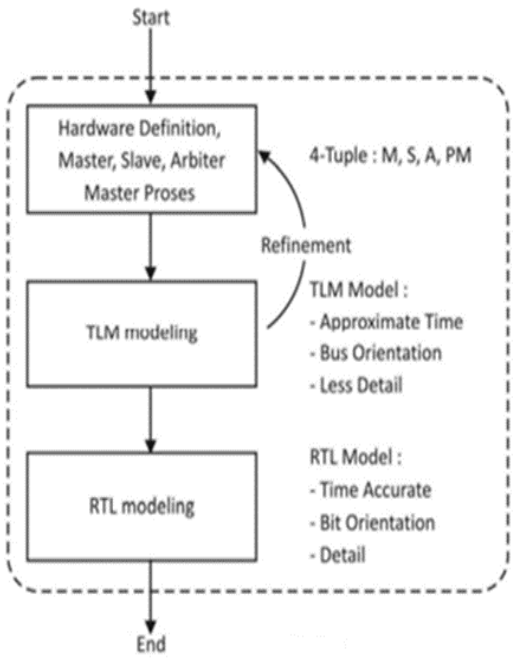


Fig.3. the new design methodology

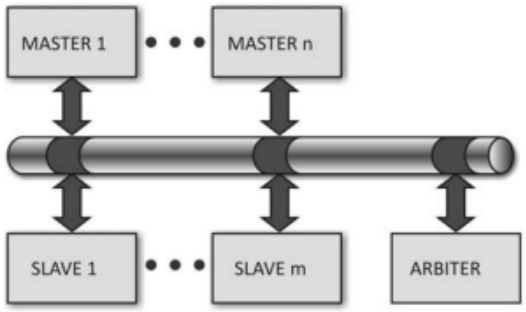
EXPERIMENT AND RESULTS ANALYSIS

A. Line of Code of TLM-RTL Model

Two, three, four, and five elements (master, slave, and arbiter) are used in the tests, respectively. The master creates and distributes information to the slave, whose role is to passively accept it.

Tiers are general terms of displaying embedded system components which communicate each other, for example, 2- tiers means there are two components communicating each others, 3-tiers

means there are 3 components communicating each others, and so on.



Experimental results point to TLM's advantage over RTL in terms of the number of lines required to model a system's architecture. This is possible because RTL provides more nuanced definitions of master, slave, and bus than TLM does.

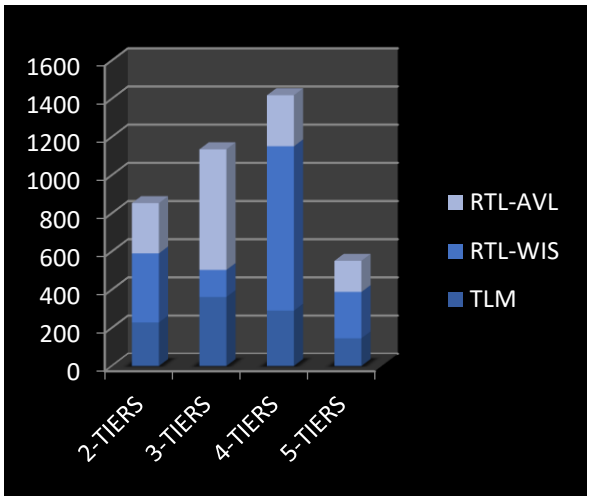


Fig.1 TLM-RTL Model

Several aspects of the software are susceptible to the extent of detail in which RTL is modeled.Definition of ports for both master and slave devices.

- Initial specification of system at the highest level, including port addition, instantaneity, port mapping, and port destruction.
- With this new definition of the multiplexor component, we have a brand new kind of electronic device. Six additional multiplexors must be installed on the Avalon bus for each new slave.
- TLM requires fewer instructions than RTL since its component definition procedure and port mapping are less complex. After a master is added, a new port mapping is implemented, which includes the bus and the clock.

	TLM	RTL-WIS	RTL-AVL
2-TIERS	26.5	27.3	26.3
3-TIERS	27.4	17.4	63.2
4-TIERS	16.97	27.4	26.7
5-TIERS	36.8	39.8	13.9

Table.2. RTL-WIS MODEL

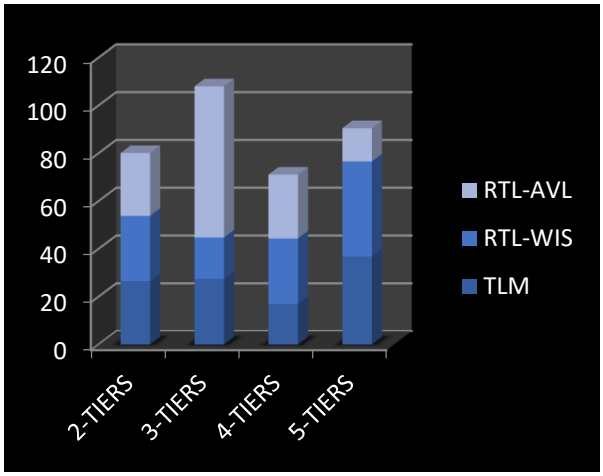


Fig.2. RTL-WIS MODEL

As a result, the design process will become more time-consuming due to an increase in the number of overall components and transactions in the modeling. Given the aforementioned, the design process on TLM modeling improves in the conditions wherein several components interact with one another. Such are the gains from implementing modeling at the transaction level using TLM techniques.

CONCLUSION

This paper's original contribution is a revised process for creating hardware RTL models, one that makes use of a transaction-level modeling approach and industry standards.The three new procedures will generate a new framework under embedded system design. The first framework is made up of a series of methodical embedded system building processes. The next step is to fully automate the processes involved.

TLM level demonstrating will be better executed in acomplex framework, in the state of more than

twoparts having communications in which there happenserratic interaction.

REFERENCES

1. F.Furtek. A field-programmable gate array for systolic computing. pages 183–199. Research on Integrated Systems: Proc. of the 1993 Symposium, G. Boriello and C. Ebeling, 1993.
2. D. T. Hoang. Searching genetic databases on splash2. pages 185–191. Proc. of IEEE Workshop on FPGAs for Custom Computing Machines, D.A. Buell and K.L. Pocek, 1993.
3. B.K. Fawcett. Applications of reconfigurable logic. pages 57–69. More FPGAs: Proc. of the 1993 International workshop on field-programmable logic and applications, W. Moore and W. Luk, 1993.
4. P.C. French and R.W.Taylor. A self–reconfiguring processor. pages 50–59. Proc. of IEEE Workshop on FPGAs for Custom Computing Machine, D.A. Buell and K.L. Pocek, 1993.
5. Swapna. C, Vijay Kumar. J, Raghavendra Rao. K, Ramanjappa.T,"FPGA Based Heart Beat Measurement System", International Conference on Information Technology, Electronics and communications (ICITEC), July 14 – 15, 2012, Hyderabad, India.
6. ArunaKommu, Raghavendra Rao Kanchi, "Design and development of a project-based embedded system laboratory using LPC1768", American Journal of Embedded Systems and Applications, Vol.1, No.2, November 2013. pp 46-53.
7. John Bowles and Gang Quan, "An FPGA – Based Embedded System Design Laboratory for the Undergraduate Computer Engineering Curriculum", American society for Engineering Education, AC2009-761.
8. Berger, Arnold S. "Embedded System Design: An Introduction to Processes, Tools, and Techniques". CMP Books. 2002.
9. P.I. Okwu and I.N. Onyeje "Ubiquitous Embedded Systems Revolution: Applications and Emerging Trends", International Journal of Engineering Research and Applications (IJERA) Vol. 3, Issue 4, Jul-Aug 2013, pp.610-616
10. Steve Heath "Embedded Systems design" 2nd ed.
11. highered.mheducation.com/sites/dl/free/00734045 6x/167 481/sample1.pdf –" Introduction to Embedded System"
12. Berger, Arnold S. "Embedded System Design : An Introduction to Processes, Tools, and Techniques". CMP Books. 2002.
13. Øystein Ra and Torbjø "Trends and Challenges in Embedded Systems – CoDeVer and HiBu Experiences".
14. http://training.linuxfoundation.org/free-linuxtraining/linux-training-videos. Linux device drivers, internals, Embedded Linux.