Chapter 1 : Introduction

1.1. Significance of Embedded Systems

Embedded systems are a part of our daily life and play an important role in various areas of modern society such as entertainment, banking, communication, home and office automation, transport, health, security, governance, education, etc. The areas benefiting from embedded systems are wide and steadily increasing. Embedded systems allow people to communicate all over, rapidly, and effortlessly, e.g., in autos, trains, and airplanes. Embedded systems speak to a chance for affordable healthcare systems. Recent improvements in medical gadgets range from medical imaging to pacemakers and artificial hearts. These systems are gaining importance due to rapid growth in automotive and other industries. Embedded systems are becoming increasingly used in emerging technologies like social networks and the Internet of Things (IoT). ARTEMIS Strategic Research Agenda 2011 assessed that “There will be over 40 billion devices worldwide, that is, 5 to 10 embedded devices per person on earth by 2020” [26]. This evolution has brought complete change in lifestyle in terms of retailing, connectivity, learning, working, and entertaining.

1.2. A Typical Embedded System

A system can be defined as an arrangement in which all its units are assembled and work together according to the set of instructions or program. A system is also called as one, having inputs and outputs [28, 57]. Figure 1.1 shows the view of a general system. Amplified RF pulse signal is applied to a magnet through the duplexer and resonance signal received from magnet to receiver after excitation. Signal generator includes a synthesizer and digital to analog converter. Receiver includes analog to digital converter and signal conditioning circuit. Analog is converted to digital signal with proper signal conditioning and is processed by the signal-processing unit. Computer is used here to provide user interface to the system. Processing units are utilized for image construction and signal processing and control the operation of all units within the system. Embedded system is one in which the computer is an integral part of the system and is hidden from the external view. Embedded systems are those having the characteristics of performing a specific task that is tightly constrained and interactive.
The example shown in Figure 1.1 is also an embedded system because it has the performing a particular imaging task. A typical embedded system consists of inputs and outputs, processing units, memories, communication units, operating system components, and application software or algorithms as shown in Figure 1.2.
Input and Output Units

Data acquired from the external world to the system through the input ports. Examples of input devices are touch screen, keypad, sensors, receiver, network card, fax, and modem. Byte information from the system is sent to the external world through output ports. LED, LCD, touch screen display, printer, robot, network and various motors are examples of output devices. Each port is represented by a unique address. Certain ports can be used as both input and output port, for example, a wireless communication channel mobile touch-screen. Timer sends timeout interrupts when the present time equals the preset time. Each action generates an interrupt. An interrupt can be generated when transmitting buffer empty. Set of instructions in memory are fetched and executed by processing units.

Processing Units

Processing units can be a single processor or multi-core. The processing unit is the heart of the system that controls the activity of other units. It is the hardware that processes the software and brings life to the embedded system. Processing unit consists of ALU, pipeline stages, superscalar units. GPU, ARM, and ASIP Processing units are identified by their properties such as instruction set, bit size, clock frequency, and speed.

Memories

Memories are storage of data and instruction sets. A processor will fetch and execute the program which is stored in the memory. Furthermore, the data which is in the form of variables, execution results, execution monitoring details, and other data created during processor execution is stored in the memory [27]. RAM, ROM, Cache, are the examples of memories. The operation of getting and storing instruction or data from or to the memory is called memory access. Time for accessing data is called memory access time [35]. If the data or instruction to be accessed from memory is available, then it is called ‘hit’ otherwise it is called ‘miss.’

Communication Interfaces

Devices in the system are connected via the set of lines called bus and communicated with the communication protocol called bus arbitration. UART, PCI, USB, AMBA, AXI, and Ethernet
Protocols are examples of bus protocols. In serial communication, de-multiplexer takes input and transfers to the multiple channels, and multiplexers are used to receive digital inputs from the multiple channels then transferred to the selected channel.

**Application Software**

Application software consists of tasks. Set of instructions under execution is called a task. Different applications used in the system are handled by the operating system. It is the one of the software component of the system.

**Operating System**

Supervision and execution of an application is done by the operating system. Task scheduling, context switching, process communication, and resource management are the functions of an operating system.

1.3. **Embedded System Design**

Embedded system design is the art of selecting the right platform for the development of efficient embedded system [40]. This process starts with system specification, concept proofing, design exploration, implementation, testing, and evaluation as shown in Figure 1.3.

**System Specification**

In system specification stage, functional and nonfunctional requirements of the system are specified in the form of text.

**Concept Proofing**

The functionality of the system is modeled and verified by algorithmic tool (MATLAB). Once the functional and nonfunctional design requirements are decided then R&D engineer has to decide the right platform based on his/her experience.

**Design Exploration**

In this stage, entire system specification is partitioned into hardware and software components to meet the design requirements. Software components are the operating system, application
software, and software IPs. Hardware components can be microcontroller or microprocessor, memory, bus, and other hardware units. Embedded system designer has to build system’s architecture among available resources and applications. The careful decision has to be taken while selecting hardware and software parts, because even small variation in selecting from right design can cause system failures resulting in more re-spins.

![Conventional Embedded System Design Flow](image)

**Figure 1.3: Conventional Embedded System Design Flow**

**System Implementation**

In this stage, hardware-software components are implemented separately. Software component can be an existing IP (Intellectual Property) or can be developed using IDE (Integrated Development Environment). C/C++, JAVA, and assembly languages are used to develop the software part. Similarly hardware component can be either selected from available components or can be custom designed for a specific purpose. Using available components can reduce design and development efforts whereas custom designs provide efficient and optimized design. Custom hardware designs include the front-end and back-end design process. Steps required for front-end design are RTL design entry using HDL programming, functional verification, logic synthesis, hardware prototyping using FPGA. Back-end design steps are transistor design, SPICE simulation, layout design, design rule check, and place and route. Back-end design steps are
required to get the physical design. This layout design is sent through the fabrication process to get custom semiconductor IC. This semiconductor IC undergoes the process of packaging to provide mechanical support and to prevent the semiconductor circuit from external environment.

**Testing and Evaluation**

Testing is specifically used to debug the system functionality. Design for testability and boundary scanning techniques are used to debug the hardware failures. Similarly software testing is used to debug software errors. Evaluation is the process of examining the attainment of expected results in terms performance, power, and cost. Finally, Custom IC delivered as the end product. If performance, power and cost requirements are not met then redo the same procedure to start from specification to implementation.

**1.4. Embedded Systems Design Challenges**

In conventional approach, hardware-software partition and validate the system architecture by simply profiling and manually calculating design metrics from past experience. These design metrics are evaluated after implementation. Manual calculation prone to inaccurate selection of design results in system failures realized after the implementation requires re-spins. Number of re-spins can lead to multiple issues: delayed product launch and loss of revenues. This approach is expensive and can lead to inefficient design loop if further debugging or optimization is requested. Various challenges of embedded system design as shown in Figure 1.4.

The challenge of conventional approach is to evaluate and optimize the design metrics and to decide the right design early in the implementation stage itself. This difficulty influences the time to market pressure. To achieve successful implementation, right decision has to be taken during the selection of architecture and software components. Hence there is need of system design approach to predict system failures, evaluate and optimize the design metric, and select the right design before implementation to avoid re-spins.

Design matrices are the nonfunctional requirements of embedded systems such as performance, power dissipation, cost and other design metrics. Performance matrices are latency utilization and throughput. Response time is the time interval between request and response of the system. Throughput is the amount of inputs processed per unit of time. The amount of work handled by
the resource is called utilization. Latency of the system depends on task processing, task waiting, memory access, and IO operations.

**Figure 1.4: Embedded Systems Design Challenges**

Size of the system is measured in terms of physical space or memory required, RAM in KB and internal flash memory requirements in MB or GB for running the software and for data storage and the number of million logic gates in the hardware. System cost includes one-time non-recurring cost (cost of designing, developing, debugging and testing the hardware and software) and cost of manufacturing each unit. System cost can be optimized by conducting trade studies. The battery needs to be recharged less frequently if power dissipation is small. Power consumption during the operational and idle state of the system should be optimal. System power can be optimized by reducing Clock Rate and Voltage, Wait Stop and Cache instruction disable technique [48]. Power consumption is an important feature for battery operated systems.

Evolution of embedded systems as discussed in section 1.1 demands exponentially more features and functions to be adapted by these systems. This creates the expectation of larger processing and storage capability, which leads to more complexity [38]. Addressing this complexity is another design challenge. There is an absence of design approach and automation that could enable the system designer to successfully handle system complexity.
Hardware design, verification, and software-development engineers are not able to identify where the fault is in the system and they blame each other because of the narrow focus on system design. Communication is required between the engineers to have the common understanding of the right system design and requirements. The competitive industrial world demands these complex systems to be developed as fast as possible along with tight attainment of nonfunctional requirements. In the design process, wrong decisions at early design stage results in system failures.

To resolve system failures, needs the repetition of design process. Number of re-spins can delay time-to-market, leading to severe reputational and business losses. Considering an instance of time required to release a product to market is 52 weeks, if it is delayed by four weeks resulting in income loss of 22% and if it is delayed by 10 weeks that causes the loss of 50%. [27,28]. Hence there is a need of an efficient design approach to address system complexity, achieve faster time to market, predict the system failures, and optimize design metrics before implementation.

1.5. System Level Modeling

To address these design challenges, the conventional design approach changed new approach by including system-level modeling and evaluation steps as shown in Figure 1.5. System-Level Modeling is the process of creating executable system specification by pre built analysis specific modeling elements to study the credibility, bottlenecks, or failures of the system. Abstract modeling elements are the generalized pre-built modeling blocks used for analysis specific modeling. The representations at the higher level of abstraction are generic, in the sense that they can deal with different simulative modeling components, and performance numbers are created by simulating these models. Analysis particular information is required to build these blocks.

Hence abstract modeling elements mimic the actual system components precise to analysis, and they are simple enough to understand. Using this model, probability of a successful implementation can be realized early in the design stage in a cost-effective manner. Early prediction of system bottlenecks, optimization of design metrics can be done using this approach to reduce the number of re-spins.
System-level modeling and analysis provide greater flexibility in selecting alternative components so that system designer can conduct trade studies to validate the right design and eliminate the risk of not meeting the requirements. This approach is an inexpensive and efficient way to predict bottlenecks, optimize design space, and get the right specification in the early design stage. The hierarchical approach in system-level modeling helps to visualize complete system and provides the solution for system complexity. Validated system model can be shared between system development teams so that each one can have a common understanding. Steps involved in this approach are data collection, defining modeling objectives, selecting or building abstract components and templates, building the complete system model for particular analysis, and conducting the experiments using simulation to evaluate and optimize design metrics to get viable design specification for further implementation.

1.6. System Modeling at Different Levels Abstraction

System modeling can be cauterized into following types of analytical modeling by mathematical calculation, Instruction level, Cycle accurate level, RTL or logical level, and System-level modeling by simulation.
1.6.1  **System Level Analytical Modeling by Mathematical Calculation**

This approach comes at the highest level of abstraction. Models are represented in a formal manner. In analytical approach system level performance modeling is done by computing set of numerical equations to estimate the performance and power metrics. These models describe only approximate functional behavior, and timing. This modeling is faster in evaluation time because of over simplified models and can be evaluated just by solving the mathematical equations, but needs to be compromised in results and this approach can be applicable only for a system with single application. An individual detailed component modeling is not possible in analytical method. Resource contention modeling is not possible in this approach. The modeling efforts are small and cover a wide range of design space.

1.6.2  **Instruction Level Modeling**

In Instruction-Level Modeling, complete functional detail by the application is required. The application which is to be mapped onto the processor is translated to machine instructions either manually or by using a compiler. Performance numbers are determined by the delay of instruction execution and memory access. Instruction-level models provide more accurate results for the single core but take longer simulation time. This cannot handle the communication between the CPU and other parts. It cannot support for capturing resource contention. Hence instruction-level models support only for uni-processor system and difficult to represent the multiprocessor systems.

1.6.3  **Cycle-Accurate Level Modeling**

In Cycle-accurate modeling, micro-architecture details are required. In this simulation, instructions are executed sequentially. C/C++ and VHDL can be used for cycle accurate modeling. Simulation engine and mechanism for modeling of the concurrent process is used to capture the resource contention respectively [41, 56].

1.6.4  **RTL and Logic Level Models**

In RTL and Logic Level, Hardware Description Languages are used for modeling. Functional and timing details are required and can be realized in silicon. The information provided is much
more extensive than what is found in cycle-accurate models. For performance evaluation of systems, it suffices to have the clock cycle accurate detail of events. Therefore, one doesn’t need logic level models.

### 1.6.5 Analytical Modeling and Evaluation by Simulation

In this approach, simulation execution goes through a sequence of states, one at a time. It can be handled at any level of abstracted models and gives more accurate results. This can be used for modeling of complex systems and resource contention [54]. Simulation speed is based on modeling languages. Fast simulation can be modeled for multiprocessor and application-specific hardware units. By and large, a framework expected to learn a numerical representation flourished along the assist regarding simulation software program.

### 1.7 Levels of Abstraction and Trade Offs

Modeling effort, evaluation effort, accuracy, adaptability, and generality are the issues in system-level modeling and analysis.

#### 1.7.1 Evaluation Effort

Evaluation effort is the amount of effort and time required to conduct the experimental study by simulation. Evaluation efforts increase as an increase in modeling details. Analytical evaluation is the fastest one. Again, if the evaluation is simulation-based, then detailed model means the simulation kernel has to handle the larger number of events and time is incremented in small units. Simulation time is another important factor that influences the evaluation effort. Simulation is faster at the higher level and slower at the concrete level. An important issue with evaluation effort is how to achieve quicker simulation without compromising the accuracy. Modeling efforts, evaluation efforts and accuracy increase with the decrease in the level of abstraction whereas, the opportunity to change the models decreases with the decrease in the level of abstraction. Figure 1.6 shows the tradeoffs between accuracy, evaluation speed and modeling efforts at various levels of abstraction.
1.7.2 Accuracy

Accuracy is the closeness of performance numbers evaluated to the exact values of performance metrics.

![Diagram](image)

**Figure 1.6: Trade Offs at Various Levels of Abstraction**

Accuracy increases with an increase in modeling details. Modeling at concrete details, that is at RTL provides more accurate simulation results on other side it takes large modeling effort with less flexibility to alter system elements. Instruction-level simulators are not able to handle multiple processors because timing details are neglected to increase the simulation speed. In the cycle accurate simulation, much logic and instruction-level details are not considered to fasten the simulation runs. Cycle accurate simulation is also more time consuming because of detailed mapping of application onto the architecture that leads to addressing and partitioning problem.

1.7.3 Adaptability

Adaptability is the ability to modify the level of details of different models without changing the framework. This offers a means to generate performance numbers and conduct experiments at different levels of abstraction. Hence more adaptability minimizes the modeling efforts.

1.7.4 Generality

The generality of Modeling and Evaluation is a process of developing common modeling components, which can be used in various applications. A slight modification is made with the generic model to develop specific components. Performance evaluation methodology is
generally at the system level if it can take different types of system components in same exploration iteration, without making component specific changes. Using generic models, less effort is required to change the component architecture. Using generality and reusability, faster time to market can be achieved and provides the solution for system complexity. Hence there is a need for quicker modeling and evaluation that captures the system complexity. Mapping of application behavior on to the hardware architecture has to be done for every design point. This mapping includes the challenges of partitioning and synchronization and code synthesis [9]. At the higher level of abstraction, models attempt to capture only timing details. Functional of application and architecture is either completely ignored or captured in less detail. Modeling at lower levels of abstractions, timing, as well as functionality, is captured in larger detail. If a system is simulated at the logic level, running software over it is ruled out. Thus there is the needed for a fast performance evaluation methodology that captures architecture as well as application behavior to support speedy exploration. To support rapid exploration of different design choices, the methodology should support variations in architecture. The methodology should capture application behavior to the extent that exploration methodology can cover low-cost and low power design. In order to support multiprocessor systems, it should also be able to model performance degradation due to contention on interconnections.

1.7.5 Modeling Effort

The amount of efforts required to develop analysis specific models for hardware and software components is called modeling effort. Modeling effort is high at higher level of abstraction. Selective features of component are considered while modeling component for specific analysis. Each component is modeled before building the complete system. Learning and modeling each component requires more modeling effort. Modeling effort depends on the learning curve, defining modeling objectives, data collection, modeling component selection, and system modeling as given in equation 1.1.

\[ ME = f (lc, mo, dc, ms, sm) \]  
\[ \text{Equation (1.1)} \]

Where

- \(lc\) → learning curve
- \(mo\) → modeling objectives
Modeling effort can be optimized by adopting systematic approach with enhanced technique and increasing the number of templates and frame works of system models. Hence learning curve, modeling component selection and system modeling is directly proportional to the systematic approach, and number of frame works as given in equation 1.2.

\[ lc, ms, sm \propto (sa, fw) \]  
\[ Equation \ (1.2) \]

Where

- \( sa \rightarrow \) systematic approach
- \( fw \rightarrow \) number of frame works or pre built models

### 1.8. Ptolemy Simulator Based Modeling Environment

Ptolemy is the open source environment in which, researchers can build methodologies for embedded and hybrid systems. Ptolemy was established at U.C. Berkeley by an informal group of researchers. This provides heterogeneous simulation and modeling [49]. The different types simulation engine used depends on the need. Discrete event simulator is specifically used for embedded system modeling and simulation. In this simulation, model triggered by every event of time and result of the each trigger is recorded along the simulation time. Ptolemy provides the actor oriented design approach in which each component in the model considered as actor and they are communicated through the channels. Models are represents the behavior of the actual system.

The proposed approach demonstrates the modeling of input traffic to the system, applications and hardware configurations used for the system with minimal amount modeling cost. VisualSim modeling library with Ptolemy's simulator is used in this approach. Faster modeling and simulation can be achieved using this modeling environment. Discrete Event Simulator maintains displays the present time and computes timing arrivals cumulatively. In this simulation, the model is being triggered as an event. Statistics of performance metrics are calculated and displayed at each event [30].
1.9. Challenges of System Level Modeling and Analysis using Simulation

The huge number of prebuilt modeling components and faster simulation provides benefits to be modular to build the system model without putting effort to develop the modeling components from scratch. Certain challenges still remain to be addressed in optimizing the modeling and evaluation effort because of the following regions.

A lot of learning is required to understand the modeling components and frameworks so that modular can select the right modeling components for his/her requirements and objectives. There are chances of learning irrelevant modeling components and frameworks due to lack of guidance and information. For conducting experiments and performing analysis, there is a need of repetitive executions [39]. Multiple simulations run for various values of parameters and collecting and plotting the results is the lengthy process. For example, to run the simulation for a parameter of its 100 different values, it takes 100 simulations runs, also collect the results for 100 different values, if there is any number of parameters, then again, it will become the more lengthy process. In such cases, batch simulation and post-processing are required for analysis. Following steps are required to be followed for batch simulation and collecting results. Hence there is the need for proper direction or systematic approach to developing the system model and batch simulation setup for faster simulation and evaluation.

Knowledge of synthesizing the new idea or concepts is the required ingredients to develop the system model. Definition of the system in real word is the set of components connected and working together according to set of rules and is having inputs and outputs. In general, system components are input/output devices, processors, bus, memory, system, and application software. But system model represents the flow of tasks, how the system responds to events of requests. Each task represents set of instructions processing on the specified resource. System model consists of behavior (task flow), the source of input traffic, and outputs to measure the response time. Hence there is the need of the process to translate system specification, modeling objectives to the instruction to build the system model.
1.10. **Problem Statement and Objectives**

Proposed research work addresses the challenges of long learning curve, larger modeling and evaluation effort in simulation based system level performance modeling of embedded systems. This is achieved by demonstrating the systematic system-level modeling approach, exploring the synthesis process from idea, requirements and modeling objectives to the basic instructions to build the system model and generating the automatic batch file for multiple simulations to make faster and easy for evaluation process. Research objectives are as follows.

- **To demonstrate the systematic approach for Simulation based System Level Performance Modeling.**

Proposed simulation method demonstrates the steps involved in this approach by considering case studies. Case studies shows representation, prediction, estimation and analysis at system-level that involves the modeling the behavior of multitask software, hardware components and allocation of hardware units to tasks is modeled using system modeling and estimation tool. This approach includes the pre and post-system modeling, component selection, system modeling, simulation, and analysis steps. Specifying problem definition and modeling objectives, collecting the real system data, and documenting analysis results is done in the pre and post system modeling step. In system modeling step, complete system is built using system-level modeling environment. Order of the stages in this modeling flow may not be the same, it can be vary based on availability of modeling information and required accuracy.

- **To explore the process of translating modeling guidelines from idea, requirements and modeling objectives to build the system model.**

The effort in learning and selecting the simulation models can be reduced by extracting the modeling instructions from the given necessary modeling information. The necessary modeling details are acquired in questionnaires form. These questionnaires are related to system modeling and analysis goals. Extracted instructions are synthesized information of relevant modeling blocks, templates, and steps required to build complete system in Ptolemy based system modeling environment.
To develop the technique to generate batch simulation setup for fast simulation of system level models of embedded systems.

Multiple simulation runs are needed for analysis and optimization of non-functional metrics by varying system parameters and input variables. Batch simulation can be used to achieve multiple simulation runs with less manual setup. Algorithm is developed to generate batch file for different values of input variable and parameters.

1.11. Thesis Organization

In this chapter 1 discussed about significance of embedded systems, system design challenges, and proposed solution. The remaining chapters are organized as follows.

Chapter 2 highlights literature review of related work. Discussion made in this chapter is about existing method for system level modeling. Survey explored the need of proper teaching method for modeling skills, generalized modeling frameworks, and techniques to enhance the system level modeling approach.

Proposed methodology is discussed in Chapter 3. Proposed method includes five steps: acquisition of system information; mechanism of translating the acquired system details to the modeling guidelines, system modeling using Ptolemy's simulator based modeling environment, simulation, and analysis.

Chapter 4 explains about system-level modeling concepts, types of system level modeling, issues in system level modeling, Concept of simulation based system level modeling, developing simulation model at different level of abstraction, what is simulation, performance analysis and briefed about Ptolemy Simulator Based Modeling Environment.

Chapter 5 demonstrates the proposed approach by considering case studies. Discussed here is about system level performance modeling of high computational algorithm using Ptolemy simulator based modeling environment, systematic approach for simulation based system level modeling and enhancement of system level modeling and simulation using modeling instruction generation and batch simulation.
Experimental results are presented in *chapter 6*. Experimental simulation results are performance metrics latency utilization. Modeling effort for three case studies are compared and discussed.

Concluding remarks and future scope are discussed in *Chapter 7*. Integrating the enhanced approach for modeling and batch simulation and systematic approach enhance the system level modeling. Highlighted the future scope of the proposed work that are required to increase the number of modeling frameworks, and make menu based modeling approach by up-gradation of synthesis frameworks through online in huge and faster manner.