Master’s Thesis

Using Iterative Simulation for Timing Analysis of Complex Real-Time Systems

Mohsin Qasim
MSc Computer Science with Software Engineering Profile
Department of Computer Science and Electronics (IDE)
Mälardalen University, Sweden
mqm05001@student.mdh.se
ch.mohsinqasim@gmail.com

Supervisor
Johan Kraft
Department of Computer Science and Electronics (IDE)
Mälardalen University
Sweden

MÄLARDALEN UNIVERSITY
ABSTRACT

The size and complexity of software in embedded systems is increasing with every new product launched in the market. When changing such complex systems i.e. adding a new feature, it is hard to predict the consequences on the behavior. This is especially the case for temporal properties of the system e.g. response time, which depends upon factors which one cannot figure out during the implementation, such as execution time. This is a big problem when developing industrial and embedded systems, which often are **real-time systems**. The main problem is that these real time systems cannot be analyzed by the analytical methods that already exist for response time analysis like rate monotonic or formal methods like model checking. The main focus of this thesis is to find a proper simulation based analysis method for real time systems where simulation is used in a iterative process we call this iterative simulation. The thesis will not only explain the main timing analysis methods in detail but also provide a comparison between these methods. Moreover, this thesis contains a literature study on the topic of iterative simulation. In the end the whole process of the iterative simulation will be explained in detail, including the important parameters of the method.
ACKNOWLEDGEMENT

This master thesis was carried out at Mälardalen University (MDH), Department of Computer Sciences and Electronics (IDE). First of all I will like to thanks GOD, Who is the most merciful and most gracious and whose blessings are copious. I would sincerely like to thanks my supervisor Johan Kraft from IDE not only for his support, guidance and encouragement but also giving his worth time and discussing the idea of this thesis. I am really thankful to IDE as it provides an excellent thesis place and research environment.

In the end I will like to thank my parents to whom I owe all I am and friends specially the people in Burhovda and Junior 4th floor for there moral support through out my study period in MDH.
# Table of Contents

1 INTRODUCTION .................................................................................................................. 5

2 METHODS FOR TIMING ANALYSIS .................................................................................. 7
   2.1 ANALYTICAL METHODS ............................................................................................... 7
   2.2 MODEL CHECKING ......................................................................................................... 9
      2.2.1 Model checking Real Time systems ..................................................................... 10
   2.3 SIMULATION ................................................................................................................ 11
      2.3.1 Discrete event Simulation .................................................................................... 11
      2.3.2 Modelling languages for discrete event simulation ........................................... 12
      2.3.3 Discrete event simulation of real-time systems .................................................. 12
   2.4 CONCLUSION ............................................................................................................... 13

3 ITERATIVE SIMULATION - A METHOD FOR TIMING ANALYSIS ................................... 15
   3.1 THE CONCEPT ............................................................................................................. 15
   3.2 RELATED WORK .......................................................................................................... 16
   3.3 GENETIC ALGORITHMS .......................................................................................... 18
      3.3.1 Overview ............................................................................................................... 18
      3.3.2 The GA procedure ............................................................................................... 19

4 ITERATIVE SIMULATION – A THEORETICAL FRAMEWORK ........................................... 21
   4.1 WHAT IS PSEUDORANDOM NUMBER GENERATION? ............................................... 21
   4.2 THE ITERATIVE SIMULATION PROCESS ................................................................ 21

5 CONCLUSION AND FUTURE WORKS ............................................................................. 26

6 REFERENCES ..................................................................................................................... 27
1 INTRODUCTION

Sectors like telecommunication, industrial automation systems and industrial robotics are developing increasingly complex software systems. These systems contain millions of lines of code. When changing such complex systems, it is hard to predict the consequences. Will the system still meet its requirements? For each feature that is added the complexity increases, which makes the system increasingly harder to maintain and the productivity therefore decrease. To stay away from this hazard we must develop the system more efficiently with proper testing and inspection of code, but this is not enough as we might miss some bugs during testing as bugs related to timing are hard to find. This is a big problem when developing industrial and embedded systems, which often are real-time systems, where correct operation is not only depending on the output, but also when the result is delivered. Important terms of real time terminology are:

- **Tasks** - A process in a real time system usually with some deadline and a period.
- **Response time** - The time in which system gives output after taking input.
- **Worst case response time** - The maximum possible response time of a task. This occurs when all tasks in the system wants to execute at the same time, and with the individual worst-case execution time.
- **Worst case execution time (WCET)** – The longest possible execution time of the task.
- **Deadline** - Is the time required for system to finish or submit something? Deadlines can be classified into two groups, hard deadlines and soft deadlines. Hard deadline is that deadline which must be met. Soft deadline does not need to meet at all times. The results can be used although, they come late. Deadlines can be specified for individual tasks (task deadline) or for chains of related tasks (end-to-end deadlines).
- **Real time scheduling** - An activity that determines the order in which concurrent tasks are executed on a processor. [2] There are many types of scheduling policies for real time systems.
- **Soft real-time systems** - Systems containing only tasks with soft deadlines
- **Hard real-time systems** - Systems containing tasks with hard deadlines
- **Scheduability analysis** – An analysis that determines if a set of tasks are schedulable, i.e. possible to run without violating their deadlines.

Take a simple example which will explain scheduling, if we are given 5 jobs to be finish and each job will take one day. All the jobs will be finish after 5 days but what if one of them has a deadline, e.g. the 4th task was supposed to be done on 3rd day so doing each task in order one receives them there will be a deadline miss. Since the response time of a task depends on the execution times of all tasks with higher priority, any change that increases the execution time of a task can make other tasks violate their deadline. This might require special conditions which are easy to miss during testing. In that case, the error instead occurs after the product has been delivered and it may cause major problem to the customers.

Analytical methods for scheduability analysis are very pessimistic for complex systems. In fact, for some systems analytical scheduability analysis is infeasible [14] due to the high pessimism, since an analysis would always give a negative result - even though the real system works perfectly. For systems that can not be analyzed, there are two methods for introducing analyzability, intrusive methods and non-intrusive. The first way, in which the system is redesigned to fit an existing analysis method, e.g. rate monotonic analysis, is very costly since it requires a lot of work and there is a high risk of
causing new bugs. A non-intrusive method is when we do not change the system but instead construct a model of the existing system that can be analyzed, using e.g. simulation. Simulation gives us a way in which we can make analyzes of a model of the system in order tell us how the system will work after the changes have been made, but we must keep this in mind the results are only the predictions.

By using the vast knowledge of simulation and real time systems, the goal of this thesis is to develop a simulation technique, better and more accurate than the “random” simulation currently used for simulation based analysis of real-time systems. The simulation technique proposed in this thesis is an iterative process. The purpose of using this technique is not only to do the simulation randomly but also search for the worst case scenario of a system. The idea is randomly, do many simulations, find the most interesting ones (e.g. the 5% with highest response time for some task), and then use these as starting points of new simulations in order to find a better approximation of the worst-case scenario. This is an iterative process where each iteration finds even more extreme scenarios, until the termination point is reached.
2 METHODS FOR TIMING ANALYSIS

There are many ways by which we can do timing analysis and we will be describing three methods here, how they work, their limitations and strengths.

2.1 Analytical methods

Many methods have been proposed for scheduability analysis. A common method is Rate Monotonic Analysis. This method requires that the tasks have their priority assigned according to the Rate Monotonic (RM) policy in which tasks are assigned fixed priorities ordered as the rate. The tasks with shortest period are assigned the highest priority, the longer period, the lower priority. Priorities are assigned before execution and it do not change over time. Rate monotonic has assumptions that must be fulfilled

- Tasks must be periodic
- Deadline is equal to period
- Tasks must be independent

The steps involved in rate monotonic scheduability analysis are

- Assign the priorities according to RM
- Calculate the worst case CPU utilization
- Check if the task is schedulable by rate monotonic by satisfying the condition
  \[ U \leq n(2^{\ln n} - 1) \]

Where \( U \) is the utilization upper bound and it depends upon the number of tasks and for large number of tasks the value is 69.3 %.

Utilization based testing are not exact, not applicable to more general process models, and utilization gives answer only in yes or no. It does not give any indication of actual response time of the tasks.

An important result presented by Joseph and Pandya in 1986 [3] overcomes this drawback, RTA (Response time analysis) which comes from the fixed priority scheduling. It predicts if all tasks have met the deadlines. RMA calculates the worst case execution time of each task.
Example of Scheduability analysis

<table>
<thead>
<tr>
<th>TASK</th>
<th>TIME PERIOD</th>
<th>EXECUTION TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>TASK A</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>TASK B</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>TASK C</td>
<td>10</td>
<td>3</td>
</tr>
</tbody>
</table>

Task has lowest time period has the highest priority.

<table>
<thead>
<tr>
<th>TASKS</th>
<th>PRIORITIES</th>
</tr>
</thead>
<tbody>
<tr>
<td>TASK A</td>
<td>HIGH</td>
</tr>
<tr>
<td>TASK B</td>
<td>MEDIUM</td>
</tr>
<tr>
<td>TASK C</td>
<td>LOW</td>
</tr>
</tbody>
</table>

First we have to calculate the utilization:

\[ U = \frac{2}{5} + \frac{1}{8} + \frac{3}{10} \]

\[ \Rightarrow U = 0.825 \]

FORMULA:

\[ U \leq n \left( 2^{\frac{1}{n}} - 1 \right) \]

Where \( n = 3 \)

\[ 0.825 \leq 3 \left( 2^{\frac{1}{3}} - 1 \right) \]

\[ \Rightarrow 0.825 < 0.78 \quad \text{false} \]

Therefore we cannot say anything about weather they are schedulable or not.

So we need to run Response Time Analysis.

Response Time Analysis:

\[
R_t = C_t + \sum_{j \in hp(i)} \left[ \frac{R_i}{T_j} \right] C_j
\]

Here hp stands for high priority tasks i.e. the tasks with higher priority then i. C is the worst case execution time (Wcet) of the task, R is the response time of the task which is refined for each iteration of the formula until the value doesn’t change anymore and T is the period of the task.
Using Iterative Simulation for Timing Analysis of Complex Real-Time Systems

Wcet of Task A = 2
Response Time of (Task B)
RB1= 1 + (1/5)*2 => 3
RB2=1+ (3/5)*2 => 3
Wcet of Task B = 3
Response Time of (Task C)
RC1=3+ (((3/5)*2) + ((3/8)*1)) => 3+2+1 => 6
RC2=3+ (((6/5)*2) + ((6/8)*1)) => 3+4+1=>8
RC3=3+ (((8/5)*2) + (8/8)*1)) => 3+4+1=8
Wcet of Task C = 8
All the 3 tasks are schedulable.

2.2 Model checking

Model checking is a technique used for verifying requirements on a design and can be used to analyze real-time systems.
The idea behind mode checking can be illustrated by the Figure1 from [4]

As Figure 1 shows a model checking tool which accept system requirements or design (called models) and property (called specification) which the final system is expected to satisfy. The tool then outputs yes if the given model satisfies given specification and generate a counter example otherwise. The counter example tells us why the model doesn’t satisfy the specification. By looking at the details of the counterexample we can know the errors in the model correct the model and try again. [4]
The question come which formal language to use well the answer simply is that as requirements are different for different systems so it’s hard to have one formal language. Most of the model checking tools has their own rigorous formal language for defining models.
SPIN is a model checking tool. It has been evolved for more then 15 years. Promela (Process Meta Language) is the modeling language for SPIN its syntax is quite similar to programming language C. It is an open source so is available for all the platforms. SPIN can perform verification of very large state-space models. [5]
Model checking has proven to be a successful technology for verifying requirements and design of many systems, especially in real time embedded systems. For example, the SPIN model-checker was used to verify
the multi-threaded plan execution module in NASA’s DEEP SPACE 1 mission and discovered five previously unknown concurrency errors. [4]

2.2.1 Model checking Real Time systems

In Real time systems we have time requirements but the model checkers like SPIN do not have a notation (or a concept) of time. For real time systems there are specialized model checking tools like UppAal [10] developed by Uppsala University, Sweden and Aalborg University, Denmark. Another well known tool is KRONOS [11] developed at Verimag in France. These tools analyze models described in timed automata, i.e. finite state machines with real-valued clocks. The clocks can have different rates.

![Figure 2](image)

Above figure has two variables x and y, the system can remain in S1 as long as clock y hasn’t reached the value “a”, after that it is forced to go to S2. It can also non-deterministically decide at anytime to go to the state S2, or after b time units have passed according the edge time constraint x>=b (enabling condition), it can also perform action m and reset clock x. [6]

**UppAal**

The UppAal tool consists of a model-checking engine and a graphical user interface. UppAal can verify safety, bounded liveness and reachability properties on real time systems. Safety properties are some properties that are required to always hold, i.e. they are invariants. Bounded liveness properties are properties that have to hold within a certain time bound. A reachable property is a property that can eventually hold. [6]

UppAal has been used in many industrial case studies; For example [7] where the Swedish company Mecel AB10 uses UppAal for design and analysis of a prototype gearbox for vehicles.

**KRONOS**

It is also a well-known tool for model checking and is also based on timed automata but it uses a more powerful query language, Timed Computation tree Logic (TCTL). TCTL is an extension of the temporal logic CTL. It is available for several platforms and it’s distributed for free.

Kronos has been used for many industrial applications such as Philips uses audio transmission protocol where errors have been found to the previously hand made proofs. [16]
2.3 Simulation

Simulation is another method which can be used for analysis of response time. When using simulation, a detailed model of the system is executed in a simulator environment. Simulation can be used to evaluate the performance of the system when it has to be altered, which helps to reduce the risks of failure.

There has been changes in simulation software's with the growth of computer system. Simulation is used for many purposes now days especially in fields like medical, computer sciences, education.

The two important types of simulations are Continuous and discrete event simulations. An example of a continuous simulation is the simulation of water flowing through pipes and tanks, the state is always changing according to mathematical formulas. Imagine instead of water flowing continuously through the pipe, ice cubes are coming through the pipe. The arrival of an ice cube is then a discrete event [9]. Computer systems, including real-time systems, are discrete systems; we therefore focus on discrete event simulation.

2.3.1 Discrete event Simulation

In Discrete event simulations, the change in system is represented by a separate event. All these changes occur instantaneously in the system state, and are described as discrete events. There are many areas where discrete event simulation is being used these days. One of them is area of planning of new manufacturing. It's also used for the system that already exists and requires minor changes. For example if a company wishes to build a new production line, then the line can be first simulated to assess feasibility and efficiency [12]. The key stages involved in discrete event simulation are described below in Figure 3 [12]

![Figure 3](image-url)

*Three Key stages used in Discrete Event Simulation*

The ability to model random events and predict the effects of complex interaction between these events is the major strength of discrete event simulation. Many experiments are done to get many answers by changing the input of a model and then the result is studied by comparing outcomes. This kind of simulation is a decision support tool. [12, 13]

The simulation can be operated in any one of these two manners

- Time Slicing – it does not keep in consideration about the event that needs to be carried out and the model is advanced by a fix amount of time.
• Next event – it is vice versa of time slicing it moves the model to next event, without keeping in consideration of time. It is considered more efficient then time slicing.

2.3.2 Modelling languages for discrete event simulation

Most of the modelling languages have GUI and they are capable of getting the statistic for analysing the results. A simulation language helps in understanding the working of simulation on a computer.

Some of the discrete event simulations are explained below. [13]

• GPSS (General purpose simulation system)-it was famous in 1960s and 1970s but not that much used nowadays.
• Siman-It has a good GUI and is used by Rockwell Company.
• Simula-it’s a discrete event programming language developed in 1960s by Ole-Johan Dahl and Kristen nygaard at Norwegian computing centre Oslo.
• SimPy- it is based on python and is an open source package. It is language independent and runs on all operating systems today.

2.3.3 Discrete event simulation of real-time systems

The simulation tools used for real time systems are explained below.

STRESS

It is used for the analyses of hard real time systems. It is a language based simulator and can be used to study the general behaviour of an application. Algorithms for resource sharing and task scheduling can be defined in STRESS. It is mostly used for testing resource sharing and scheduling algorithms. STRESS tool include a simulator, a presentation tool and examples of modelling language. It does not support modelling distributions of execution times or memory allocation. [14, 1]

DRTSS

It is a simulation framework that “allows its users to easily construct discrete-event simulators of complex, multi-paradigm, distributed real-time systems” [1]. We can build a simulator by picking it from different set of algorithms and protocols in DRTSS; we can also add new algorithms and protocols to the original set. DRTSS is a part of PERTS family which was developed by University if Illinois at Urbana-Champaign. DRTSS helps in getting more detailed analysis of a system [14, 1]

ART-ML

ART-ML (architecture and real-time behaviour modelling language) has been developed at Mälardalen University at Department of computer engineering, and it has been used in an industrial case study [14]. It was designed as a mean to describe complex real time systems and is an imperative language similar to ANSI C, and it is based on the concepts of tasks. ART-ML has support for probabilistic modeling through the “chance” statement. The simulation time is advanced explicitly through an “execute” statement, which “consumes” CPU time according to a given probability distribution. [14]

ARTISTST

It's an event driven simulation tool which gives us the accurate simulation of real time systems. It is used for simulating the complex systems.
As shown in the Figure 4 below ARTISST is able to simulate: An application executing on top of a real-time operating system (RTOS), interacting with an external environment by way of events. [15]

![ARTISST Framework Diagram](image)

**Figure 4: ARTISST Framework**

The ARTISST framework comes with a set of default modules, and a set of default schedulers and task models. It uses C or C++ as the general programming language which allows the simulation to be implemented very closely to the applications which are real, by allowing the code to be reused. The simulator of the ARTISST is fully customizable as it doesn’t depend upon any operating system due to its extensible object oriented architecture. The result of the simulation can be seen in the form of a Gantt chart or series of evaluation metrics. The main advantage of using the ARTISST is that it takes all the system overhead into account. [15]

### 2.4 Conclusion

In this section we will compare all the three analysis method we discussed above

<table>
<thead>
<tr>
<th>Properties</th>
<th>Response Time analysis</th>
<th>Model checking</th>
<th>Simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements.</td>
<td>It require some assumption to be fulfilled</td>
<td>Requires a detailed analyzable model of the system, which can take a lot of time to develop and maintain</td>
<td>Requires a detailed analyzable model of the system, which can take a lot of time to develop and maintain</td>
</tr>
<tr>
<td></td>
<td>- All tasks must be periodic</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Tasks are independent</td>
<td></td>
<td></td>
</tr>
<tr>
<td>State space explosion problem</td>
<td>No</td>
<td>yes</td>
<td>No</td>
</tr>
<tr>
<td>Accuracy</td>
<td>Safe but Pessimistic</td>
<td>Safe and Accurate</td>
<td>Accurate but not safe (Optimistic)</td>
</tr>
<tr>
<td>Learning threshold</td>
<td>Low</td>
<td>High</td>
<td>Medium</td>
</tr>
</tbody>
</table>

[15]  Figure 4: ARTISST Framework
In this chapter the three analysis methods have been discussed and explained how they work but still we have not found a way by which can do timing analysis of complex real time system as each of topic discuss has some pitfalls. As Response time analysis (RTA) can be very pessimistic, especially for complex systems as it never guarantee that the system is schedulable. Model checking is less pessimistic as it gives a detailed model as compared to RTA, but the problem is state space explosion it may take to long to verify a given property. We therefore turn to simulation as simulation allows us to do predictions of very detailed and complex models, without state space explosion problems. The downside with simulation is that the result is often optimistic, i.e. the result might not be the actual “worst case”.
3 ITERATIVE SIMULATION- A METHOD FOR TIMING ANALYSIS

The iterative simulation is less optimistic as compared to ordinary simulation. We can say that iterative simulation is in between RTA and Model checking. If we give the proper parameters it might not have the problem of state space explosion and the predictions which we will get from the simulation can be less pessimistic.

3.1 The Concept

Simulation based analysis of complex real-time systems gives a good estimation of the system’s behaviour, but since simulation does the search by random and does not cover the whole state space it’s not possible to know if the worst case has been found or not.

In this chapter we will be investigating a better simulation based analysis technique and it is an iterative process, where random simulation is extended with a search technique for finding simulations parameters leading to extreme simulation result. The basic steps that it will involve during the simulation are.

- Run a large set of (random) simulations
- Select the most interesting results, e.g. the 5% with highest response time for some task
- In each selected simulation, use the state of the simulation at some appropriate points in time before the observed extreme behaviour (i.e. a task with very high response time), as starting states of new random simulation.
- Run new simulations of the model, from the selected starting states. Repeat this process until an extreme case is found that is not surpassed in the next iteration of the process.

The main weakness of this analysis method can be overconfidence in it since it doesn’t search the whole state space of the model. It’s doing the search randomly and focusing on some selected simulations and it might miss some of the worst ones. We might only find the “local” maximum while missing the “global” maximum.
### 3.2 Related Work

A natural part of developing a new concept like iterative simulation is to search for related work. We will search for the related work on internet in general and search different database by using different keywords. Below is the table which will show the keywords we used and the results we got from different databases.

<table>
<thead>
<tr>
<th>Keywords</th>
<th>Google</th>
<th>ACM digital library</th>
<th>IEEE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterative simulation techniques</td>
<td>[A]</td>
<td>No relevant result found</td>
<td>No relevant result found</td>
</tr>
<tr>
<td>Iterative simulation techniques for real time systems</td>
<td>No relevant result found</td>
<td>No relevant result found</td>
<td>No relevant result found</td>
</tr>
<tr>
<td>Simulation modelling</td>
<td>No relevant result found</td>
<td>No relevant result found</td>
<td>No relevant result found</td>
</tr>
<tr>
<td>Simulation modelling of real time systems</td>
<td>No relevant work</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multi pass simulation</td>
<td>No relevant work</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Genetic algorithms and real-time</td>
<td>No relevant work found</td>
<td></td>
<td></td>
</tr>
<tr>
<td>simulation response time analysis</td>
<td>No relevant work</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Iterative simulation based scheduling</td>
<td>No relevant work found</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simulation and genetic algorithms</td>
<td>No relevant work</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In [A] the search result we found a paper that is not related to our concept but they have used Markov chain Monte Carlo (McMc) rule and done the statistic to simulate sampling distribution of statistics “Model –based analysis to improve the performance of Iterative simulation”, all the other search showed up were not at all relevant to our search work.

We changed the keyword for search this time we tried ”Iterative simulation techniques for real time systems” but no relevant papers was found.

In [B] “Simulation system for real-time planning, scheduling, and control”[19] in which the frame work for online simulation system is described and it also explains the operational planning, scheduling and control of manufacture systems.
In [C] "Future of simulation: Real-time control: the extended use of simulation in evaluating synchronized control systems of AGVs and automated material handling systems" in this article the author explains briefly what important role simulation can play in complex logistic systems and their control systems before commissioning. The author uses a three step approach in evaluating both logistic and logistic control systems.

In [D] “A real world objects modelling method for creating simulation environment of real-time systems” [20]. The authors discuss about the two main problems of simulation of a target software system and its environment, and then they develop a method which supports incremental specification and simulation of both the target software and its environmental objects. They have also explained a method and a tool that visualizes the simulated real world object in three dimensional graphical displays.

In [E] Hamad I. Odhabi and Ray J. Paul have given an interesting knowledge about the “The four phase method for modelling complex systems”[21] in this article they have investigated an attempt combine different simulation tools in order to build a simulation environment which can be used to model complex system behaviour. They have given a new method named as (FPM) and given its importance in the context of iconic representation and the automatic generation of model code.

In [F] Sreeram Ramakrishnan and Mayur Thakur in there paper “AN SDS MODELLING APPROACH FOR SIMULATION-BASED CONTROL” in there paper they told that simulation based controllers make decisions by looking ahead in future i.e. if the system is in state So at time t=0 the number of possible states the system can be in time t=n is in general exponential in n. they have presented an work which formalize and will facilitate efficient simulation based controls. [26]

In [G] Gokhan Metan and Ihsan Sabuncuoglu in there article “A SIMULATION BASED LEARNING MECHANISM FOR SCHEDULING SYSTEMS WITH CONTINUOUS CONTROL AND UPDATE STRUCTURE” purpose a learning simulation bases mechanism. In which the systems construct a tree in the manufacturing environment automatically and selects the dispatching rule for each scheduling period. The tree is also updated automatically and the system works well over time. [27]

In most of the key words used there was not any relevant work found the material found was about the certain topics , that how they work and relevant with other technologies related, the most closet found has been listed above.

Smith and Jeffery has done a very good job and they wrote a survey paper name as “Survey on the use of simulation for manufacturing system design and operation” in this paper they have classified all the literature about discrete event simulation and its uses for manufacturing system design and operation problems. It covers all gives the name of all the papers from 1969 to 2002. The authors then classify the simulation in three primary classes for this paper that are manufacturing system design, manufacturing system operation, and simulation language for development of applications. The important part of the survey paper is where the authors gave the detail of real time simulation and who have worked on it. They start with Harmonosky in 1995 as he provided a review of real-time scheduling research. Harmonosky and Robohn in 1995 they provide about how much time is needed to run the simulation on physical manufacturing systems. Lindau, Kanfio, and Lumsden in 1994 they used simulation to show how valuable real time systems can be for decision making in a car-body manufacturing system. Drake and Smith in 1996 describe the use of a discrete event simulation for controlling a flexible manufacturing system. Authors have given many names of the different researchers and there areas of study in simulation relevant to simulation in this research paper. [23]
3.3 Genetic Algorithms

After going through the entire search Genetic Algorithm (GA) seems to be an area closely related to iterative simulation. In this section Genetic algorithm will be explained in details and compared with Iterative Simulation.

3.3.1 Overview

Genetic Algorithms (GAs) are the search algorithms based on idea of selection and genetics. It follows the rule given by Charles Darwin of survival of the fittest. It does a random search within a defined space for solving a problem. The idea of Genetic algorithm was first given by John Holland in 1960s. Since then a lot of studies and experiments have been done on GA and it's been also applied in many fields. In general GA is used for calculating the behaviour of the adaptive systems i.e. automatic programming, machine and robot learning. GA follows the principle of evolution via natural selection in which a certain population of individuals goes under selection. The technique is inspired by evolutionary biology such as mutation and recombination [18, 17]. Genetic algorithms can be implemented as computer simulation where one can encode the given problem of chromosomes, which is basically a blueprint of something. In GA a chromosome is some value that is part of a solution to the problem and then we compare the fitness of each individual, which is a possible solution to the problem, a combination of chromosomes. The GA's are used mainly due to its simple search techniques it uses and giving the good solution of some difficult and complex problems. The GA is mostly used when we have large, complex and poorly understood search space, and other search techniques have failed to give a possible solution.

Genetic algorithm starts with a population of completely random individuals and happens in generations. The fitness of the whole population is evaluated and individuals are selected depending upon their fitness form a new population i.e. the next generation. The new population is then used in next iteration of the algorithm.

The general steps of algorithm are [18]

- Randomly generate the first generation $S(0)$ from solution space i.e. the population of all possible solutions.
- Calculate the fitness $F(s)$ for each individual in the current generation $S(t)$
- Apply a selection technique $t(s)$ on each individual $m$ in $S(t)$ where $t(s)$ is proportional to $F(s)$
- Generate $S(t+1)$ by selecting individuals from $S(t)$ to produce offspring via some genetic operator where the genetic operator via a method for recombination defined for the specific problem.
- Repeat second step until the best solution is obtained.

This procedure of Genetic algorithm is explained in more detail in the next section.
3.3.2 The GA procedure

In general GA requires two important things to start the procedure [17]

- Genetic representation of the whole solution space
- Fitness function to be defined - It measures the quality of the represented solution, and it is always problem dependent. Where it is hard to define fitness expression, interactive genetic algorithms can be used.

When we have defined the genetic representation and fitness function, the steps GA will go through are as follows.

- Initialization - In the beginning random solution are picked to form a certain population, the size of the population and selection depends upon the problem dealing with and it contains many solutions.
- Selection – a portion of the population is selected for breeding a new generation, and the selection is made on the bases of fitness. There are different selection methods to select the fittest one depending upon the problem in hand, these selection methods are different functions i.e. like fitness function discussed earlier.
- Reproduction – In this step the second generation of population is generated by selecting through a method called recombination. First a parent solution is selected from the pool selected before and a child solution is produced by using the method of recombination and it has the characteristics of the parents. New parents are selected for each child until a small population is generated. This helps to increase the fitness of the population
- Termination – The process continues until termination condition is reached and there are many terminating conditions
  - A solution is found
  - Maximum number of generations reached
  - Fixed budget reached

Pseudo-code algorithm - The pseudo code algorithm as described in [17]

```plaintext
Choose initial population
Repeat
Evaluate the individual fitness's of a certain proportion of the population
Select pairs of best-ranking individuals to reproduce
Breed new generation through recombination
Until terminating condition
```
Relationship between Iterative simulation and Genetic algorithm (GA)

As we have given some details about Genetic algorithm and iterative simulation so now we will discuss about some of the similarities and differences between them.

Similarities and differences

<table>
<thead>
<tr>
<th>Genetic algorithm</th>
<th>Iterative simulation</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Starts the search randomly</td>
<td>Starts the search randomly</td>
<td>Its same for both processes</td>
</tr>
<tr>
<td>Selects a portion of population.</td>
<td>Selects a portion of population.</td>
<td>Both uses the same way of selecting a certain population to generate the next population</td>
</tr>
<tr>
<td>While generating the next generation, GA applies recombination on individuals to generate new individuals.</td>
<td>Iterative simulations don’t use recombination method, but next generation is generated through mutations of the best fitting simulation case in the last generation.</td>
<td>GA uses recombination. Iterative simulation doesn’t use recombination. In iterative simulation, each individual in the new generation is based on exactly one individual from the previous iteration. This is the main difference.</td>
</tr>
<tr>
<td>It will terminate when termination condition is reached which depends upon the problem.</td>
<td>It terminates when we get do not any better value then the last iteration.</td>
<td>These steps are very similar. The termination condition of the iterative simulation could be used in GA as well, Iterative simulation is a special case of GA in this case.</td>
</tr>
</tbody>
</table>
4 ITERATIVE SIMULATION – A THEORETICAL FRAMEWORK

Before going into the details of iterative simulation, we will explain pseudo random number generation. The problems which we might face when using the iterative simulation will also be explained at the end of this chapter.

4.1 What is pseudorandom number generation?

We can say that random number is a number that is generated from a process which does not give us predictable outcome [24]. John Von Neumann in 1951 said

“Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin.”[24]

Generating “true” random numbers is not possible in software, unless there is specific hardware support. Software can however generate pseudorandom numbers. The pseudo random number generator produces long sequence of random result, which is determined by a short initial seed.

The main characteristics that a good pseudo random number generator must have are:

- No repetition: - if one sequence is generated it must not repeat itself.
- Good numeric distribution: - if we are producing a random number between 0-10 then the number of each digit must be distributed equally.
- Lack of predictability: - the only way to predict the next number will be by knowing the seed and the formula by which we are generating the sequence.

The seed value is the important value as it determines the sequence of pseudo random numbers. If you initialize a pseudo number generator with the same seed, you get the same pseudo random numbers. The information that used by seed does not need to be truly random. [25]

4.2 The Iterative Simulation Process

A simulation based analysis technique called as iterative process is used, where random simulation is extended with a search technique for finding simulations parameters leading to extreme simulation result i.e. a very high response time for a certain task. As presented in the previous chapter the main steps in the iterative simulation process are

- Run a large set of (random) simulations
- Select the most interesting results, e.g. the 5% with highest response time
- In each selected simulation, use the state of the simulation at some appropriate points in time before the observed extreme behaviour (i.e. a task with very high response time), as starting states of new random simulation.
- Run new simulations of the model, from the selected starting states. Repeat this process until an extreme case is found that is not surpassed in the next iteration of the process.
To perform all the steps described above the first thing to do is to define all of our parameters. The best value of these parameters is unknown as no one as far as searched has tried this analysis method before. Each simulation case represents the point where random simulation will be performed. The identified parameters are:

- The population size (N) the main thing that has to be defined is that which method to use a constant population size or increasing population size produced after each execution.
- The run length of the simulation (L)
- Fraction of results to investigate in the next iteration (F)
- Starting time of the simulation ($T_s$)

The state of a simulation at a certain point in time, $T$, is determined by the sequence of random numbers used. This sequence can be summarized by the seed used. In order to set the starting state for the simulator from the previous simulation the simulator need to know the value of starting time ($T_s$) and Seed ($S$); A set of seeds and $T_s$ values needs to be set, i.e. a set of pairs ($S_0$, $T_0$), ($S_1$, $T_1$),...... ($S_n$, $T_n$) since we might want to change seeds several times.

The simulation case can be shown in the Figure below,

![Figure 5: Simulation case](image)

The outputs will include the property value which is typically the highest response time for a certain task in the simulation case and $T_i$ which is the start time of the task instance that got that response time. For the first iteration the values for both the Seed and $T_s$ are zero, since there is no restart time and the seed should not be specified but instead picked from the system clock, to get a new sequence of random numbers. As $T_i$ is the time at which the task instance got the response time, it will give the starting point $T_s$ is a point in time before $T_i$ ($0 < T_s < T_i$), to use as starting state in the next iteration.
The initial model state can be shown by the Figure below which will have a specific population represented by N. As N numbers of traces are produced and each simulation case will have a property value and time value Ti.

**Figure 6: The first iteration of the process**

The simulation case with the highest response times will be used as input for next iteration, it will be done for getting the best result, as the figure below shows the how each trace is used for the second iteration.

**Figure 7**
The outputs will be the property value and $T_i$. The process of iterative simulations will keep on running on the all simulation cases that have the highest response time, so the question comes when to terminate the simulation. The one possibility can be that when there will be no increase in property value and the output gives the same trace property value as the previous one that means that this is our required value and terminate at this point.

There is another main question that needs to be solved is that should the population size $N$ kept constant every time we run iterative simulation on a set of trace or we can allow it to increase. It can be shown by two different diagrams. In the first figure as shown, If the population size is allowed to increase then there are chances that it will take too long to give the final result as the number of simulation for each iteration will increases very fast.

On the other hand if we use a constant population size, iteration will be much faster,
The problem which we can face is that we do fewer analyzes in the first iterations compared to the other method. But if we have a Constant population size, it can be much larger, since it doesn’t grow. If it grows, then initial population size has to be quite small.

There is a slight possibility that when the iteration is done the simulation case with highest response time will be missed, as defined number of simulation cases has already been selected. It can be shown by the figure below, where (o) represents the simulation cases we didn’t try and (*) represents the cases on which the simulation is done.

![Graph showing simulation cases](image)
5 CONCLUSION AND FUTURE WORKS

In this thesis we have done the literature study on the main methods for Timing Analysis and develop a new iterative process for simulation. In the first phase of the thesis the three timing analysis method had been briefly described and compared. It’s been seen that Response time analysis (RTA) is safe but can be very pessimistic and it does not face any state space explosion problem. Model checking is less pessimistic as the models may be more detailed and is therefore more accurate, but the problem it faces is state space explosion i.e. it can take too long to verify a property. Simulation allows us to analyze detailed models but the method is not safe, it is an optimistic estimation i.e. the result might not be the actual “worst case”.

In the second phase did the search for the related work that if some one has already worked on this iterative simulation method. It was found that Genetic Algorithm is quite similar to iterative simulation method, the only difference we have is GA uses recombination for selecting the next generation, but in iterative simulation each individual in the new generation is based on exactly one individual from the previous iteration.

In the third phase of the thesis we have investigated and given an idea about the improved simulation based analysis method, discussing the steps involved in the iterative simulation process in detail including important parameters for this method. Future work is to implement this method and identify good values for these parameters in different experiments.
6 REFERENCES


2. Real time scheduling-part1 by Damir isovic Mälardalen University.


10. Uppaal Website www.uppaal.com

11. Kronos website www-verimag.imag.fr/temporise/kronos


13. Simulation Languages from Wikipedia, the free encyclopaedia http://en.wikipedia.org/wiki/Simulation_language


17. Genetic algorithms from Wikipedia, the free encyclopaedia http://en.wikipedia.org/wiki/Genetic_algorithm


25. Whitepaper-Pseudo random number generators by jansuku ouspg weblink: http://randomnumber.org/bibliography.htm
