

**A probabilistic approach to performance estimation at the
un-timed communication abstraction level on system-level design**

by

Ramón A. Mercado Reyes

A research proposal submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Engineering

Program of Study Committee:
Diane T. Rover, Major Professor
Akhilesh Tyagi
Arun Somani
Joseph Zambreno
Zhao Zhang
Ying Cai

Iowa State University

Ames, Iowa

2009

Copyright © Ramón A. Mercado Reyes, 2009. All rights reserved.

TABLE OF CONTENTS

LIST OF FIGURES	iv
ABSTRACT	v
CHAPTER 1. Introduction	1
1.1 Problem Statement	3
1.2 Objectives	4
1.3 Expected Contributions	4
CHAPTER 2. Related Work	6
2.1 System Level Design	6
2.2 System Level Communication Modeling	9
2.3 System Level Performance Estimation	11
CHAPTER 3. Probability as a System Metric	14
3.1 Communication Architecture Design Alternatives	14
3.1.1 System Performance	15
3.2 Communication Modeling at Higher Abstraction Levels	17
3.2.1 Performance Estimation for Un-timed Communication Models	17
3.2.2 The Probabilistic Metric	19
3.3 Probability Metric for System Level Design	21
CHAPTER 4. Work Plan	23
4.1 Research Questions	23
4.1.1 Probability Modeling	23
4.1.2 Probability Computation	24

4.1.3	Design Space Exploration	24
4.2	Work Outline	25
4.2.1	Probability Model	25
4.2.2	Probabilistic Metric	26
4.2.3	Design Space Exploration	27
4.3	Deliverables	28
4.4	Impact of the Proposed Work	29
BIBLIOGRAPHY		33

LIST OF FIGURES

1.1	Productivity Gap	1
2.1	SpecC Design Flow	7
2.2	Communication Design Space	8
3.1	3x3 Mesh Interconnection Network	15
3.2	Typical Routing Flow	16
3.3	Node P-Model	18
3.4	Random Variable Model	19
3.5	Paths p_m and p_n that share one channel.	20
3.6	System Level Exploration	21
3.7	Model to Implementation	22
4.1	Work Flow Relations	29
4.2	Research Time Table	32

ABSTRACT

Today's embedded system designers face the challenges of ever increasing complexity and shorter time-to-market deadlines. System-level methodologies emerge to meet these challenges. Refinement-based methodologies, such as the SpecC methodology and Transaction Level Modeling, continue to gain popularity in the embedded system designers' community. However, as more communication-dominated applications and architectures appear in the market, designers find that the lack of models allowing system-level communication analysis is a major limiting factor in current system-level design methodologies. Thus, modeling for system-level communication analysis is key for a design methodology to thrive with today's embedded system designers. This work presents a new approach to system-level modeling that allows better communication analysis earlier in the design process. This approach defines a new model that utilizes random variables to include the communication details at higher abstraction levels. This work proposes a probabilistic model to include and evaluate the system communication features in the higher abstraction level. Guidelines to include the propose model into a refinement-based methodology are presented, and methods for performance estimation are shown.

CHAPTER 1. Introduction

System complexity continues to grow according to the well-known Moore's law [41]. In contrast, designer productivity grows at a much smaller rate. The International Technology Roadmap [19] indicates a productivity growth of only 21% (designed transistors/staff-month) in recent years. This imbalance between complexity and productivity clearly shows a gap between the number of new transistors available and the complexity designers are capable of handling. This is known as the productivity gap and is shown in figure 1.1.

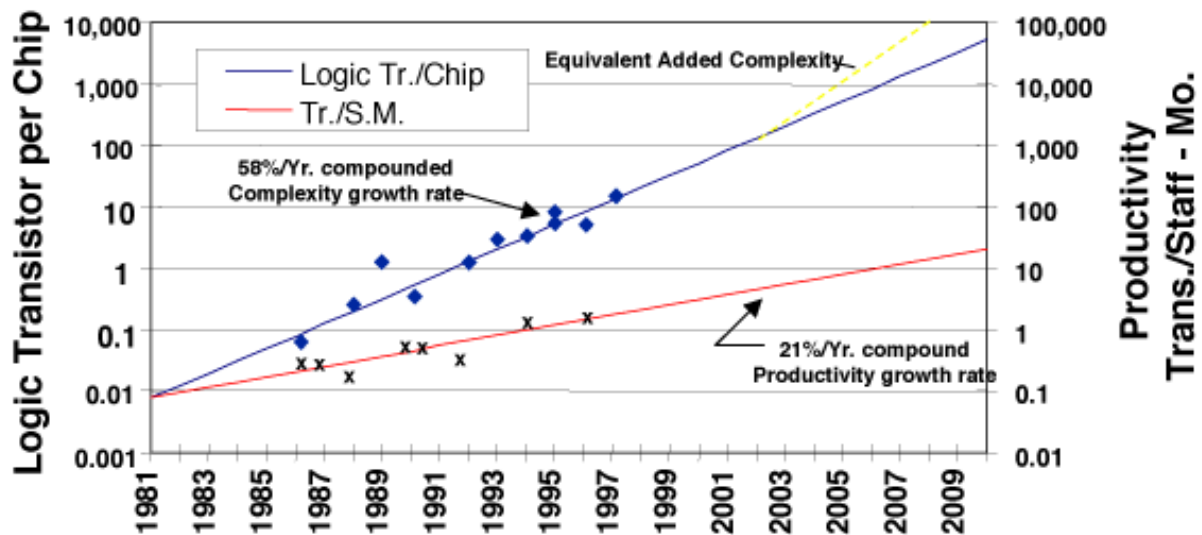


Figure 1.1 Productivity Gap

The expansion of the productivity gap results in increasing non-recurrent engineering (NRE) costs, and larger time-to-market periods. To deal with the productivity gap increase designers turn to System-level design [28]. System-level design tries to reduce the productivity gap by introducing new abstraction levels that allow designers to handle progressively more complex systems.

System-level design introduces higher abstraction levels that hide non-essential details from the designers. At a given abstraction level the system is represented by an abstracted model. The abstracted system models let designers evaluate larger areas of the design space more efficiently.

The highest level of abstraction in system-level design is the specification level. At this level, a specification model represents the functionality of the target application, and design constrains. However, none of the implementation details are included in the specification model. The specification model is the starting point for the design space exploration.

A useful representation of the design space is an orthogonal composition of the computation and communication design alternatives. This orthogonal relation between computation and communication is known as separation of concerns. Separation of concern allows the designer to explore the computation architecture neglecting the communication effects on the system design.

Traditionally system-level design searches the design space along the computation axis first. The result of this computation architecture exploration is the architecture model. The architecture model is generated from the specification model through some refinement process. The architecture model reflects the structure of the final implementation, and it provides high-level performance estimates of the computational complexity on the design.

With the structural details in place, the next step is to introduce the communication details. The architecture model is refined into a communication model through exploration of the design space along the communication axis. The communication model introduces more implementation details and allows more accurate an performance estimation. This model is capable of producing accurate performance estimate of the entire implementation architecture, accounting for the communication dependencies between the architectural blocks.

The manner in which the design space is explored, first through computation architecture exploration followed by the narrow communication exploration, reveals a computation-centric design. Computation-centric design is reasonable where higher computation power translates to higher performance. However, with the proliferation of embedded communication systems

(e.g. smart-phones, GPS, etc) and multi-core systems, improvements in computation power are negated by communication delays. Today's processors are capable of processing large amounts of data faster than the data can be transferred to the processor. The system-level design community reacts to this paradigm change and introduces new tools and methodologies for better communication analysis integration into the earlier phases of the design space exploration.

1.1 Problem Statement

Recently the area of system-level design has seen a migration from *computation-centric* design, to *communication-centric*. This migration is caused by the realization that communication is becoming the performance bottleneck in today's complex systems. System-level design can no longer perform architecture exploration neglecting the communication effects on the system performance.

Communication analysis must be included at the architecture abstraction level. Nevertheless, to perform communication analysis at higher abstraction levels is not trivial. The main challenge is the lack of timing information at these abstraction levels. Without this timing information, it is difficult to define communication performance metrics and acquire accurate performance estimates to guide the space exploration.

This research focuses on the methods, tools, and modeling guidelines needed to estimate communication performance at the abstraction levels where the required timing details are not available. The key issues are (1) the abstraction of the communication architecture features, and (2) performance estimation given the abstracted communication features.

Thesis Statement: *System-level communication characteristics provide meaningful information that in the past has only been used for interconnect optimization. Random variables and statistical methods may be used in a novel manner to estimate communication performance at higher levels of abstraction, where most communication details are not available. A new system design paradigm is defined that evaluates communication characteristics at higher levels of abstraction in the design methodology and performance analysis. New in this communication-*

centric design is the extraction of the application's communication behavior and the abstraction of the platform communication characteristics. The application communication behavior and platform communication characteristics may be introduced at higher levels of abstractions using random variables, and statistical methods can provide the tools to better estimate the system performance at these levels.

1.2 Objectives

Section 1.1 defined the two key issues addressed in this work. The abstraction of the communication features, to include them at higher abstraction levels; and communication performance estimation at these levels. The objectives of this research are as follow:

1. **To investigate innovative ways to include communication architecture features at higher abstraction levels.** More specific this research focuses on investigating how to use random variables to include communication architecture features in higher abstraction models. Random variables may be used when not enough information is available. At higher levels of abstraction most of the timing information required for performances evaluation is not available, this is where random variables become useful.
2. **To investigate methods for performance estimation with random variables.** The purpose of the random variable model based is to evaluate different communication events and architecture characteristics. The key to this evaluation is estimation. To correctly estimate the communication performance it is necessary to combine the dynamic communication behavior with the random variable based model. This research will investigate different methods for combining dynamic behaviors with the random variable model for computation performance estimation.

1.3 Expected Contributions

The expected contributions for this research are in two main areas. (1) Modeling to include communication details at higher abstractions levels, and (2) performance estimation for design

space exploration.

1. *Communication Modeling at Higher Abstraction Levels*

- **This research will provide a more complete model at higher abstraction levels.** These models will include more communication details than current models. Designers will use these new models to evaluate the communication effects earlier on the design processes.
- **This research will provide the tools and guidelines for building the high abstraction level communication models.** For the models built by this research to become useful to system designers, it is necessary to provide designers with the tools and guidelines for them to construct similar models for different systems.

2. *Performance Estimation*

- **This research will provide the methods for communication performance estimation for the new high abstraction level models.** For any model to be useful in system design, it must provide performance estimates of the details that it models. This research will provide the tools and methods for performance estimation from the new high abstraction level communication models.
- **This research will provide the methods for comparing the communication estimates to guide the design space exploration.** Performance estimates give information about the current model, but to perform design space exploration it is necessary to compare the estimates from different models. Traditionally estimates can be directly compared, but this is not so for the random variable based models. This research will provide the methods and tools to compare the estimates from the random variable models and perform the design space exploration.

The rest of this document is organized as follows. Chapter 2 presents the current state of the research in system-level design, and communication modeling and performance estimation. Chapter 3 introduces the proposed solution to the communication modeling problem. Finally, chapter 4 shows the roadmap this research will follow to produce the expected contributions.

CHAPTER 2. Related Work

System level design is characterized by the different abstraction levels and the refinement process that guides the designer from one abstraction level to the next. This chapter presents the state of the research on system level design, and especially how communication details are included throughout the design flow.

2.1 System Level Design

In the literature there are two major approaches to system level design, refinement-based[20] and platform-based[45]. For both design approaches, refinement and platform, the initial step is to represent the application(s) to be implemented as a specification model. The specification model is the highest level of abstraction containing the application behavior and the design constraints, but none of the implementation details. The difference between the two system level design approaches resides in the process by which the specification model is transformed into a system implementation meeting the design constraints.

The SpecC methodology [23] is probably the best well known example of a refinement-based design methodology. The SpecC design flow is shown in figure 2.1. The design starts with a specification model of the application. The specification model represents the behavior of the application and includes the design constraints (e.g. power, performance, area, etc). In the SpecC methodology the specification model is written in the SpecC language[21], other methods for specifying the system specification include MS Excel sheets[26], XML[43], and even UML[42].

Through the architecture exploration the specification model is refined into an architecture model. This architecture model is the second abstraction level in the SpecC methodology. At

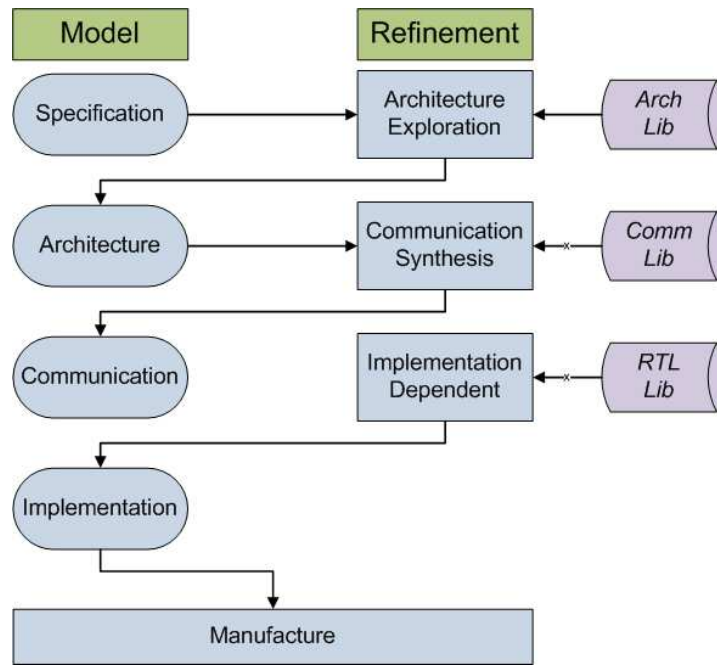


Figure 2.1 SpecC Design Flow

this level the system architecture is modeled as a collection of processing elements (PE) that are approximate-timed models of the computation elements of the architecture.

Cai et al. [11] show how the approximate timed PE models may be used for rapid design space exploration. This design space exploration depends on the performance estimates available at the current abstraction level. Cai [9] presents a method for performance estimation using weight-tables to represent the PE timing characteristics. Schneider [47] shows how to improve the method in [9] by introducing more accurate low-level aware metrics. After the system architecture is decided, the next step is to investigate the communication architecture.

The architecture model is refined into the communication model through the communication synthesis process. The communication synthesis refers to the narrow exploration of the communication design space, as shown in figure 2.2. In the SpecC methodology the broader search in figure 2.2 is, in fact, part of the architecture exploration. Other works that do broader communication exploration, as shown in figure 2.2, are introduced in later sections.

After the communication synthesis, the communication model reflects all the details of the final system implementation. With the resulting communication model, designers can collect

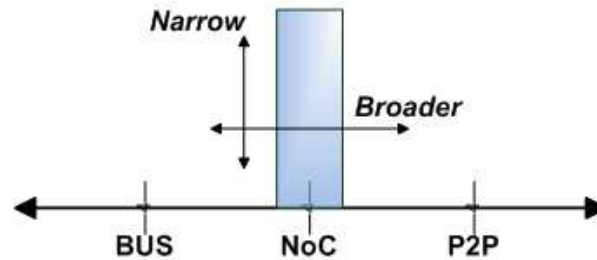


Figure 2.2 Communication Design Space

very accurate performance estimates to evaluate the final implementation before manufacturing. The final step is to convert the final communication model into a real implementation, this is done through the manufacturing process.

Another example of a refinement-based design approach is the Transaction Level Modeling (TLM) [10]. While not a methodology, in the strictest sense, TLM borrows a lot of concepts from the SpecC methodology. Like the SpecC methodology, TLM defines different abstraction levels and the amount of detail that a model at each level contains. Unlike SpecC, TLM does not define the refinement processes by which one model is transformed into another.

Instead TLM lets the designer use any refinements he/she may see fit for the current design. Several tools and methodologies became available to fill this gap in TLM. SystemC [1] is the tool most associated with TLM. SystemC is a C++ library that defines the constructs needed to build TLM models and includes a discrete event simulator used for executing SystemC models. Other SystemC-based TLM design tools and frameworks include Simics[39], CoWare[53], and SystemVerilog[48].

The second most common approach to system level design is the platform-based approach[45]. This approach differs from refinement-based in that the specification model is not recursively refined into an implementation. Instead, the specification model is directly mapped into a target platform that models all the implementation details. Performance estimates are gathered for the current mapping. If the design constraints are met the process stops, else another platform is considered, etc.

Metropolis [4] is an example of platform-based design framework. In Metropolis formal models are used to model each platform separately. The Metropolis methodology defines how

a specification model is mapped into one of the platform models.

All of the tools mentioned above, regardless of the approach, initially trim the design space evaluating the effects of the PEs and neglecting the communication cost. While this order may seem intuitive, current research shows the pitfalls of not considering the communication cost early in the design process. The next sections show what researchers are currently doing to include communication behavior at higher abstraction levels.

2.2 System Level Communication Modeling

In the literature there are two distinct approaches to system level communication modeling. The first approach is to model the application communication behavior and use this information to guide the communication architecture design. The second approach is to abstract the communication features of the target platform and include these features in higher level models.

Representative to the first approach Deb et al. [17] evaluate the impact of control and data flow for DSP applications on system design. Tedesco et al. [51] explore the impact of different traffic models for the same application, on the interconnect design, specifically on the quality of service (QoS). While this is not the first work to evaluate traffic models for interconnect design [31, 7, 24, 55], it is the first in evaluating the usefulness of the different models for a certain application class. Santi et al. [46] is another work on the impact of traffic on the QoS of the interconnect. Similar to [51], Santi et al. [46] characterize the traffic in terms of injection rates. What is new in this work is the use of traffic statistics to justify the need for QoS in the system implementation.

While the previous works used models to represent the applications, traces are also common on system design. Mahadevan et al. [40] presents a trace-based simulation environment. Unlike traditional trace-based design exploration, the traces used in this work are annotated relative timing. This relative timing information is used to map the trace to an architecture different than the reference design.

In system design, the application characteristics are also used for automated architecture generation. An example of this automated architecture generation is the two phase synthesis

flow of \times pipes[5]. The first phase is where the system constraints are specified and the application characteristics are introduced. The second phase is the automated process of NoC architecture generation. The use of application characteristics for automated architecture generation is different from previous works, where the application characteristics were used to directly evaluate some performance metrics. Ho and Pinkston [30] present another work where the communication characteristics are used for automated on-chip interconnection network architecture generation. Different from [5], Ho and Pinkston [30] focus only on well-behaved communication patterns.

Chandraiah et al. [14] show yet another approach to the application communication analysis. Instead of focusing on how to better represent the application behavior, Chandraiah et al. [14] addresses the issue of how to build the specification model to better include the application communication features. Based on the SpecC methodology[23], this work presents an automated process to convert an specification model with non-explicit communication through global variables, into a model with explicit through abstract channels.

The second approach to system level communication modeling is to evaluate the communication characteristics of the implementation platform and integrate these characteristics into the higher abstraction level models. Knudsen and Madsen [33] is one of the earlier attempts at integrating architecture details into the system design. As part of the LYCOS co-design framework [27], this work evaluates the timing information and implementation metrics (e.g. area, and power) for PCI and USB protocols and shows how to use these information to guide the partitioning step. Most recently, Pasricha et al. [44] takes a different approach, where instead of including the protocol timing information, the communication is evaluated in terms of transactions. Pasricha et al. [44] presents a model where (1) the communication behavior is characterized by the type of transactions, and (2) cycle accurate figures are know for each transaction type on the target platform.

In a more direct approach to communication architecture abstraction Kumar et al. [35] describes NoC architecture as a collection of communication resources and computation placeholders. The communication resources are further abstracted through the use of communica-

tion layers based on the OSI model. This layering approach to communication architecture abstraction has been adopted by others [49, 6, 25]. A similar approach is found in Coppola et al. [15], where a C++ library is introduced to facilitate the modeling of layered interconnection networks.

Other works look at the effects that different communication architectures have on an application or set of applications. For example, Lee et al. [37] evaluate different communication architectures for an implementation of the MPEG-2 video application. The application is implemented in three different communication architectures, bus, point-to-point (P2P), and NoC. This work is the first showing the true impact of these very different communication architectures in the system performance of one application. In a similar study, Bononi and Concer [8] evaluate several architectures and compares analytical versus quantitative results for a ring, 2D mesh, and the new spidergon mesh.

2.3 System Level Performance Estimation

While most work in the area of system level design tries to define new models or methods to include more information into the existing abstractions levels, all of this work aims at better performance estimation. Performance estimation is key to system level design space exploration. Simulation is the most common method for system level performance analysis.

Gajski et al. [23] is an example of simulation-based performance analysis. Simulation is used to introduce the application behavior, and the architecture details are included through back annotation of the timing details. In Gajski et al. [23] communication is included in the system performance only after the computation architecture has been explored and chosen. A similar approach is adopted by Baghdadi et al. [3]. However, Lahiri et al. [36] show the pitfalls of exploring the computation architecture without considering the communication cost.

In contrast Dey and Bommu [18] introduce a technique for estimating the communication performance of concurrent processes during the computation architecture exploration. Communication layers are defined as the relative times where the concurrent processes synchronize. Performance estimation is done in each layer separately. Another example of communication-

centric performance analysis is Loghi et al. [38]. Loghi et al. [38] presents a SystemC[2] on-chip communication simulation environment for multi-processors system-on-chip (MPSoC) architectures. Other examples of communication-centric performance analysis may be found in Kim et al. [32] and Fummi et al. [22].

In this research, a probabilistic approach is proposed for system level performance estimation. However, this research is not the first to propose such an approach. The works of Kumar et al. [34] and Sonntag et al. [50] are two good examples of probabilistic approaches to system level performance estimation.

Kumar et al. [34] evaluates the case where multiple application content for shared processing elements, and probability is used to estimate the system delay due to the contention for the shared computation resources. To use the probabilistic approach of [34] it is necessary to know (1) the application execution times on the processing elements, and (2) which application fractions content for the shared computation resources. Through the approach described in [34] it is possible to compute better than worst case estimates in a fraction of the time required to simulate the cycle-accurate design. The drawback is the amount of information required for this approach, since it is necessary to know the application execution times on the shared resources. Also, this method does not account for the communication overhead between the processing elements, or shared communication resources.

Another common probabilistic approach to system level performance estimation is queuing modeling, as it is done in SystemQ[50]. SystemQ is a SystemC[2] queuing-based simulation environment. The different abstraction levels are defined through queuing theory. Different to previous approaches SystemQ does incorporate communication effects into their design environment.

In SystemQ, separation of concerns[54] is defined along three orthogonal axes: function, structure, and communication. *Functional* refers to the algorithmic behavior, *structural* refers to the computational architecture, and *communication* refers to the communication architecture. A queuing model is built to represent the system at an abstraction level. Refinement steps are defined to transform the queuing model by adding *functional*, *structural*, and *com-*

munication details.

A system is refined throughout four levels of abstractions, named setup 1 through 4. Each setup adds details across one or more of the orthogonal concerns defined above.

[Setup 1] This is the highest level, most abstracted, and the entire system is modeled as a queuing network of two queuing systems, a producer and a consumer. Average service delays, derived from expert knowledge, are used in this model.

[Setup 2] This setup is generated through structural refinement of Setup 1. In Setup 2 each queuing system of Setup 1 is replaced by queuing network that reflects the structural details of the implementation platform.

[Setup 3] Functional and communication refinements are applied to generate Setup 3. The service time in this setup is determine for variable size packets, instead of using average packet length as in Setup 2. Communication is refined to account for mean arbitration and contention delays in the target communication architecture.

[Setup 4] This last setup is the result of further structural refinement to Setup 3. Setup 4 includes shared components between the original producer and consumer.

This chapter showed the current state of the system level design research. In particular, it showed the trends on system level communication analysis, and the approaches for communication performance estimation. Subsequent chapters will show the details of the proposed approach.

CHAPTER 3. Probability as a System Metric

This research proposes the use of probability as a high level estimator for system performance. The use of probability as a metric stems from the lack of communication details at the higher abstraction levels. This chapter will show, in a matter of example, how the probability metric may be used to evaluate different design alternatives at the same abstraction level.

3.1 Communication Architecture Design Alternatives

For this discussion on performance estimation it is necessary to first understand the communication architecture design alternatives and their effects on the systems performance and cost. As the system model is refined, the relations between the design alternatives, system performance, and implementation cost become more concrete. This is intuitive, performance estimation is more accurate at lower abstraction levels due to the introduction of implementation details. This section focus on some of the relations between a subset of communication architecture design alternatives, whereas the next section shows why it is impossible to directly measure the effects of these design alternatives on system performance at higher abstraction levels.

The communication architecture design space is a three-dimensional space along the topology, routing, and flow control axes. All of these communication architecture features has a direct influence on the system performance. The following is a list of the design alternatives and their broader effects on system performance and cost.

- **Topology** is the static arrangement of nodes and links. The topology has a direct effect on the throughput and latency of the interconnection network. The topology cost is reflected in the number and complexity of communication components, and in the

density and length of the interconnections.

- **Routing** is a policy for selecting a communication path from those available on the topology. While in a lesser degree than topology, routing still has a direct effect on the interconnect throughput and latency. Routing costs are measured in the node complexity.
- **Flow Control** represent the policies for resource allocation. Flow control is probably the feature that has the biggest dynamic impact on the communication performance, since it handles contention resolution. As for routing, the cost of flow control implementation is measured in the node complexity.

3.1.1 System Performance

System performance is affected by the throughput and latency of the interconnection network. The rest of this chapter only deals with latency. Similar analysis may be done for throughput.

The key issue in system design is the relation between the design alternatives, system performance, and implementation cost. In the case of latency, and for the sample system of figure 3.1, a well known relation is the zero-load latency.

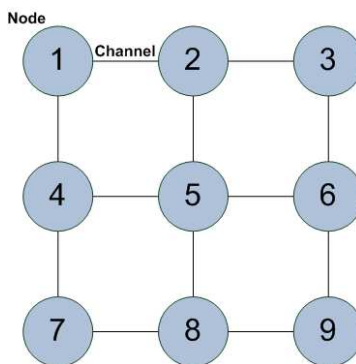


Figure 3.1 3x3 Mesh Interconnection Network

Zero-load latency, T_0 , may be defined as [16]

$$T_0 = H_0 \left(t_{0r} + \frac{L}{BW} \right). \quad (3.1)$$

H_0 is the number of nodes between source and destination, this is as much influenced by topology as by routing. L represents the size of the packet in terms of flits. L is determined by the flow control policy. BW is the bandwidth of the communication channels. Finally, t_{0r} , known as routing time, refers to the time a packet resides in an intermediate node in the network. Routing time is a function of the topology, routing, flow control, and even traffic.

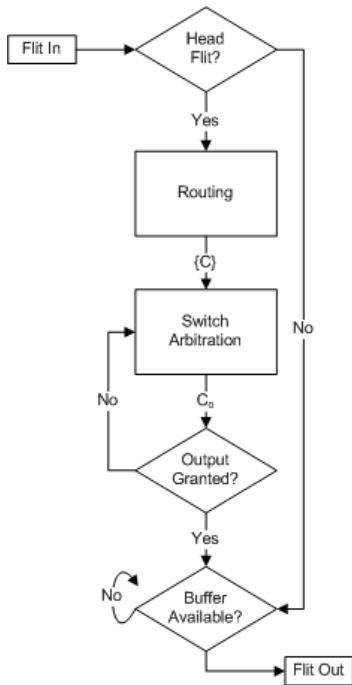


Figure 3.2 Typical Routing Flow

Equation (3.2) is the total path latency for a path with $H = n$ when the network load is not zero. In the current example t_{ir} is the routing time on each node, and it represents the implementation of the routing flow of figure 3.2.

From figure 3.2 it is clear how the design alternatives on topology, routing, and flow control have a direct effect on t_{ir} . For example, topology and routing determine the amount of time a head flit spent in the *Routing* process. Further, the time required for switching arbitration is a function of flow control as well as topology, routing, and load. Finally, the time a flit waits for buffers to become available depends on flow control and load.

For the simple example of figure 3.1, let's consider the typical routing flow shown in figure 3.2. This routing flow allocates resources at the flit level. For a common implementation of the routing flow of figure 3.2 routing time (t_r) may be between 40 to 50 cycles, depending on the number of available output channels and buffers[29].

Because the analysis so far focuses on zero-load latency the effects of traffic are abstracted from the latency computation and every component of equation (3.1) becomes constant. However, if we consider that in the presence of traffic the routing time may vary on each node, the following relation is derived:

$$T_n = \sum_{i=0}^n \left(t_{ir} + \frac{L}{BW} \right) \quad (3.2)$$

3.2 Communication Modeling at Higher Abstraction Levels

This work focuses on performance estimation at the highest levels of abstraction. These abstraction levels are characterized by un-timed (or approximate-timed) computation, and un-timed communication. This section presents the key issues that make communication performance estimation problematic at these high abstraction levels, and how to use probability as a metric to overcome these issues.

3.2.1 Performance Estimation for Un-timed Communication Models

As shown in section 3.1.1 the performance for the sample system of figure 3.1 is determined by the path latency of equation (3.2). This section presents how the different components of equation (3.2) are found in an un-timed communication model. Initially, it is necessary to define the un-timed communication model.

As defined by [10], a model with un-timed communication is characterized by concurrently executing processing elements and communication through abstract channels. These channels are message passing channels, which only represent data transfer or synchronization between processing elements. No timing information about the communication architecture is included in either the processing elements or the channels.

Using the previous definition, equation (3.2) is evaluated for the un-timed communication model of the sample system in figure 3.1. On each node a packet is delayed by the routing time, t_{ir} , and the travel time, $t_t = \frac{L}{BW}$. This example focus on routing time, and it is assumed that L and BW are known.

Routing time is determined by the implementation of the routing flow (figure 3.2) in the routing nodes. For the case of un-timed models, only the behavior of the routing flow is included. This means that whenever a packet arrives at a routing node the routing decision is made instantaneous, furthermore, because a packet is never held by the routing node there is never contention for the node resources by other packages.

From the previous analysis, it is clear that, $t_{ir} = 0$ for the un-timed model. Unfortunately

this reduces path delay to

$$\begin{aligned} T_n &= \sum_{i=0}^n \left(\frac{L}{BW} \right) \\ &= H \left(\frac{L}{BW} \right) \end{aligned}$$

which is clearly not a useful relation for accurate performance estimation. Further, the number of nodes in the path, H , may change depending on dynamic effects due to traffic.

To proceed with the system design, it is necessary to include timing details of the routing implementation without moving away from the un-timed communication model. The proposed solution is to use a probabilistic timing model, or p-timed model. Contrary to an approximate-timed model, a p-timed model does not use back annotation to include timing information. Instead a p-timed model relies on a probabilistic description of the target implementation.

For the current example a good probabilistic description of the routing flow of figure 3.2 is a discrete uniform distribution. Routing time may be expressed as a random variable of the form

$$t_r \sim U(45, 50),$$

or

$$t_r \sim U(0, 5), \tag{3.3}$$

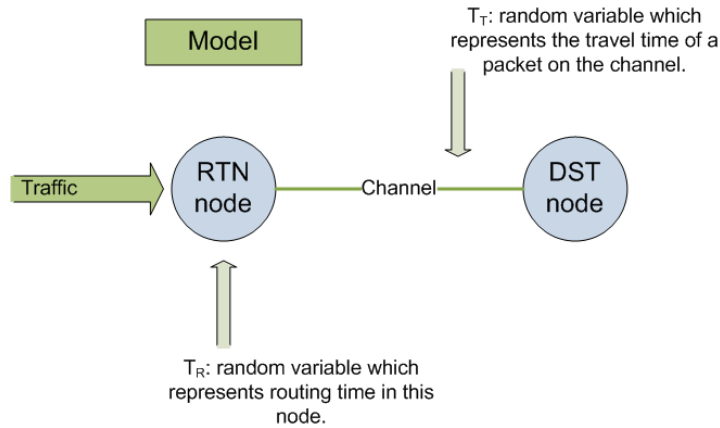


Figure 3.3 Node P-Model

since only the absolute difference is important¹. A refined node p-model is shown in figure 3.3 on page 18. With the definition (3.3) in hand the next step is to compute the path delay of equation (3.2).

Since $\frac{L}{BW}$ is assumed constant for the current analysis, the only part left to evaluate is path length H . For better system performance estimation, it is necessary to account for the dynamic behavior of H . That is, the path length is affected by the traffic and flow control policy.

Through simulation of the un-timed communication model it is possible to determine the zero-load paths $\{H_0\}$ for a given traffic. Figure 3.4 shows the simulation model. The final step to estimate the system performance is to combine the p-timed model with the set of zero-load paths $\{H_0\}$ to generate the probabilistic metric.

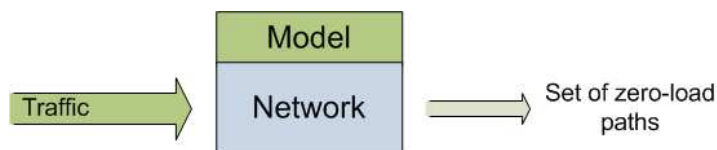


Figure 3.4 Random Variable Model

3.2.2 The Probabilistic Metric

Section 3.2 showed how to generate a p-model of a sample design, and how simulation may be used to include the dynamic effects of traffic. In this section everything comes together to generate the final performance estimate, i.e. probabilistic metric.

The first step is to partition the zero-load path set $\{H_0\}$ into smaller sets $\{h_0\}$. Where

¹Note that contention effects are included in t_i .

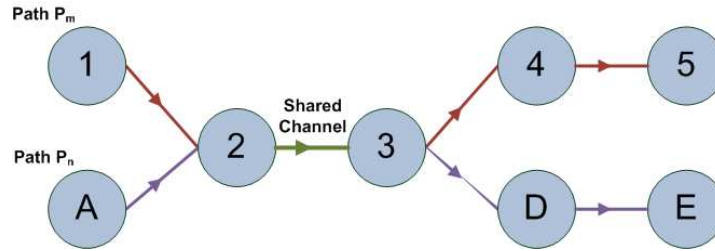


Figure 3.5 Paths p_m and p_n that share one channel.

$\{h_{0i}\}$ is a set of paths, such that:

$$\begin{aligned}
 H_0 &= \{h_{01}h_{02} \dots h_{0n}\}, \\
 \bigcap^n h_{0i} &= \emptyset, \\
 h_{0i} &= \{p_1p_2 \dots p_k\}, \\
 \bigcap^k p_i &\neq \emptyset, \\
 P_c(p_m, p_n) &> 0.
 \end{aligned}$$

Each path p_i is the zero-load path for a given packet, and $P_c(p_m, p_n)$ is the probability that the packet in path p_m collides with the packet in path p_n . Therefore, only paths carrying packets that have a probability of collision greater than zero ($P_c > 0$)² may belong to the same set (h_{0i}).

Given packets A and B with respective paths p_m and p_n , shown in figure 3.5, collision will occur if both packets require the shared link at the same time T . Packet A utilizes the shared link at time T_A , given by

$$T_A = t_{1r} + t_{2r}, \quad (3.4)$$

where t_{1r} and t_{2r} represent the routing times for nodes 1 and 2 respectively. From the analysis in section 3.2.1

$$t_{1r} = t_{2r} = t_r \approx U(1, 5), \quad (3.5)$$

and with equations (3.4) and (3.5) it is possible to compute the probability of collision $P_c(p_n, p_m) = P(T_A = T_B) = 1/7$.

²It's important to talk about how I will ensure this constrain, since this is a big part of the research.

The same analysis is done through the paths of $\{h_0\}$ to compute the probability of collision $P_c(h_0)$. The process is then repeated for all the sets in $\{H_0\}$. This produces a probability mass function which represents the system at the current level of abstraction, a probabilistic metric.

This section presented a sample system at the un-timed communication abstraction level, discussed a communication architecture option for routing, and showed the limitations for performance analysis at the current abstraction level. To overcome these limitations a probabilistic approach is proposed. Lastly, a probabilistic metric is derived using the probabilistic representation of the architecture features, and simulation to include the dynamic behavior of traffic.

3.3 Probability Metric for System Level Design

Previous sections showed how to compute the probabilistic metric for an specific example. This section shows how this metric may be use for system level design.

Recalling from section 3.2.2, it is possible to compute a probabilistic metric for a set of architecture options and a given traffic pattern. Figure 3.6 shows how different p-models represent certain architecture details, and a probabilistic metric is generated for the every combination of p-model and traffic. The probability metric, then, becomes a representation of the chosen architecture features and their interaction with the traffic pattern.

System design now continues by comparing the probabilistic metrics for different model-traffic combinations, P_c^1 through P_c^N , in figure 3.6. The key to this probabil-

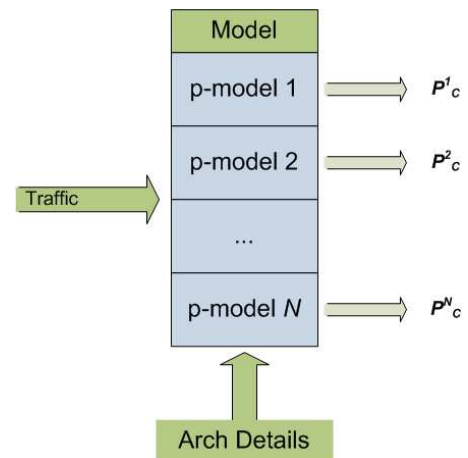


Figure 3.6 System Level Exploration

ity driven system design is the relation between the high level probability of the model and the low level performance of the implementation, as show in figure 3.7.

This relation between probability and performance is one of fidelity. That is, there is a

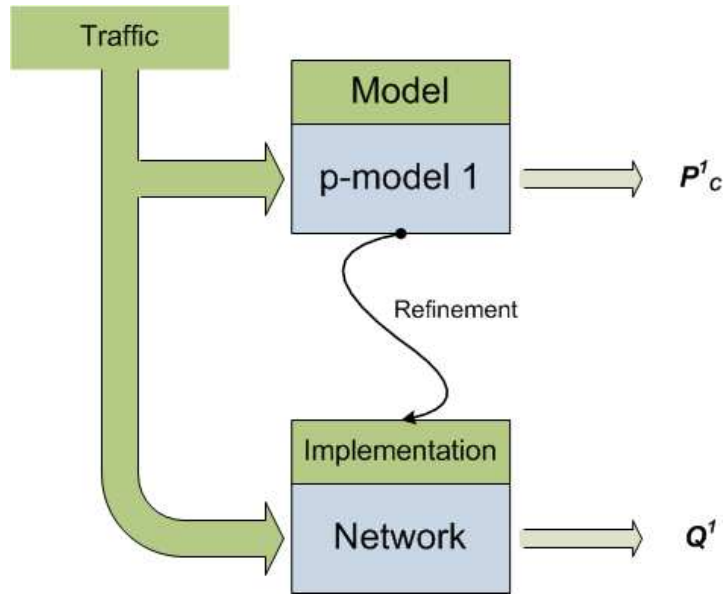


Figure 3.7 Model to Implementation

relation for comparing P_c^1 , P_c^2 , and P_c^3 , such that

$$P_c^1 > P_c^2 > P_c^3 \implies Q^1 > Q^2 > Q^3. \quad (3.6)$$

The derivation of this relation is not trivial and may be dependent on the architectural features under testing.

This chapter introduced the major limitations for performance estimation at the un-timed communication abstraction level. A probabilistic approach to system modeling is proposed to surmount these limitations, and through an example it was shown how the proposed approach may be used for performance estimation by generating a probabilistic metric. Lastly, it was shown how the probabilistic metric may be used to guide the system level design.

CHAPTER 4. Work Plan

The previous chapter showed an example of how to compute the probabilistic metric for an specific architecture feature, routing time. Further, chapter 3 introduced the probabilistic timing (p-timed) model, where a random variable is used to include the routing characteristics into an untimed communication model. Several questions remain open in regard to the p-timed model. This chapter addresses these questions and presents a roadmap for answering them.

4.1 Research Questions

This section presents, in detail, the research questions this research answers. These questions span three major issues *modeling*, *performance estimation*, and *design space exploration*.

4.1.1 Probability Modeling

As shown in section 3.2.1 correct probability modeling is critical for the p-timed model. In the sample design of chapter 3 routing time t_r is modeled as a uniformly distributed discrete random variable. Uniformly distributed random variables are known to be good first approaches [13], when little is known about the system. However, it is not clear that this is the best model for routing time.

- **Which distribution best fits a certain architecture feature?** To answer this question it is necessary to evaluate different probability models for several architecture features, and their impact on the probabilistic metric. This research will focus on routing and flow control protocols only. I will investigate different oblivious and adaptive routing algorithms, as well as packet and flit-based flow control protocols. To determine how to best describe these protocols as probability mass functions.

4.1.2 Probability Computation

Section 3.2.2 showed how to compute the probabilistic metric for a sample design. The computation on section 3.2.2 uses simulation information, zero-load path, as well as the p-timed model distribution. Nevertheless, the key issue in this computation is the relation between the high level probabilistic metric and the low level performance metric.

As seen in section 3.3 the probabilistic metric is only useful if it holds the fidelity relation of equation (3.6). Therefore to show the viability of the proposed probability approach it is necessary to show a correlated case of the probability of collision and the low level metrics. The following questions are inferred from this discussion.

- **Which probability computation method produces the best probabilistic metric?** Depending on the probability functions from section 4.1.1, it is necessary to investigate different quantitative and analytical tools to compute the probabilistic metric. Moreover, the probability computation also depends on the information available from the system simulation, as well as observed network event.
- **What information is necessary in the p-timed model for the probability computation?** As defined in section 3.2.1 the p-timed model uses a random variable to include a certain architecture feature, and using simulation introduces the dynamic traffic behavior. Section 4.1.1 investigates the first part of the p-timed model, the probability modeling. However, traffic considerations are also critical for the probability computation [12, 52]. This research will investigate the effect of different traffic models (e.g. random, transpose, etc) on the probabilistic metric.

4.1.3 Design Space Exploration

All the effort to include architecture features at higher abstraction levels is meaningless unless it serves to make design decisions and explore the design space. The key to making good design decisions is the fidelity. As shown in section 3.3 it is necessary to have a method for comparing the probabilistic metrics such that equation (3.6) holds.

- **What methods are available for comparing the probabilistic metrics?** Statistics are the most common methods for comparing probabilities. Familiar statistics are the mean and standard deviation, but the usefulness of these statistics greatly dependent on the problem at hand. This research will investigate several statistics for comparing the probabilistic metrics, and their fidelity relations to the low level metrics.

4.2 Work Outline

The previous section introduced the specific questions addressed by this research. This section outlines how this research will answer the previous questions. Like the research questions, the outline is organized along the three major issues *modeling*, *performance estimation*, and *design space exploration*.

4.2.1 Probability Model

The probability models consist of a random variable characterized by a probability distribution. This research will investigate routing and switching protocols, and will determine which random variable best models the different protocols. Specifically, this research will investigate the following routing protocols and switching schemes.

- Routing Protocols:
 1. Deterministic - XY, west first, north last, and negative
 2. Random - uniformly random, and weighted random
- Switching Schemes:
 1. Packet-based - store-and-forward, and cut-through
 2. Flit-based - wormhole, and virtual channel

Experiment I Setup: The goal of this experiment is to find the probability distribution that best models a given routing/switching combination. This research will investigate 6 routing protocols and 4 switching schemes, for a total of 24 possible combinations.

1. For a given routing/switching combination build the probability distribution from empirical data, if possible. If empirical data is not available, use a uniformly distributed distribution as an initial approximation.
2. Simulate the p-timed model to generate the zero-load path set $\{H_0\}$, as in section 3.2.2.
3. Compute the probability of collision. The probability of collision (P_c) is determined combining the probability distribution in step (1) and the path set in step (2). How P_c is computed depends mainly on the model distribution. Depending on the distribution, P_c could be a close form function, or it can require numerical methods.
4. Compare the probability of collision, the resulting distribution as well as the mean and standard deviation, to the actual number of collisions from an implementation, or cycle accurate simulator.
 - A good probability model produces a probability of collision that accurately characterizes the implementation's collision behavior.

4.2.2 Probabilistic Metric

The probabilistic metric depends on the probability distribution of the models, and the simulation traffic. This research focuses on 4 traffic mode:

- random,
- transpose (two kinds), and
- hot-spot (many-to-one).

The experiments in this section evaluate the effects of these traffics on the probabilistic metric, and analyze probabilistic metrics for the packet dropped and packet misroute network events.

Experiment II Setup: The goal of this experiment is to evaluate the effects of the different traffic patterns on the probability computation.

1. Using a p-timed model from Experiment I, simulate to generate the zero-load path set $\{H_0\}$ for every traffic pattern.
2. Compute the probability of collision, P_c .
3. Compare the probability of collision, the resulting distribution as well as the mean and standard deviation, to the actual number of collisions from an implementation, or cycle accurate simulator.

4.2.3 Design Space Exploration

The key to design space exploration is the fidelity of the performance estimates. For the proposed p-timed model, the performance estimate is the probabilistic metric. Unlike other traditional models, for the p-timed model the performance estimate cannot be directly compared to the low level performance metrics. This research will evaluate how different statistics may be used to relate the probabilistic metric to the low level metrics.

Experiment III Setup: The goal of experiment III is to determine the fidelity of a given set of statistics, to guide the design space exploration.

1. Select a set of p-time models. For example:
 - **Model A** - XY routing and wormhole switching,
 - **Model B** - negative routing and virtual channel switching,
 - **Model C** - weighted random routing and cut-through switching,
2. Simulate each p-timed model to generate the zero-load path set $\{H_0\}$.
3. Compute the probability of collision, P_c , for each model.
4. For each model compute several statistics:
 - central tendency,
 - standard deviation,

- skewness,
 - variance, and
 - kurtosis.
5. Analyze the relations of these statistics across the p-timed models, and compare these relations to the implementation performance metrics. Specifically the communication metrics of latency, and throughput, and the system costs in area, and power¹.

4.3 Deliverables

Section 4.1 presents the detail questions addressed by this research, and section 4.2 outlines the roadmap to tackle these questions. This section shows the models and tools that will result from the completion of this proposed research.

- **Probabilistic Models:** in its completion this research will deliver p-timed models for the 24 routing/switching combinations of section 4.2.1.
- **Simulation Framework:** this research will produce a simulation framework for the p-timed models. This framework includes:
 - SystemC node models that implement the routing and switching behaviors,
 - tools for traffic patten modifications, and
 - tools for path information generation and extraction.
- **Probabilistic Metric:** an important end product of this research is the probabilistic metric. This research will provide the designer with the tools and methods for computing the probabilistic metric for the routing/switching combinations of section 4.2.1. More precisely, this research will provide rules and guidelines for computing the probabilistic metric for the p-timed models and simulation paths from the previously described framework.

¹Area and power are measured for the routing and switching implementation on the routing nodes only.

- **Design Space Exploration:** at the culmination of this research the designer will have a set of statistics to guide the space exploration.

Figure 4.1 depicts a graphical representation of the relations between the work outlined in the previous section and the deliverables presented here. The colored bars indicate which deliverables are produced by which experiment. The gradients in the bars indicate that certain experiments are design for a particular deliverable, but the darker area, but its results affect other deliverables, the lighter area. The time table for this research is shown in figure 4.2 on page 32.

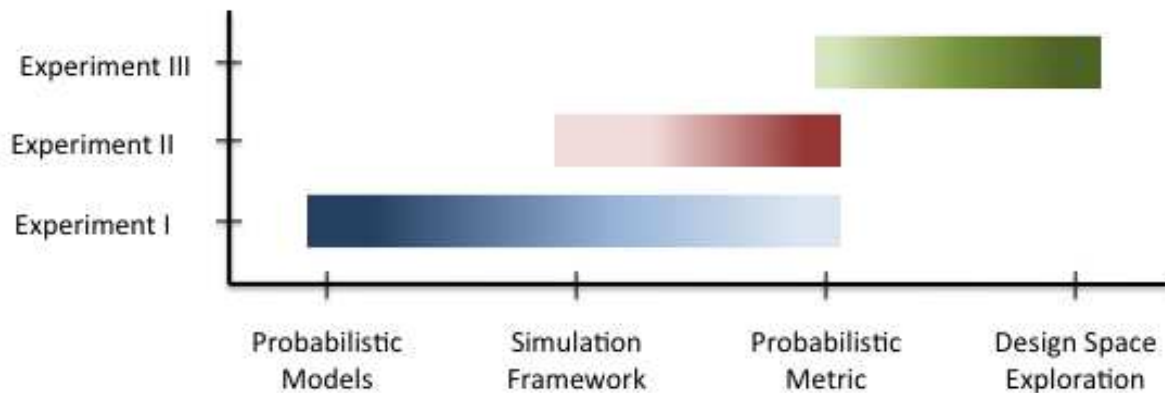


Figure 4.1 Work Flow Relations

4.4 Impact of the Proposed Work

Section 4.1 introduced the specific research questions addressed in this research. Section 4.2 outlines the tasks required to answer these questions, and section 4.3 shows the tangible results of this research. This section presents the impact this research will have in the research and design community. This research impacts three major areas: *system modeling*, *performance estimation*, and *design space exploration*.

System Modeling

- Shown in section 3.2, a p-time model is built using a random variable to represent the communication architecture features for which very little details are available. This research provides 24 new un-timed communication models that represent several common

communication choices. These models give designer the needed ability to perform early communication analysis.

- This research provides new models ready to use in system-level design, but more importantly, this research presents how the probabilistic models are derived. Probability distributions are derived from empirical data, or modeled by known density functions, section 4.2.1. The experiment outlined in section 4.2.1 serves as a guideline for designers to build their own p-timed models, and further improve the un-timed communication models for better communication analysis.

Performance Estimation

- The probabilistic metric combines the proposed p-timed model with the traffic characteristics to produce a communication performance estimator at higher abstraction levels, than any previous research up to date. Using the probabilistic metric, designers can evaluate the communication effects earlier on the design processes. As shown in section 1.1 early communication analysis is necessary for today's embedded systems. The probabilistic metric supplies the system level designer with the capacity to do this necessary early communication analysis.
- This research provides new tools for communication performance. As shown in section 3.2.2, the probabilistic metric incorporates the traffic behavior through simulation. A deliverable of this research is a SystemC simulation framework. This framework integrates the p-time models behavior, and produces the necessary data for the probabilistic metric computation. With this simulation framework designers and researchers can use the provided models for system design, or any new models they derived.

Design Space Exploration

- This research contributes with new models and performance estimation tools for system-level design. Performance estimates give information about the current model, but to perform design space exploration it is necessary to compare the estimates from different

models. This research provides a set of statistics designers will use to guide the design space exploration. With this statistics designers can include the communication effects during the early architecture exploration, a major problem in today's system-level design, as pointed out in section 1.1

This document starts by pointing out the productivity gap of figure 1.1 in page 1. System-level design, with its abstraction levels and refinement methodologies is the only solution to the productivity gap. To further improve system-level design it is necessary to include communication modeling and analysis in the higher abstraction levels, but this is difficult due to the lack of communication details at these higher levels. This research addresses this challenge and proposes a random variable based model to overcome this lack of details.

The proposed random variable based model, and resulting modeling methods and performance analysis tools, form a new system-level simulation environment through which communication analysis is moved to the higher abstraction levels. This new un-timed communication analysis enables system designer to manage higher complexity, directly resulting in a reduction of the productivity gap.

Figure 4.2 Research Time Table

Research Tasks	Task Timeline					
	1 Mo	2 Mo	3 Mo	4 Mo	5 Mo	6 Mo
Deliverables						
Probabilistic Models	•	•				
SystemC simulation environment	•	•	•			
Probabilistic Metric		•	•	•	•	
Design Space Exploration				•	•	•
SystemC Simulation Environment						
Build SystemC models for behavioral simulation	•	•				
Extend simulation environment to include all 4 traffic patterns	•	•	•			
Experiment I: Probability Model						
Gather probability distributions for all routing/switching combinations	•	•				
Run behavioral simulations and gather path information	•	•				
Compute probabilities		•	•			
Validate probability figures		•	•	•		
Experiment II: Probabilistic Metric						
Run behavioral simulations and gather path information for new traffic patters			•			
Compute probabilities			•	•		
Compute based statistics and validate probability figures			•	•	•	
Experiment III: Design Space Exploration						
Run behavioral simulations and compute probability figures for selected models				•		
Compute the second level statistics				•	•	
Evaluate the statistics relations				•	•	•
Validate the statistics fidelity					•	•

BIBLIOGRAPHY

- [1] Guido Arnout. Systemc standard. In *ASP-DAC '00: Proceedings of the 2000 conference on Asia South Pacific design automation*, pages 573–578, New York, NY, USA, 2000. ACM. ISBN 0-7803-5974-7. doi: <http://doi.acm.org/10.1145/368434.368808>.
- [2] Guido Arnout. Systemc standard. In *ASP-DAC '00: Proceedings of the 2000 conference on Asia South Pacific design automation*, pages 573–578, New York, NY, USA, 2000. ACM. ISBN 0-7803-5974-7. doi: <http://doi.acm.org/10.1145/368434.368808>.
- [3] Amer Baghdadi, Nacer-Eddine Zergainoh, Wander O. Cesário, and Ahmed Amine Jerraya. Combining a performance estimation methodology with a hardware/software codesign flow supporting multiprocessor systems. *IEEE Trans. Softw. Eng.*, 28(9):822–831, 2002. ISSN 0098-5589. doi: <http://dx.doi.org/10.1109/TSE.2002.1033223>.
- [4] Felice Balarin, Yosinori Watanabe, Harry Hsieh, Luciano Lavagno, Claudio Passerone, and Alberto Sangiovanni-Vincentelli. Metropolis: An integrated electronic system design environment. *Computer*, 36(4):45–52, 2003. ISSN 0018-9162. doi: <http://dx.doi.org/10.1109/MC.2003.1193228>.
- [5] Luca Benini. Application specific noc design. In *DATE '06: Proceedings of the conference on Design, automation and test in Europe*, pages 491–495, 3001 Leuven, Belgium, Belgium, 2006. European Design and Automation Association. ISBN 3-9810801-0-6.
- [6] Luca Benini and Giovanni De Micheli. Networks on chips: A new soc paradigm. *Computer*, 35(1):70–78, 2002. ISSN 0018-9162. doi: <http://dx.doi.org/10.1109/2.976921>.
- [7] Evgeny Bolotin, Israel Cidon, Ran Ginosar, and Avinoam Kolodny. Qnoc: Qos architec-

- ture and design process for network on chip. *J. Syst. Archit.*, 50(2-3):105–128, 2004. ISSN 1383-7621. doi: <http://dx.doi.org/10.1016/j.sysarc.2003.07.004>.
- [8] Luciano Bononi and Nicola Concer. Simulation and analysis of network on chip architectures: ring, spidergon and 2d mesh. In *DATE '06: Proceedings of the conference on Design, automation and test in Europe*, pages 154–159, 3001 Leuven, Belgium, Belgium, 2006. European Design and Automation Association. ISBN 3-9810801-0-6.
- [9] Lukai Cai. *Estimation and exploration automation of system level design*. PhD thesis, 2004. Chair-Gajski,, Daniel.
- [10] Lukai Cai and Daniel Gajski. Transaction level modeling: an overview. In *CODES+ISSS '03: Proceedings of the 1st IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pages 19–24, New York, NY, USA, 2003. ACM Press. ISBN 1-58113-742-7. doi: <http://doi.acm.org/10.1145/944645.944651>.
- [11] Lukai Cai, Andreas Gerstlauer, and Daniel Gajski. Retargetable profiling for rapid, early system-level design space exploration. In *DAC '04: Proceedings of the 41st annual conference on Design automation*, pages 281–286, New York, NY, USA, 2004. ACM. ISBN 1-58113-828-8. doi: <http://doi.acm.org/10.1145/996566.996651>.
- [12] Lukai Cai, Andreas Gerstlauer, and Daniel Gajski. Multi-metric and multi-entity characterization of applications for early system design exploration. In *ASP-DAC '05: Proceedings of the 2005 conference on Asia South Pacific design automation*, pages 944–947, New York, NY, USA, 2005. ACM. ISBN 0-7803-8737-6. doi: <http://doi.acm.org/10.1145/1120725.1120763>.
- [13] George Casella and Roger L. Berger. *Statistical Inference*. Duxbury Press, second edition, 2001.
- [14] P. Chandraiah, Junyu Peng, and R. Domer. Creating explicit communication in soc models using interactive re-coding. pages 50–55, Jan. 2007. doi: 10.1109/ASPAC.2007.357791.

- [15] Marcello Coppola, Stephane Curaba, Miltos D. Grammatikakis, Giuseppe Maruccia, and Francesco Papariello. Occn: A network-on-chip modeling and simulation framework. In *DATE '04: Proceedings of the conference on Design, automation and test in Europe*, page 30174, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2085-5-3.
- [16] Williams James Dally and Brian Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers, San Francisco, CA, 2004.
- [17] Abhijit K. Deb, Johnny Öberg, and Axel Jantsch. Control and communication performance analysis of embedded dsp systems in the masic methodology. In *ISSS '01: Proceedings of the 14th international symposium on Systems synthesis*, pages 274–273, New York, NY, USA, 2001. ACM. ISBN 1-58113-418-5. doi: <http://doi.acm.org/10.1145/500001.500064>.
- [18] Sujit Dey and Surendra Bommur. Performance analysis of a system of communicating processes. In *ICCAD '97: Proceedings of the 1997 IEEE/ACM international conference on Computer-aided design*, pages 590–597, Washington, DC, USA, 1997. IEEE Computer Society. ISBN 0-8186-8200-0.
- [19] Don Edenfeld, Andrew B. Kahng, Mike Rodgers, and Yervant Zorian. 2003 technology roadmap for semiconductors. *Computer*, 37(1):47–56, 2004. ISSN 0018-9162. doi: <http://doi.ieeecomputersociety.org/10.1109/MC.2004.1260725>.
- [20] S. Edwards, L. Lavagno, E.A. Lee, and A. Sangiovanni-Vincentelli. Design of embedded systems: formal models, validation, and synthesis. *Proceedings of the IEEE*, 85(3):366–390, Mar 1997. ISSN 0018-9219. doi: 10.1109/5.558710.
- [21] Masahiro Fujita and Hiroshi Nakamura. The standard specc language. In *ISSS '01: Proceedings of the 14th international symposium on Systems synthesis*, pages 81–86, New York, NY, USA, 2001. ACM. ISBN 1-58113-418-5. doi: <http://doi.acm.org/10.1145/500001.500019>.

- [22] F. Fummi, D. Quaglia, and F. Stefanni. A systemc-based framework for modeling and simulation of networked embedded systems. pages 49–54, Sept. 2008. doi: 10.1109/FDL.2008.4641420.
- [23] Daniel D. Gajski, Rainer Dömer, Junyu Peng, and Andreas Gerstlauer. *System Design: A Practical Guide with Specc*. Kluwer Academic Publishers, Norwell, MA, USA, 2001. ISBN 0792373871.
- [24] N. Genko, D. Atienza, G. De Micheli, J. M. Mendias, R. Hermida, and F. Catthoor. A complete network-on-chip emulation framework. In *DATE '05: Proceedings of the conference on Design, Automation and Test in Europe*, pages 246–251, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2288-2. doi: <http://dx.doi.org/10.1109/DATE.2005.5>.
- [25] Andreas Gerstlauer, Dongwan Shin, Rainer Dömer, and Daniel D. Gajski. System-level communication modeling for network-on-chip synthesis. In *ASP-DAC '05: Proceedings of the 2005 conference on Asia South Pacific design automation*, pages 45–48, New York, NY, USA, 2005. ACM. ISBN 0-7803-8737-6. doi: <http://doi.acm.org/10.1145/1120725.1120740>.
- [26] Kees Goossens, John Dielissen, Om Prakash Gangwal, Santiago Gonzalez Pestana, Andrei Radulescu, and Edwin Rijpkema. A design flow for application-specific networks on chip with guaranteed performance to accelerate soc design and verification. In *DATE '05: Proceedings of the conference on Design, Automation and Test in Europe*, pages 1182–1187, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2288-2. doi: <http://dx.doi.org/10.1109/DATE.2005.11>.
- [27] J. Grode, P. V. Knudsen, and J. Madsen. Hardware resource allocation for hardware/software partitioning in the lycos system. In *DATE '98: Proceedings of the conference on Design, automation and test in Europe*, pages 22–27, Washington, DC, USA, 1998. IEEE Computer Society. ISBN 0-8186-8359-7.

- [28] Omar Hammami and Muhammad Omer Cheema. Graduate education to fight system level design productivity gap in soc design. In *MSE '07: Proceedings of the 2007 IEEE International Conference on Microelectronic Systems Education*, pages 45–46, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2849-X. doi: <http://dx.doi.org/10.1109/MSE.2007.46>.
- [29] H. Hashemi-Najafabadi, H. Sarbazi-Azad, and P. Rajabzadeh. An accurate performance model of fully adaptive routing in wormhole-switched two-dimensional mesh multicomputers. *Microprocess. Microsyst.*, 31(7):445–455, 2007. ISSN 0141-9331. doi: <http://dx.doi.org/10.1016/j.micpro.2006.12.006>.
- [30] Wai Hong Ho and Timothy Mark Pinkston. A methodology for designing efficient on-chip interconnects on well-behaved communication patterns. In *HPCA '03: Proceedings of the 9th International Symposium on High-Performance Computer Architecture*, page 377, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-1871-0.
- [31] Jingcao Hu and Radu Marculescu. Dyad: smart routing for networks-on-chip. In *DAC '04: Proceedings of the 41st annual conference on Design automation*, pages 260–263, New York, NY, USA, 2004. ACM. ISBN 1-58113-828-8. doi: <http://doi.acm.org/10.1145/996566.996638>.
- [32] Sungchan Kim, Chaeseok Im, and Soonhoi Ha. Efficient exploration of on-chip bus architectures and memory allocation. In *CODES+ISSS '04: Proceedings of the 2nd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pages 248–253, New York, NY, USA, 2004. ACM Press. ISBN 1-58113- 937-3. doi: <http://doi.acm.org/10.1145/1016720.1016779>.
- [33] Peter Voigt Knudsen and Jan Madsen. Integrating communication protocol selection with partitioning in hardware/software codesign. In *ISSS '98: Proceedings of the 11th international symposium on System synthesis*, pages 111–116, Washington, DC, USA, 1998. IEEE Computer Society. ISBN 0-8186-8623-5.

- [34] Akash Kumar, Bart Mesman, Henk Corporaal, Bart Theelen, and Yajun Ha. A probabilistic approach to model resource contention for performance estimation of multi-featured media devices. In *DAC '07: Proceedings of the 44th annual conference on Design automation*, pages 726–731, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-627-1. doi: <http://doi.acm.org/10.1145/1278480.1278662>.
- [35] Shashi Kumar, Acel Jantsch, Juha-Pekka Soininen, Martti Forsell, Mikael Millberg, Johny Öberg, Kari Tiensyrjä, and Ahmed Hemani. A network on chip architecture and design methodology. In *ISVLSI '02: Proceedings of the IEEE Computer Society Annual Symposium on VLSI*, page 117, Washington, DC, USA, 2002. IEEE Computer Society. ISBN 0-7695-1486-3.
- [36] Kanishka Lahiri, Anand Raghunathan, and Sujit Dey. Efficient exploration of the soc communication architecture design space. In *ICCAD '00: Proceedings of the 2000 IEEE/ACM international conference on Computer-aided design*, pages 424–430, Piscataway, NJ, USA, 2000. IEEE Press. ISBN 0-7803-6448-1.
- [37] Hyung Gyu Lee, Naehyuck Chang, Umit Y. Ogras, and Radu Marculescu. On-chip communication architecture exploration: A quantitative evaluation of point-to-point, bus, and network-on-chip approaches. *ACM Trans. Des. Autom. Electron. Syst.*, 12(3):23, 2007. ISSN 1084-4309. doi: <http://doi.acm.org/10.1145/1255456.1255460>.
- [38] Mirko Loghi, Federico Angiolini, Davide Bertozzi, Luca Benini, and Roberto Zafalon. Analyzing on-chip communication in a mp soc environment. In *DATE '04: Proceedings of the conference on Design, automation and test in Europe*, page 20752, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2085-5-2.
- [39] Peter S. Magnusson, Magnus Christensson, Jesper Eskilson, Daniel Forsgren, Gustav Hållberg, Johan Högberg, Fredrik Larsson, Andreas Moestedt, and Bengt Werner. Simics: A full system simulation platform. *Computer*, 35(2):50–58, 2002. ISSN 0018-9162. doi: <http://dx.doi.org/10.1109/2.982916>.

- [40] Shankar Mahadevan, Federico Angiolini, Michael Storgaard, Rasmus Grondahl Olsen, Jens Sparso, and Jan Madsen. A network traffic generator model for fast network-on-chip simulation. In *DATE '05: Proceedings of the conference on Design, Automation and Test in Europe*, pages 780–785, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2288-2. doi: <http://dx.doi.org/10.1109/DATE.2005.22>.
- [41] G.E. Moore. Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 86(1):82–85, Jan 1998. ISSN 0018-9219. doi: 10.1109/JPROC.1998.658762.
- [42] W. Mueller, A. Rosti, S. Bocchio, E. Riccobene, P. Scandurra, W. Dehaene, and Y. Vanderperren. Uml for esl design: basic principles, tools, and applications. In *ICCAD '06: Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design*, pages 73–80, New York, NY, USA, 2006. ACM. ISBN 1-59593-389-1. doi: <http://doi.acm.org/10.1145/1233501.1233518>.
- [43] Srinivasan Murali, Martijn Coenen, Andrei Radulescu, Kees Goossens, and Giovanni De Micheli. Mapping and configuration methods for multi-use-case networks on chips. In *ASP-DAC '06: Proceedings of the 2006 conference on Asia South Pacific design automation*, pages 146–151, Piscataway, NJ, USA, 2006. IEEE Press. ISBN 0-7803-9451-8. doi: <http://doi.acm.org/10.1145/1118299.1118344>.
- [44] Sudeep Pasricha, Nikil Dutt, and Mohamed Ben-Romdhane. Fast exploration of bus-based communication architectures at the ccatb abstraction. *Trans. on Embedded Computing Sys.*, 7(2):1–32, 2008. ISSN 1539-9087. doi: <http://doi.acm.org/10.1145/1331331.1331346>.
- [45] Alberto Sangiovanni-Vincentelli and Grant Martin. Platform-based design and software design methodology for embedded systems. *IEEE Des. Test*, 18(6):23–33, 2001. ISSN 0740-7475. doi: <http://dx.doi.org/10.1109/54.970421>.
- [46] S. Santi, B. Lin, L. Kocarev, G.M. Maggio, R. Rovatti, and G. Setti. On the impact of traffic statistics on quality of service for networks on chip. pages 2349–2352 Vol. 3, May 2005. doi: 10.1109/ISCAS.2005.1465096.

- [47] Joseph P. Schneider. Low-level estimation at high-levels of abstraction in system-level design. MS in Computer Engineering, Iowa State University, Ames, IA, USA, 2007.
- [48] Donatella Sciuto, Grant Martin, Wolfgang Rosenstiel, Stuart Swan, Frank Ghenassia, Peter Flake, and Johny Srouji. Systemc and systemverilog: Where do they fit? where are they going? In *DATE '04: Proceedings of the conference on Design, automation and test in Europe*, page 10122, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2085-5-1.
- [49] M. Sgroi, M. Sheets, A. Mihal, K. Keutzer, S. Malik, J. Rabaey, and A. Sangiovanni-Vencentelli. Addressing the system-on-a-chip interconnect woes through communication-based design. In *DAC '01: Proceedings of the 38th conference on Design automation*, pages 667–672, New York, NY, USA, 2001. ACM. ISBN 1-58113-297-2. doi: <http://doi.acm.org/10.1145/378239.379045>.
- [50] Sören Sonntag, Matthias Gries, and Christian Sauer. SystemQ: Bridging the gap between queuing-based performance evaluation and systemc. *Design Automation for Embedded Systems*, 11(2):91–117, September 2007. URL <http://dx.doi.org/10.1007/s10617-006-9002-3>.
- [51] Leonel Tedesco, Aline Mello, Leonardo Giacomet, Ney Calazans, and Fernando Moraes. Application driven traffic modeling for NoCs. In *SBCCI '06: Proceedings of the 19th annual symposium on Integrated circuits and systems design*, pages 62–67, New York, NY, USA, 2006. ACM Press. ISBN 1-59593-479-0. doi: <http://doi.acm.org/10.1145/1150343.1150364>.
- [52] Brian Towles and William J. Dally. Worst-case traffic for oblivious routing functions. In *SPAA '02: Proceedings of the fourteenth annual ACM symposium on Parallel algorithms and architectures*, pages 1–8, New York, NY, USA, 2002. ACM. ISBN 1-58113-529-7. doi: <http://doi.acm.org/10.1145/564870.564872>.
- [53] K. Van Rompaey, I. Bolsens, H. De Man, and D. Verkest. Coware—a design environment

for heterogenous hardware/software systems. In *EURO-DAC '96/EURO-VHDL '96: Proceedings of the conference on European design automation*, pages 252–257, Los Alamitos, CA, USA, 1996. IEEE Computer Society Press. ISBN 0-8186-7573-X.

- [54] Guang Yang, Alberto Sangiovanni-Vincentelli, Yosinori Watanabe, and Felice Balarin. Separation of concerns: overhead in modeling and efficient simulation techniques. In *EMSOFT '04: Proceedings of the 4th ACM international conference on Embedded software*, pages 44–53, New York, NY, USA, 2004. ACM. ISBN 1-58113-860-1. doi: <http://doi.acm.org/10.1145/1017753.1017765>.
- [55] Ki Hwan Yum, A. Vaidya, C.R. Das, and A. Sivasubramaniam. Investigating qos support for traffic mixes with the mediaworm router. pages 97–106, 2000. doi: 10.1109/HPCA.2000.824342.