José Eduardo Chiarelli Bueno Filho

**Multifractal traffic generator modeled at the transaction level for
integrated systems performance evaluation**

São Paulo
2017

José Eduardo Chiarelli Bueno Filho

**Multifractal traffic generator modeled at the transaction level for
integrated systems performance evaluation**

Dissertation presented to the Polytechnical
School of the Universidade de São Paulo
for the obtainment of the Title of Master in
Sciences.

Concentration Area:

Microelectronics

Supervisor:

Prof. Dr. Wang Jiang Chau

São Paulo
2017

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, _____ de _____ de _____

Assinatura do autor: _____

Assinatura do orientador: _____

Catálogo-na-publicação

Bueno Filho, José Eduardo Chiarelli

Multifractal traffic generator modeled at the transaction level for integrated systems performance evaluation / J. E. C. Bueno Filho -- versão corr. -- São Paulo, 2017.

111 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Sistemas Eletrônicos.

1.Sistemas integrados em larga escala 2.Fractais 3.Modelos em séries temporais I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Sistemas Eletrônicos II.t.

ACKNOWLEDGEMENTS

To Prof. Dr. Wang Jiang Chau, for the relentless and careful supervision, and for his constant patience and good humor.

To Jorge Gonzalez Reaño, for initiating me in the subject of traffic modeling and for all the offered support.

To my mother, father and brother, for the infinite faith in my efforts and capacity.

To all of my friends, for the relaxation moments and precious advices.

To all the members of the Grupo de Projeto de Sistemas Eletronicos Integrados e Software Aplicado (GSEIS), for the valuable conversations and advices.

To CNPq, for the financial support on these two years of study and work on my masters degree.

RESUMO

O presente trabalho visa oferecer uma contribuição para o aumentar a eficiência do fluxo de projeto de sistemas integrados, focando, especificamente, na avaliação do desempenho de suas estruturas de comunicação. É proposta a utilização de simulações com modelos no nível de transações (TLM), com o objetivo de se obter vantagens da redução de esforço e tempo de projeto oferecidos por esta abordagem. Dentro das propostas de análise de desempenho, a utilização de geradores de tráfego ao invés simulações de sistema completo tem sido adotada devido a sua maior eficiência no tempo. Trabalhos iniciais na geração de tráfego intrachip focaram-se em processos de Poisson e em modelos de Markov clássicos, os quais não capturam Dependência de Longa Duração (LRD). Este fato levou a adoção de modelos fractais/auto-similares. Avanços posteriores mostraram que o tráfego produzido pelos elementos de sistemas multiprocessados podem apresentar maior grau de complexidade, que pode ser atribuída à presença de características multifractais. Neste trabalho, é proposta uma metodologia para a avaliação de tráfego intrachip para o desenvolvimento de um gerador de tráfego TLM. As principais contribuições deste trabalho são uma análise detalhada das séries temporais de tráfego obtidas nas simulações TLM e o estudo dos efeitos que o gerador de tráfego exerce sobre estas simulações, se concentrando, principalmente, na relação entre precisão e aceleração da simulação. As análises propostas se baseiam no paradigma multifractal, o qual permite (1) um maior entendimento da natureza estatística do tráfego pelos desenvolvedores de sistemas, (2) a obtenção de uma representação precisa deste tráfego e (3) a construção de geradores de tráfego que substituam elementos processantes de maneira realista. Outra contribuição deste trabalho é a comparação do desempenho, no que concerne a precisão das séries de tráfego sintéticas obtidas, de modelos monofractais e multifractais. Todas as contribuições mencionadas foram agrupadas na metodologia detalhada, apresentada no presente documento, sobre a qual experimentos foram realizados.

Palavras-chave: Tráfego intrachip, modelagem no nível de transações (TLM), gerador de tráfego, modelo multifractal, relação precisão-aceleração.

ABSTRACT

The present work aims to provide a contribution to improve the efficiency the design flow of integrated systems, focusing, specifically, on the performance evaluation of its communication structures. The use of Transaction Level Modeling (TLM) is proposed, in order to take advantage of the reduction of design effort and time. Within the performance evaluation approaches, the utilization of traffic generators instead of full system simulations started to be adopted due to its higher time efficiency. Initial works on on-chip traffic generation focused on Poisson processes and classic Markovian models, which are unable to capture Long Range Dependence (LRD). This fact led to the adoption of fractal/self-similar models. Later advancements have shown that the traffic produced in multiprocessed systems can show higher degrees of complexity, what can be attributed to the presence multifractal characteristics. In this work, a methodology to evaluate the on-chip traffic and to the development of a transaction level traffic generator is proposed. The main contributions of this work are a detailed analysis of traffic time series obtained by TLM simulations and the study of the effects of the traffic generator on these simulations, concerning, mainly, the speedup-accuracy trade-off. The proposed analysis follow the multifractal paradigm, allowing system developers to (1) understand the statistical nature of on-chip traffic, (2) to obtain accurate representations of this traffic and (3) to build traffic generators that mimic processing elements realistically. Another contribution of this work is a comparison of the performance, considering the accuracy of the obtained synthetic traffic time series, between monofractal and multifractal models. All of the mentioned contributions were grouped throughout the detailed methodology presented on the present document, for which experiments were carried out.

Keywords: On-chip traffic, transaction level modeling, traffic generator, multifractal model, speedup-accuracy trade-off.

LIST OF FIGURES

1.1	ESL methodology	18
1.2	Bus classifications	21
1.3	Example of a Network-on-Chip (3×3 mesh topology)	22
1.4	Functioning principle of a traffic generator	24
1.5	Failure of Poisson based traffic modeling	26
1.6	Effect of the packet injection rate (λ) in the multifractality of a processing element traffic	27
2.1	Modeling accuracy of various approaches	33
2.2	Five iterations of the Sierpinsky triangle	34
2.3	Example of a statistically self-similar object	35
2.4	Scaling function ($\tau(q)$) of a multifractal process	37
2.5	Behavior of the autocorrelation function for different values of the Hurst exponent	40
2.6	Example of (smoothed) periodograms	41
2.7	Example of Abry-Veicht spectra	44
2.8	Example of multifractal spectra	45
3.1	Architecture of the platform for traffic measurement	53
3.2	Applications traffic series with detected phase	54
3.3	Applications traffic series Abry-Veicht spectra	54
3.4	Overview of the Transaction Generator tool	57
4.1	Methodology diagram	61
4.2	The TLM simulation platform	62
4.3	Traffic modeling formalism	63

4.4	Method for obtaining the aggregated throughput	64
4.5	Model fitting process	66
4.6	The traffic generator	67
4.7	Conversion method 1	68
4.8	Conversion method 2	68
4.9	SoCLib's CA VCI signaling scheme	70
5.1	Functioning of the MJPEG decoder	73
5.2	Functioning of the MFT application	73
5.3	Multifractal spectra of the TLM traffic profiles	75
5.4	Huffman application time series: synthetic (25ns TLM capture aggregation window) vs. CA	80
5.5	Huffman application time series: synthetic (500ns TLM capture aggregation window) vs. CA	81
5.6	MJPEG application time series: synthetic (25ns TLM capture aggregation window) vs. CA	82
5.7	MJPEG application time series: synthetic (500ns TLM capture aggregation window) vs. CA	83
5.8	MFT application time series: synthetic (25ns TLM capture aggregation window) vs. CA	84
5.9	MFT application time series: synthetic (500ns TLM capture aggregation window) vs. CA	85
5.10	Synthetic traffic profiles errors	88
A.1	Multifractal spectra of the CA traffic profiles	102
B.1	Huffman application time series: synthetic (50ns TLM capture aggregation window) vs. CA	103
B.2	Huffman application time series: synthetic (100ns TLM capture aggregation window) vs. CA	104

B.3	Huffman application time series: synthetic (200ns TLM capture aggregation window) vs. CA	105
B.4	MJPEG application time series: synthetic (50ns TLM capture aggregation window) vs. CA	106
B.5	MJPEG application time series: synthetic (100ns TLM capture aggregation window) vs. CA	107
B.6	MJPEG application time series: synthetic (200ns TLM capture aggregation window) vs. CA	108
B.7	MFT application time series: synthetic (50ns TLM capture aggregation window) vs. CA	109
B.8	MFT application time series: synthetic (100ns TLM capture aggregation window) vs. CA	110
B.9	MFT application time series: synthetic (200ns TLM capture aggregation window) vs. CA	111

LIST OF TABLES

4.1	SoCLib components	61
5.1	Simulation times	74
5.2	Hurst exponents of the traffic profiles	77
5.3	Proportions between the errors (auto-regressive/MWM)	89
5.4	Traffic generator simulation times (Huffman application)	90
5.5	Traffic generator simulation times (MJPEG application)	90
5.6	Traffic generator simulation times (MFT application)	90
5.7	Quantification of the speedup-accuracy trade-off	91

LIST OF ABBREVIATIONS

AR	Auto-Regressive
ARMA	Auto-Regressive Moving Average
ARFIMA	Auto-Regressive Fractionally Integrated Moving Average
ASIC	Application specific integrated circuits
ASIP	Application specific instruction processors
BCA	Bus Cycle Accurate
CA	Cycle Accurate
CMOS	Complimentary Metal Oxide Semiconductor
DSM	Deep Sub-Micron
DFT	Discrete Fourier Transform
DWT	Discrete Wavelet Transform
ESL	Electronic System Level
FIFO	First In First Out
fBm	Fractional Brownian Motion
fGn	Fractional Gaussian Noise
FI	Fractional Integration
GPP	General purpose processor
HDL	Hardware Description Language
IC	Integrated circuit
I/O	Input/Output
IID	Independent and Identically Distributed
IP	Intellectual Property
ISS	Instruction Set Simulator

LRD	Long Range Dependence
MFT	Micro Four Thirds
MA	Moving Average
MIMD	Multiple Instruction Multiple Data
MPSoC	Multiprocessor System-on-Chip
MRA	Multi Resolution Analysis
MTU	Maximum Transfer Unit
NoC	Network-on-Chip
PDES	Parallel Discrete Event Simulation
PV	Programmer's view
PVT	Programmer's view plus timing
RAM	Random Access Memory
RC	Reconfigurable core
RTL	Register Transfer Level
SMP	Symmetric Multi-processed System
SRD	Short Range Dependence
SoC	System-on-Chip
TLM	Transaction Level Model
TLM-DT	Transaction Level Model with Distributed Timing
VCD	Value Change Dump
VCI	Virtual Component Interconnect
VGMMN	VCI Generic Micro Network
XML	Extensible Markup Language

LIST OF SYMBOLS

$\hat{H}_{AV}$	Abry-Veicht Hurst exponent estimator
$S(j)$	Abry-Veicht Spectrum
$Y[k]$	Accumulated traffic volume (sampling process of $Y(t)$)
m	Aggregation factor
$X^m[k]$	Aggregated increment process
$\rho_X(\tau)$	Autocorrelation function
$\gamma_X(\tau)$	Autocovariance function
B	Backshift operator
dim_{box}	Box counting dimension
$d(\omega_j)$	Discrete Fourier Transform of the signal $X[k]$
$\stackrel{d}{=}$	Equality in distribution
$E()$	Expected value operator
ω_j	Fourier fundamental frequencies
ω	Frequency
$\Gamma(x)$	Gamma function
$H(q)$	Generalized Hurst exponent
H	Hurst exponent
$X[k]$	Increment process (sampling process of $X(t)$)
$f_L(\xi)$	Legendre Multifractal Spectrum
$minTU$	Minimum transfer unit
$c(q)$	Moment factor
$\phi_0(t)$	Mother scaling function
$\psi_0(t)$	Mother Wavelet function

T_{obs}	Observation (aggregation) window
q	Order of the statistical moment
λ	Packet injection rate
$I(\omega_j)$	Periodogram function
$X(t)$	Random process (stationary)
$Y(t)$	Random process (non-stationary)
$\phi_{j,k}(t)$	Scaled and shifted scaling function
$\psi_{j,k}(t)$	Scaled and shifted Wavelet function
$a_X(j, k)$	Scaling coefficients
a	Scaling factor
$\tau(q)$	Scaling function
$a_X(n, k)$	Scaling parameters
$\mathbb{Z}$	Set of the integers
$\mathbb{R}$	Set of the real numbers
$f_X(\omega)$	Spectral density function
τ	Time lag
$approx_j$	Time series approximation (Wavelet)
$detail_j$	Time series details (Wavelet)
$d_X(j, k)$	Wavelet coefficients

CONTENTS

1	Introduction	16
1.1	Context	16
1.1.1	The ESL design flow	17
1.1.2	Communication structures	20
1.1.3	Platform abstraction	21
1.1.4	The performance analysis problem	23
1.2	Motivation	24
1.2.1	On-chip traffic modeling issues	24
1.2.2	High level description in LRD modeling	27
1.3	Objectives	28
1.4	Organization of the dissertation	28
2	Theoretical foundations	30
2.1	Transaction Level Modeling	30
2.2	Fractals and traffic modeling	34
2.2.1	Monofractal processes	35
2.2.2	Multifractal processes	36
2.3	Time series analysis methods	38
2.3.1	Autocorrelation function	38
2.3.2	Spectral density function	39
2.3.3	Abry-Veicht spectrum	42
2.3.4	Multifractal spectrum	43
2.4	Time series synthesis models	45

2.4.1	Auto-regressive models	46
2.4.2	Multifractal Wavelet Model	47
3	Related works	50
3.1	Varatkar et. al. proposition	50
3.2	Soteriou et. al. proposition	51
3.3	Scherrer et. al. proposition	52
3.4	Bogdan et. al. propositions	55
3.5	Sepulveda et. al. proposition	55
3.6	Pekkarinen et. al. proposition	56
3.7	Gonzalez et. al. proposition	57
4	Synthetic traffic generation methodology	59
4.1	Proposed methodology overview	59
4.2	Platform specification and simulation	60
4.2.1	SoCLib simulation platform	60
4.2.2	The TLM platform	62
4.3	Traffic capture	63
4.4	Statistical analysis	64
4.5	Model fitting	65
4.6	Traffic generator simulation	66
4.7	Synthetic traffic validation	68
5	Results	72
5.1	Test applications	72
5.2	TLM Traffic capture and analysis	74
5.2.1	Multifractal spectra estimation	74

5.2.2	Hurst exponent estimation	76
5.3	Synthetic traffic obtainment and validation	77
5.3.1	MWM and auto-regressive model fitting	77
5.3.2	Traffic generator implementation	77
5.3.3	Computation of the synthetic traffic and statistical analysis . . .	79
5.3.3.1	Plots visual inspection and accuracy analysis	79
5.3.3.2	Error computation and accuracy analysis	88
5.4	Speedup-accuracy trade-off	90
6	Conclusions	92
	Perspectives and future works	93
	Publications	94
	References	95
	Appendix A – Multifractal spectra of the CA traffic profiles	101
	Appendix B – Complimentary plots	103

1 INTRODUCTION

It is presented in this introductory chapter, the context and the motivation for the construction of a traffic generator that is able to produce realistic traffic and that can be attached to a high level model of an integrated system. Together with all this, the objectives of the work are also presented.

1.1 Context

New integration techniques in the fabrication technology for integrated circuits (ICs) have steadily progressed resulting in higher transistor densities, i.e., higher number of transistors per area. This density increase was possible due to the reduction of the dimensions of the CMOS (Complementary Metal Oxide Semiconductor) devices, which, nowadays, present channels as short as 10nm. This whole process has provoked a shift in the industry's perception of chip integration potentiality, since the amount of transistors in a single IC has grown from the order of millions to billions (DAMARAJU et al., **Proceedings**. New York: IEEE, 2012).

This technological progress has coined the term DSM (Deep Sub-Micron), which has been designated to modern devices with extremely small dimensions. The DSM paradigm formed a basis that has allowed designers to develop ICs with an internal structure composed of heterogeneous components such as (1) programmable units¹, (2) application specific hardware², (3) in-out interfaces, (4) memories, etc. Besides that, communication architectures are also designed to interconnect the previously mentioned components, which together operate as a system, known by the name MPSoC (Multiprocessor System-on-Chip). According to (PARISCHA; DUTT, 2008), MPSoCs consist of complex chip structures dedicated to meet the computational

¹They can be general purpose processors (GPP) or reconfigurable cores (RC).

²They can be application specific instruction processors (ASIP) or application specific integrated circuits (ASIC).

needs of an application; parallel and distributed computing concepts, such as MIMD (Multiple Instruction Multiple Data) (PARISCHA; DUTT, 2008), can be applied to these chips.

As a consequence of the ever increasing hardware capabilities, embedded software complexity has also increased, given that it must be distributed onto the different processing cores and effectively executed in real time, in the MPSoCs. Thus, the software starts to exert a significant influence in the time to market of embedded systems; in an average, forty percent of this time is used to prepare the software code, another forty percent is destined to prepare the hardware and the remaining twenty percent to integrate the software to the hardware and guarantee their correct execution (COPPOLA, 2008).

1.1.1 The ESL design flow

In MPSoCs design, in order to deal with the extensive functionality to be integrated on both hardware and software, IC designers have adopted the Electronic System Level (ESL) design methodology (BAILEY; MARTIN; PIZIALI, 2007). It corresponds to a top-down methodology starting from very abstract system representations and ending on the chip layout, as presented in Figure 1.1. This methodology has the objective of minimizing the design effort and complexity, allowing, at the same time, the designer to experiment with new ways of achieving the desired functionality through the combination of the architecture's components (hardware and software). In (GARZON, 2008), the authors cite three associated ESL design strategies being condensed in two in this text, as follows:

1. Utilization of models at a higher abstraction level than the Register Transfer Level (RTL) for the description of the design hardware description. An example of a more abstract hardware representation model is the Transaction Level Model (TLM), in which the communication between the modules are accomplished by function calls and are called transactions³. The use of more abstract hardware representations leads to more concise system descriptions and faster simulations. The upper abstraction layer is separated in representation domains,

³The communication can handle time variables, like the initial and final times of the transaction, and the data load of the transaction, known as payload

Figure 1.1: ESL methodology

Adapted from: (GONZALEZ, 2013)

which correspond to the first three steps of the ESL methodology in Figure 1.1; they are related to the following models:

- (a) The functional model, on the most abstract level, which is created from optimized algorithms describing the design target application. These models are defined in high level programming languages such as C and C++.
 - (b) The architectural model, which results from the exploration of potential architectures and a selection process grounded in the analysis of the best execution possibilities of the application's algorithms. These analyses are based in a hardware-software partitioning of the application and in the mapping of the application tasks in programmable processors or in hardware modules.
 - (c) The communication model, defined by a communication architecture, which meets the requirements and requisites of the architectural model.
2. Modular blocks reutilization within the system, based on the existence of pre-designed and pre-verified Intellectual Property (IP) cores, supplied by third

party developers⁴. The IP blocks are standardized to facilitate their reuse and they tend to be provided under the idea of platform based designs. A platform is defined as a pre-configured framework/workbench of a given number of soft or hard IP cores with a specific application domain or a specific final application (KEUTZER et al., 2000).

Following the methodology in Figure 1.1, after the design is satisfactorily represented in high abstraction fashion, it goes through a process of refinement of its internal blocks, that is, the design must be represented in the level of signals and pins (behavioral RTL level). Afterwards, the design is put through logical synthesis tools, which translates it to the level of logic gates by creating a netlist. The netlist then goes through the verification phase and a GDSII⁵ file can be generated for fabrication.

It is worth mentioning that in the steps of the ESL methodology a process known as optimization loop is applied. This process consists of a repetition of one or more of the methodology steps, applied whenever, in the design current state, the objective has not been met (a performance or area restriction, for example). It must be highlighted that the optimization loop is more acceptable in the initial stages of the design, given that repeating steps as the design follows to its final stages is costlier.

The strategy adopted for the communication model is of vital importance in the ESL methodology, since the association and interoperability of the modular blocks that compose the MPSoC depend on it. According to (MARCULESCU, 2010), the selection of the communication architecture will be a major research concern in the field in the years to come. The architecture is selected by designers with the intention of meeting some targets, which, according to (NICOPOULOS; NARAYANAN; DAS, 2009), can be summarized in (1) performance, (2) cost in terms of area, (3) power/energy efficiency, (4) reliability and (5) variability. There exists a wide set of communication architectures and two of them are typically implemented in integrated systems: buses, including its derivatives like crossbars and Networks-on-Chip (NoCs).

⁴Some known developers of commercial IP cores are Xilinx, Altera, ARM and Lattice.

⁵Standard IC layout format

1.1.2 Communication structures

A bus is a system composed of channels, buffers, switches and control units, used in several SoCs to facilitate data transport between IP blocks (PARISCHA; DUTT, 2008). A channel is made of wires and it forms the propagation medium of data, control, address and arbitration signals; the channel's bandwidth is determined by its amount of wires. In a bus-based SoC, two types of components can be found: the active components (also known as masters), that initiate the communication, and the passive ones (also known as slaves) that respond to communication demands.

A bus is a shared medium (Figure 1.2a and 1.2b) that works with a centralized or a distributed address decoder. The arbiter is a multiplexer that is in charge of selecting the masters for communication. If the slave cannot answer the selected master requests the data transfer gets delayed; data contention also occurs in case more than one master are requesting communication at the same time. Given the limitations shown by buses, the multipoint crossbars (Figure 1.2c) came by as an alternative for better performance; basically they are direct connections between all elements, although the implementation costs makes this configuration impractical for very large number of connected components. Multistage crossbars (Figure 1.2d) can be used instead, where all components are connected in a crossbar style but controlled by switches.

Scalability is one of the main limitations of buses, due to the presence of a greater number of processing cores in MPSoCs. In such a case, arbitration delays, higher energy consumption in direct relation with the size of the area footprint and difficulties in integrating varied frequency blocks become increasingly more evident (PEH; JERGER, 2009). Besides these problems, with transistor sizes below 22nm the risk of errors resulting from electrical noise, electromagnetic interference or crosstalk in the wires rises greatly (MICHELI; BENINI, 2006).

NoCs were conceived as a solution for reaching proper scalability levels and using efficient and organized interconnections in MPSoCs. Differently from buses, that use *circuit switching*, NoCs adopt *packet switching* transmissions, transferring data from an originating node⁶ to a destination node in a pipelined fashion.

The NoC connects MPSoC blocks through two connections that make a link or

⁶A node is any component (or set of components) connected to the network.

Figure 1.2: Bus classifications

(a) Shared bus with centralized arbitration

(b) Segmented bus

(c) Multipoint crossbar

(d) Multistage crossbar

All figures adapted from: (GONZALEZ, 2013)

bidirectional channel via a router component. Figure 1.3 shows a NoC with a 3×3 mesh topology with its IP blocks (processing elements with hierarchical L1 and L2 caches), besides the internal structure of a router, which has buffers, arbiters and a crossbar with a switching matrix. It can be seen in this example that the IP blocks are always connected to a router, which handles the traffic produced by this block, routing the packets to their destinations. Opposed to the low level decodification-based routing used by buses, in a NoC the routing of the packets is based on high level algorithms, resulting in a more efficient utilization of the interconnections, which reduces area usage, energy consumption and delays.

1.1.3 Platform abstraction

In conventional logical simulations of MPSoCs, RTL models are the ones that, in fact, define the hardware being represented and this is done with the aid of Hardware Description Languages (HDL) such as Verilog and VHDL. Despite the capability of RTL descriptions in offering high fidelity with respect to the corresponding real implementation, they affect the hardware-software development phase, making it to

Figure 1.3: Example of a Network-on-Chip (3×3 mesh topology)

Adapted from: (GONZALEZ, 2013)

take a large portion of the design time for the system as a whole.

In order to reduce the impact of the hardware-software development in the design time, platforms that allowed hardware-software co-verification has been adopted. In a co-verification platform the hardware is still described in RTL, but processor models with less description details are used; an example of this is the Instruction Set Simulator (ISS), that, despite being less detailed than a full RTL processor model, offers high precision for instruction execution per simulation cycle. In this type of platform, the hardware description is executed in a logical simulator and the application software is connected to the ISS (LEUPERS; TEMAM, 2010) through a symbolic debugger.

To improve hardware simulation times, cycle accurate (CA) models has been proposed, by describing the functionality through high level languages, as for example, SystemC⁷ (IEEE... , 2012). CA models guarantee the correct functioning (in terms of clock cycles) of the communication protocol of the description, allowing verification methods to be applied, but not the synthesis of the design. CA models may achieve their simulation speeds in one or two orders of magnitude above RTL ones.

Both the co-verification platform and CA modeling have some advantages when compared to pure RTL design, but they were not still adequate for the integration with the ESL methodology, due to the complexities in handling clock cycles. Higher

⁷SystemC is a library of C++ classes that offers hardware modeling oriented constructs.

level abstraction models were needed. A set of techniques, called Transaction-Level Modeling or TLM, has evolved to aid with this task (BLACK et al., 2009).

In TLM, each component in the system is modeled by a finite set of states and a series of concurrent behaviors, and all of this is encapsulated in an entity called a TLM module (GHENASSIA, 2005). The TLM modules communicate between themselves over structures called channels, that can, for example, represent buses or simple NoC routers. The channels connect communication ports, that originate (in master components) and return (at slave components) groups of actions and information called transactions.

TLM components (modules, channels and ports) can be integrated into a TLM platform, in order to model a specific MPSoC. This platform can be simulated, executing an embedded software in the processor model, that can be, for example, an ISS. The main difference between a TLM and a CA platform is expressed in the number of synchronization points between the system events of the code, which is much smaller for the first case.

1.1.4 The performance analysis problem

Many metric based methodologies were proposed to address the problem of communication architecture performance evaluation, aiming to help designers in (1) the selection of system communication topologies, (2) allocation of hardware cores and (3) mapping of software tasks in some of these cores. Some existing works are based on analytical approaches, extracting performance metrics under different formulations as, for example, queuing theory (QIAN et al., **Proceedings**. New York: IEEE, 2014), network calculus (LU; YAO; JIANG, **Proceedings**. New York: IEEE, 2014) or even statistical physics (BOGDAN; MARCULESCU, 2011); other works focus on simulations, extracting the performance metrics from cycle-accurate models of the network (JIANG et al., **Proceedings**. New York: IEEE, 2013) or from full system simulations (FSS)⁸ (SCHERRER; FRABOULET; RISSET, 2009). Despite the wide range of performance evaluation methodologies presented in previous researches, all of them has converged in focusing in the influence that the traffic produced by the system processing elements has on the behavior of the communication structures.

⁸A full system simulation relies on platforms with processors models that can execute a practically unchanged software code.

Figure 1.4: Functioning principle of a traffic generator

The limited amount of available MPSoCs communication structures (and integrated systems in general) performance evaluation tools has led researchers to modify existing simulation environments, or even design these environments from scratch. In both cases, the use of traffic generators is ubiquitous, since their complexity is much smaller when compared to a processor model, what may accelerate simulations (DALLY; TOWLES, 2004). As a consequence of this, the designers can explore the design space more easily. An efficient traffic generator must represent with fidelity the real traces produced by the block it will replace, correctly replicating its behavior and, above all, its interactions with the other blocks in the system (BAILEY; MARTIN; PIZIALI, 2007).

Figure 1.4 shows an illustration of the functioning principle of traffic generators. The objective of traffic generators is to substitute processing elements in simulations and, thus, make the compilation and execution of applications unnecessary.

1.2 Motivation

1.2.1 On-chip traffic modeling issues

As mentioned in the end of Section 1.1, the use of traffic generators is widespread when concerning on-chip communication structures performance evaluation. Seminal works in this subject has focused on the so called traffic replicators, that simply replicated recorded traffic data from full system simulations. This approach, however, showed itself to be impractical, since there were significant amounts of samples in the recorded traffic, what ultimately did not presented meaningful gains in simulation times.

Researchers have soon noticed that a more efficient approach was to portrait the traffic data from full system simulations as time series and produce synthetic ones from them based on formal statistical time series models. Initial works on synthetic on-chip traffic time series generation focused primarily on Poisson processes (WIKLUND; SATHE; LIU, **Proceedings**. New York: IEEE, 2004) or classic Markovian models (KIASARI et al., **Proceedings**. New York: IEEE, 2008). These models, however, aren't able to reproduce bursts⁹ in the time series at various levels of aggregation¹⁰ (LELAND et al., 1994; PAXSON; FLOYD, 1995), therefore, being said to capture only Short Range Dependence (SRD) characteristics.

One milestone in exposing the previous models limitations was the detection by (VARATKAR; MARCULESCU, 2004) of Long Range Dependence (LRD) in a NoC-based platform executing a video application (MPEG2 decoding). After that, self-similar time series started being utilized in the modeling of on-chip traffic in several researches (GONZALEZ et al., **Proceedings**. New York: IEEE, 2013; SCHERRER; FRABOULET; RISSET, 2009). According to (LIMA, 2008), self-similarity in time series is related to the preservation of burstiness characteristics through various levels of aggregation. Processes like auto-regressive fractionally integrated moving average (ARFIMA) (GRANGER; JOYEUX, 1980) and fractional Brownian motion (fBm) (BERAN et al., 2013) are able to capture LRD, since they were developed based in the fractal geometry theory.

An illustration of how self-similar models are better suited then other ones for representing LRD traffic series is presented in Figure 1.5, for a case of Ethernet traffic analysis presented in (LIMA, 2008). The two traffic samples shown at the left correspond to real Ethernet traffic (measured directly from an Internet router), while the samples in the middle correspond to a Poisson process and the two samples at the right correspond to a self-similar process. The samples on the upper row are more aggregated (windowing of 1 second) than the samples on the bottom row (windowing of 100 milliseconds)¹¹. It is visible that the bursts occurring in the real Ethernet data are not maintained by the Poisson process in the two different aggregation scales, while the self-similar process can reproduce them.

⁹Data that is transmitted intermittently (in bursts) rather than as a continuous stream

¹⁰The process of aggregation can be roughly viewed as the data series zooming process: zoom in - less aggregation; zoom out - more aggregation

¹¹The black strips shown in the time series plots were placed in the figure with the objective of illustrating the difference in the scales of the presented cases.

Figure 1.5: Failure of Poisson based traffic modeling

Adapted from: (LIMA, 2008)

Both the fBm and ARFIMA models are called monofractal models because they present fractal properties, which are characterized by a single coefficient, known as the Hurst exponent, which lies in the interval $[1/2, 1]$ for series with LRD characteristics. Since these models can be characterized by a single coefficient, they are called monofractal models and can create only stationary time series¹² (TAQUU; TEVEROVSKY; WILLINGER, 1997).

In later observations Bogdan et al. (BOGDAN; MARCULESCU, 2011) have shown that the traffic produced by the processing elements in a MPSoC are not always stationary; when the processing elements are operating near to their full capacity, packet injection rate (λ) is at its maximum, what may introduce non-stationary behaviors in the traffic. These behaviors are a consequence of an increase in the traffic complexity and can be attributed to the presence of more than one fractal coefficient/exponent, hence the traffic is said to become multifractal.

The presence of multiple fractal exponents can be assessed by analyzing the whole series and identifying all the exponents present in it; after this process, the identified exponents may be plotted as a distribution, known as the multifractal spectrum. Figure 1.6 illustrates some multifractal spectra, where the x-axis represents fractal dimensions and the y-axis represents the probability of occurrence of the present fractal dimensions; it can be noticed that for higher λ s the spectra becomes

¹²In time series analysis, a time series is said to be stationary process if the joint probability distribution of the samples does not change when shifted in time. Consequently, parameters such as mean and variance also do not change over time

Figure 1.6: Effect of the packet injection rate (λ) in the multifractality of a processing element traffic

Source: (BOGDAN; MARCULESCU, 2011)

wider, meaning that more fractal exponents are present. For these reasons, it is expected that multifractal traffic models may be more adequate to reproduce on-chip traffic.

1.2.2 High level description in LRD modeling

Another aspect related to traffic modeling is the description abstraction level of the full system simulator from which (real) traffic series are extracted or of the simulation platform with traffic generators; as mentioned before, co-verification platform, CA modeling and pure RTL design have shown large simulation times when compared to simulations of high level descriptions; for this reason, TLM was introduced to the ESL methodology (BLACK et al., 2009). High-level representations are desirable, making simulations faster, but leading to more inaccurate results.

High level platform representation was explored before in (SEPULVEDA et al., **Proceedings**. New York: IEEE, 2010), where LRD traffic generators based on fractional Gaussian noise (fGn) were used in fast TLM simulations; however no methodology for model parameter extraction were presented. This aspect were considered in (GONZALEZ et al., **Proceedings**. New York: IEEE, 2013), besides dealing with the question if the statistical properties of traffic series obtained from more accurate simulations (lower abstraction representations) were maintained in high-level simulations. Traffic from cycle-accurate and TLM full system simulations

were indicating that extracting model parameter from high level simulations could preserve LRD characteristics in synthetic generators.

1.3 Objectives

Given the motivations above, the general objective of this work is to develop a traffic capture and modeling methodology for building TLM-based traffic generators of multifractal traffic, through the use of systems described in TLM.

The specific objectives are:

- To establish a methodology for traffic capturing from high level descriptions and traffic characterization under the multifractal paradigm.
- To design and implement a transaction level traffic generator, capable of producing traffic with multifractal characteristics.
- To compare, statistically, the traffic obtained from the multifractal traffic generator with the original ones from the processors executing the selected applications.
- To compare, statistically, the traffic obtained from the multifractal traffic generator with the one obtained with monofractal models on the selected applications.
- To assess the simulation speedup with the multifractal traffic generator against the original full system simulations and, through this process, explore the speedup-accuracy trade-off.

1.4 Organization of the dissertation

The rest of the dissertation is composed as follows: firstly, an exposition, in Chapter 2, of the main theoretical foundations recommended for the understanding of this work, starting with the presentation of the main concepts behind Transaction Level Modeling (TLM), followed by the definitions of mono and multifractal processes; later, the main statistical methods for analysis and synthesis of mono and multifractal time series are presented.

In Chapter 3, seven research papers, which have been considered closely related works, are presented, disposed in a chronological and logical order. In Chapter 4, the methodology of the presented work is detailed, encompassing: (1) the specification of the TLM platform used for the extraction of the results, (2) the traffic capturing procedure, (3) the usage of the proposed statistical analysis, (4) the model fitting procedures and (5) the specification and simulation of the traffic generator. Next, in Chapter 5, the main results derived by the implementation of the methodology, along with related analysis and discussions, are exposed, while more detailed or complementary results (plots) are listed in Appendix B. Finally, in Chapter 6, the conclusions that have been drawn from the results are listed and discussed; some perspectives for future works are also summarized in this chapter.

2 THEORETICAL FOUNDATIONS

In this chapter the theory on which the present work is based is exposed. Firstly, the concepts behind Transaction Level Modeling are presented, taking the work (GHENASSIA, 2005) as a basis. Secondly, an exposition of the formalism behind traffic modeling, or more generally, one dimensional time series, is made, focusing in the definitions of monofractal and multifractal processes; the main references for this exposition are (STENICO, 2014; KRISHNA; GADRE; DESAI, 2003; ABRY; GONCALVES; VEHEL, 2009). Thirdly, the theory behind all of the statistical tools used throughout this work are shown. Finally, in the last section, the statistical models used in the present work for generating synthetic traffic series are detailed.

Its worth noting that the theory underlying multifractals, which is the main basis of the present work's methodology presented in Chapter 4, was included in this chapter despite deserving a chapter of its own. The multifractal theory was left together with the basic time series analysis and monofractal theory for didactical and organizational reasons, aiming to avoid unnecessary text repetitions that would probably occur if the mentioned subjects were separated into two chapters.

2.1 Transaction Level Modeling

In a digital electronic system, every single component may be considered a composition of a finite set of *states* and a series of *concurrent behaviors*. In Transaction Level Modeling (TLM) each of these components can be modeled as a *module*. The internal states of a component are represented by a set of variables defined within the scope of the corresponding TLM module, whereas the different behavior pieces of the component are modeled by a collection of *concurrent processes* or *threads*, which can be executed in parallel. These TLM modules can be gathered to form a TLM system, whose module communications are established through a specific TLM

structure, namely *channel* or *interconnect*. Depending on the accuracy level required by the corresponding simulation, a channel could be a simple router, an abstract bus model, a network-on-chip model, or some other, simpler or more complex, structures. Due to the aforementioned TLM system configurations it is possible to separate communication from computation in TLM modeling.

Modules and channels are bound to each other by means of communication *ports*. Once they are bound together, data can be exchanged between them to perform the expected system behavior.

The term *transaction* denotes a closed set of data being exchanged, which, in practice, includes events at both control and data signals. A *master* or *initiator* is a module or interface that initiate transactions in a system, while a *slave* or *target* is a module or interface that receive and serve transactional requests. Any consecutive transactions may have different sizes of data transfer. This variable size corresponds to the amount of data being exchanged between two occurrences of system synchronization. *System synchronization* is an explicit action between at least two modules that need to coordinate or manage some behavior distributed over them. Such co-operation of different modules is vital to assure the predictable system behavior. Since it is the only mechanism available for synchronizing different processes in a system, the explicit system synchronization is mandatory to ensure a proper deterministic SoC TLM behavior.

TLM can be attained through an appropriate electronic system level (ESL) modeling approach, being a high-level programming language the right candidate to do this task. It is capable of developing not only a plain software program, but also of modeling electronic hardware at the conceptual level without describing the real implementation. SystemC (IEEE. . . , 2012) has been shown to be a very adequate language for this purpose.

Any SoC component can be modeled as a module in TLM, with the primary modeling effort lying on the internal computation of the given hardware block at the functional or behavioral level. None of the micro-architectural implementation details should be included, i.e. neither internal pipelines nor structures are modeled. On the other hand, input and output interfaces of the block, as well as its synchronization, are to be modeled.

A complete SoC TLM platform is constructed by instantiating and binding different

modules (e.g., memories, I/O components, processing elements) together through channels. In this way, once the platform is integrated and a proper hardware-software partitioning is completed, SoC simulation can be performed by executing an embedded software, which can be inserted in the system either through native or cross compilation procedure.

The modeling accuracy of a given modeling approach indicates the precision or correctness of the model in replicating the intended behavior and activities of a system-under-design. For any modeling strategy in the SoC design flow, there are two decisive factors to determine the degree of modeling accuracy (GHENASSIA, 2005):

1. Granularity of the communication data: this criterion reflects the fineness or coarseness of the data carried by the communication structure of a model. The data granularity can generally be categorized into three levels, i.e. application packet, bus packet, and bus size, in the order of increasing accurateness. The functioning of a video decoding IP block helps to illustrate the idea of data granularity. If the IP has a frame-based algorithm, a coarse granularity at application packet could be modeled as a frame-by-frame transfer. A finer granularity at bus packet level can be represented by a line or column-based transfer, or a macro-block transfer consisting both lines and columns. The finest grain at bus size level will be the pixel-based transfer of the video.
2. Timing accuracy: timing accuracy determines the fidelity of a model to the intended timing behavior. It can be conceptually perceived as a scale of two extremes, i.e. untimed level and cycle-accurate level. Moving from the untimed end towards the cycle-accurate end will increase the timing accuracy of a model. Any level falling in between the two ends is considered as approximately timed level.

Just as any other modeling strategies in the SoC design flow do, the TLM approach naturally revolves around the two factors above to decide its modeling accuracy. Guided by these criteria, two fundamental classes of TLM can be defined: untimed TLM and timed TLM.

The untimed TLM is an architectural model targeted specifically at early functional software development and functional verification where timing annotations are not

Figure 2.1: Modeling accuracy of various approaches

Source: (GHENASSIA, 2005)

mandatory conditions. A high simulation speed is the objective of this model. Since the untimed TLM primarily aids designers in earlier phases of the design, i.e., when there is still a faint notion of hardware, it is also called *programmer's view* (PV).

On the other hand, the timed TLM is a micro-architectural model containing essential time annotations for behavioral and communication specifications. It is relatively a less abstract model located at a lower abstraction level in the SoC design flow. The focus of timed TLM is the simulation accuracy required by real-time embedded software development and architecture analysis. Hence, the timed TLM is also known as *programmer's view plus timing* (PVT).

Figure 2.1 gives a glimpse at the modeling accuracy of the untimed and timed TLM with respect to other conventional models in the SoC design flow, including register transfer level (RTL), bus cycle accurate (BCA), and cycle accurate (CA) models. More details on these non-TLM layers can be found in (GHENASSIA, 2005).

In addition to the untimed and timed TLM classes, recently, another class has been conceived with the objective of addressing the modeling of MPSoCs, the TLM with distributed timing, or TLM-DT. It is based on the possibility of parallel simulations on SMP (Symmetric Multi-processed System) workstations, which is achieved with the association of the principles of Parallel Discrete Event Simulation (PDES) with SystemC (MELLO et al., **Proceedings**. New York: IEEE, 2010). This class of TLM is widely used in the virtual prototyping platform SoCLib (SOCLIB, 2016d).

Figure 2.2: Five iterations of the Sierpinsky triangle

Source: (STENICO, 2014)

2.2 Fractals and traffic modeling

The expression *fractus* originates from Latin and means fraction or broken. The word *fractal* was used by Benoit Mandelbrot to describe objects that were too irregular to be studied by the optics of the traditional euclidean geometry (MANDELBROT, 1983). In the deterministic sense, a geometric shape is fractal when it's observed appearance is preserved in all scales, whether in space, either in time. Geometric figures obtained by recursive processes are typical deterministic fractal sets, such as the Sierpinsky triangle, illustrated in Figure 2.2.

The property of the observed appearance preservation in all scales is called self-similarity and it is the main characteristic of a fractal. The notion of self-similarity can be generalized to surpass the deterministic sense, i.e., it can also be regarded statistically. In this way, what is preserved in different scales are not deterministic properties (observed appearance, for example), but statistical properties, leading to better representations in fields like time series analysis, with applications in hydrology, econometrics and data traffic modeling (FALCONER, 2003). An example of a statistically self-similar object, a time series, for example, is shown in Figure 2.3.

Figure 2.3: Example of a statistically self-similar object

Source: (BOURKE, 2003)

2.2.1 Monofractal processes

In the context of analysis and dimensioning of computer networks and packet-based on-chip communication structures, it has turned out to be of great importance to correctly model the data traffic. Self-similarity is an important aspect to be included in the models since it can have adverse effects on system's performance, affecting, for example, packet loss rates and communication delays (KRISHNA; GADRE; DESAI, 2003). Mathematically, a self-similar stochastic process can be defined as follows:

Definition 1. (*Self-similar stochastic process*) A random process $Y(t)$, $t \in \mathbb{R}$, is said to be self-similar with self-similarity parameter $H \in (0, 1)$ if for any positive scaling factor $a > 0$ and $\forall t \geq 0$, the processes $Y(t)$ and $a^{-H}Y(at)$ are identically distributed, i.e.,

$$Y(t) \stackrel{d}{=} a^{-H}Y(at), \quad (2.1)$$

where H is the Hurst exponent and the symbol $\stackrel{d}{=}$ represents equality in distribution.

The process $Y(t)$, according to Definition 1, is always non-stationary¹ and is also known as *fractional Brownian motion* (fBm). To the fBm, $X(t)$ is an associated stationary self-similar stochastic process known as *fractional Gaussian noise*, which

¹In mathematics and statistics, a stationary process is a stochastic process whose joint probability distribution does not change when shifted in time. Consequently, parameters such as expected value and variance also do not change over time.

obeys the following relation:

$$X(t) = Y(t_0 + t) - Y(t_0) \stackrel{d}{=} a^{-H}[Y(t_0 + at) - Y(t_0)], \quad (2.2)$$

For fBm and fGn in discrete time, their corresponding sampling processes $Y[k]$ and $X[k]$ are considered; the first one, $Y[k]$, is used to represent the accumulated traffic volume up to the sample k (therefore, the *accumulation process*), while $X[k]$ is used to represent the amount of traffic between samples k and $k + 1$ (therefore, the *increment process*).

Self-similarity can also be defined for discrete time processes as follows:

Definition 2. (*Discrete time self-similar stochastic process*) A process $X[k]$, $k \in \mathbb{Z}$, is said to be self-similar with self-similarity parameter $H \in (0, 1)$ if for any positive aggregation factor m and $\forall k \geq 0$, the processes $X[k]$ and $m^{1-H}X^m[k]$ are identically distributed, i.e.,

$$X[k] \stackrel{d}{=} m^{1-H}X^m[k] \quad (2.3)$$

where

$$X^m[k] = \frac{1}{m} \sum_{i=m(t-1)+1}^{mt} X[i] \quad (2.4)$$

is the aggregated process of $X[k]$, i.e., the sample average of $X[k]$ in non-overlapping blocks of size m .

2.2.2 Multifractal processes

The geometric complexity of a fractal can be described, in a global manner, by its degree of self-similarity, which is intrinsically related to the concept of dimension² (ABRY; GONCALVES; VEHEL, 2009). For natural objects, such as traffic series, self-similarity is regarded in a statistical sense, i.e., what is said to be preserved in different observation scales are statistical distributions, which are characterized by

²In mathematics, dimension is a parameter capable of providing a statistical index of complexity by relating the amount a pattern changes with the scale at which it is measured. There are several possible definitions for dimension, but one of the most intuitive is the box counting, defined as follows: let us imagine an object (e.g., a fractal) lying on an evenly spaced grid and we wish to count how many boxes are required to cover the object; the box-counting dimension is calculated by seeing how the number of boxes change as the grid is made finer. Suppose that $N(\epsilon)$ is the number of boxes of side length ϵ required to cover the object S . Then the box-counting dimension is given by:

$$\dim_{\text{box}}(S) := \lim_{\epsilon \rightarrow 0} \frac{\log N(\epsilon)}{\log(1/\epsilon)}$$

Figure 2.4: Scaling function ($\tau(q)$) of a multifractal process

Source: (STENICO, 2014)

its moments (FREEDMAN; PISANI; PURVES, 2007). The moment of order q of a stochastic process $Y(t)$ is the expected value of $Y(t)^q$ ($q \in \mathbb{R}$), which follows the relation³

$$E(Y(t)^q) = c(q)t^{\tau(q)+1} \quad (2.5)$$

where $E()$ is the expected value operator, $\tau(q)$ is the scaling function and $c(q)$ is the moment factor.

For a monofractal object, the scaling function $\tau(q)$ should be linear with q , what would indicate that the statistical moments of the object have the same scaling behavior. However, this is not always true for natural objects, since measurements on these objects show, as exemplified in Figure 2.4, that the scaling function is not linear with q , i.e., some moments have different scaling behaviors. This difference can be associated to the presence of multiple fractal dimensions in the same object (KRISHNA; GADRE; DESAI, 2003), that is, the object is a *multifractal*. Therefore, the following definition may be given:

Definition 3. (*Multifractal process*) A stochastic process $Y(t)$ is said to be multifractal if it has stationary increments and if its scaling function, $\tau(q)$, is non-linear with q .

Most of the times, this non-linearity is small, that is, the value of $d\tau(q)/dq$ (normally in the interval $[1/2, 2]$) shows small variations. When visual inspection is made, the

³The statistical moment of order one of a time series is its expected value. The moment of order two would be the variance.

graph of $\tau(q)$ could be perceived as almost linear, what would lead to a erroneous conclusion that a monofractal is being handled. Due to this, a more accurate analysis is needed, which is achieved by the multifractal spectrum⁴.

2.3 Time series analysis methods

In this section, the main time series analysis methods used in the literature are presented. These methods can be applied to time series with both mono and multifractal properties, giving relevant information in different domains. In this work, the time, spectral, Wavelet and dimensions domains were considered, corresponding to, respectively, the autocorrelation function (ACF), spectral density function, Abry-Veicht spectrum and multifractal spectrum methods. The theory behind these methods, extracted mainly from (SHUMWAY; STOFFER, 2010; STENICO, 2014), are exposed in the following subsections.

2.3.1 Autocorrelation function

The autocorrelation function is the most basic way to assess the dependence between two different points in time (k_1 and k_2) of a same time series $X[k]$, and it does so by relying on the autocovariance function $\gamma_X(k_1, k_2)$, defined as:

$$\gamma_X(k_1, k_2) = E\{X[k_1]X[k_2]\} - E\{X[k_1]\}E\{X[k_2]\} \quad (2.6)$$

The autocovariance measures the linear dependence between two points on the same series observed at different times, in such a way that if $\gamma_X(k_1, k_2) = 0$, $X[k_1]$ and $X[k_2]$ are not linearly related. Another important property is that if $X[k]$ is stationary, the autocovariance does not depend on the specific values of k_1 and k_2 , but only on the lag or time shift; therefore, $\gamma_X(k_1, k_2)$ can be rewritten as $\gamma_X(\tau)$, with $\tau = |k_1 - k_2|$.

Based on Equation 2.6, the autocorrelation function can be defined as:

Definition 4. (*Autocorrelation function*) The autocorrelation function (ACF) of a time

⁴The concept of multifractal spectrum will be explored with more details in Section 2.3.4

series $X[k]$ is defined as

$$\rho_X(k_1, k_2) = \frac{\gamma_X(k_1, k_2)}{\gamma_X(k_1, k_1)\gamma_X(k_2, k_2)} \quad (2.7)$$

i.e., the autocorrelation function is the normalization of the autocovariance function by the product of the variances of the time series at time k_1 and k_2 .

Given the relation between Equations 2.7 and 2.6, the ACF, for a stationary time series, can also be rewritten as a function of the lag, $\rho_X(k_1, k_2) = \rho_X(\tau)$.

As seen in Subsection 2.2.1, the Hurst exponent measures the degree of self-similarity in the process (or time series). Specifically, H expresses the rate of decay of the series' autocorrelation function. As shown in (BERAN et al., 2013), the asymptotic behavior of $\rho_X(\tau)$ should be of the form:

$$\rho_X(\tau) \cong c\tau^{-\beta}, \quad \tau \rightarrow \infty \quad (2.8)$$

where c is a constant and $\beta = 2 - 2H$ ($0 < \beta < 1$). In this way, if $1/2 < H < 1$, one can show that:

$$\sum_{\tau=0}^{\infty} \rho_X(\tau) = \infty, \quad (2.9)$$

i.e., the autocorrelation function decays slowly (hyperbolically), and, therefore, its summation diverges. When $\rho_X(\tau)$ has an hyperbolic decay, then, the process $X[k]$ is said to have *Long Range Dependence* (LRD). Otherwise, when $H \leq 1/2$, $\rho_X(\tau)$ presents an exponential (faster) decay and the process is said to have *Short Range Dependence* (SRD). These conditions can be seen in the example shown on Figure 2.5.

2.3.2 Spectral density function

The spectral density function is based on the concept that any stationary time series may be approximated by a random superposition of sines and cosines oscillating at various frequencies. For stochastic processes, such as the time series of interest in this work, this frequency representation is given as the Fourier transform of the autocovariance function, therefore, the autocovariance function and the spectral density function *contain the same information*. The autocovariance function expresses information in terms of lags, whereas the spectral density expresses the

Figure 2.5: Behavior of the autocorrelation function for different values of the Hurst exponent

Adapted from: (STENICO, 2014)

same information in terms of frequencies. Some problems are easier to work with when considering lagged information, handled in the time domain, while other ones are better dealt considering periodic information, handled in the spectral domain.

Definition 5. (Spectral density function) The spectral density function f_X of a time series $X[k]$ is the frequency representation of its the autocovariance function γ_X , therefore, they form a Fourier pair, i.e.,

$$f_X(\omega) = \sum_{\tau=-\infty}^{\infty} \gamma_X(\tau) e^{-2\pi i \omega \tau} \quad -1/2 \leq \omega \leq 1/2 \quad (2.10)$$

Given that γ_X is non-negative $f_X(\omega) \geq 0, \forall \omega$. Also, it follows immediately from Equation 2.10 that $f_X(\omega) = f_X(-\omega)$ and $f_X(\omega) = f_X(1 - \omega)$, verifying the spectral density is an even function of period one. Because of the evenness, $f_X(\omega)$ is only plotted for $\omega \geq 0$.

The spectral density function f_X is a theoretical function, therefore it is not possible to obtain it directly for most time series (unless for the ones that can be expressed by closed formulas). Because of this, the spectral density function must be estimated and the most common method to do so is the periodogram method, defined as:

Definition 6. (Periodogram) Given the time series $X[k]$, composed of n samples,

Figure 2.6: Example of (smoothed) periodograms

based on its Discrete Fourier Transform (DFT)

$$d(\omega_j) = n^{1/2} \sum_{t=1}^n X[k] e^{-2\pi i \omega_j k} \quad (2.11)$$

for $j = 0, 1, \dots, n-1$, the periodogram $I(\omega_j)$ of a time series $X[k]$ can be computed using the following expression:

$$I(\omega_j) = |d(\omega_j)|^2 = \sum_{h=-(n-1)}^{n-1} \gamma_X(h) e^{-2\pi i \omega_j k} \quad (2.12)$$

Given that its definition relies on the autocovariance function, the periodogram is also impacted by the presence of self-similarity in the time series. As shown in Equation 2.9, the summation of the autocorrelation function (directly related to the autocovariance function) diverges when the series presents self-similarity, generating a pole near frequency zero in the spectral density function. Therefore, it can be shown that the spectral density function has the following asymptotic behavior:

$$f_X(\omega) \cong c_f |\omega|^{1-2H} \quad (2.13)$$

where c_f is a constant and H is the Hurst exponent. This behavior can be noticed in the examples of periodograms shown in Figure 2.6; in Figure 2.6a, the value of the periodogram near frequency zero is in the same order of magnitude of the other values, while in Figure 2.6b the values near frequency zero are in a higher order of magnitude.

2.3.3 Abry-Veicht spectrum

The Abry-Veicht spectrum is based on Multi Resolution Analysis (MRA), which, on its turn, is based on Wavelets. The key idea behind MRA is to analyze the loss of information in subsequent approximations of a time series⁵ $X[i]$. These approximations are given by:

$$approx_j = \sum_k a_X(j, k) \phi_{j,k}(t) \quad (2.14)$$

where $\phi_{j,k}(t)$ is the scaled and shifted scaling function, defined as

$$\phi_{j,k}(t) = 2^{-j/2} \phi_0(2^{-j}t - k), k \in \mathbb{Z} \quad (2.15)$$

and ϕ_0 is the mother scaling function⁶.

Since $approx_j$ is a coarser approximation of $X[i]$ than $approx_{j-1}$, the loss of information when going from one approximation to the next is called the detail and is given by:

$$detail_j(t) = \sum_k d_X(j, k) \psi_{j,k}(t) \quad (2.16)$$

where $\psi_{j,k}(t)$ is the scaled and shifted Wavelet function, defined as

$$\psi_{j,k}(t) = 2^{-j/2} \psi_0(2^{-j}t - k), k \in \mathbb{Z} \quad (2.17)$$

and ψ_0 is the mother Wavelet⁷.

The Wavelet coefficients $d_X(j, k)$ measure the amount of energy in the analyzed time series about the time instant $2^j k$ and frequency $2^{-j} f_0$, where f_0 is an arbitrary reference frequency selected by the choice of ψ_0 . Therefore, a time average of $|d_X(j, k)|^2$ at a given scale j can be an estimator for the spectral density function (ABRY; GONCALVES; FLANDRIN, **Proceedings**. New York: IEEE, 1993), that is

$$\hat{\gamma}_X(2^{-j} f_0) = \frac{1}{n_j} \sum_k |d_X(j, k)|^2 \quad (2.18)$$

⁵The index for discrete time series is changed from k to i in this subsection and in Subsection 2.4.2 in order to avoid misinterpretations, since one of the indexes of the scaling and Wavelet functions are commonly written as k in the literature.

⁶ $a_X(j, k)$ are called the scaling coefficients

⁷ $d_X(j, k)$ are called the Wavelet coefficients

The right hand side of Equation 2.18 is known as the Abry-Veicht spectrum, leading to the following definition

Definition 7. (Abry-Veicht spectrum) *The Abry-Veicht spectrum, $S(j)$, computed for every possible j , is given by*

$$S(j) = \frac{1}{n_j} \sum_k |d_X(j, k)|^2 \quad (2.19)$$

where n_j is the available number of wavelet coefficients at octave j . Essentially $n_j = 2^{-j}n$ where n is the length of the time series.

Given its relation to the spectral density function, the Abry-Veicht spectrum is also affected by the Hurst exponent of the time series under analysis, therefore, this exponent can be estimated from $S(j)$. Through some non-trivial manipulations, presented in (ABRY; GONCALVES; FLANDRIN, **Proceedings**. New York: IEEE, 1993), it was shown that

$$\log_2(S(j)) = (2\hat{H}_{AV} - 1)j + c \quad (2.20)$$

where $\hat{H}_{AV}$ is the Abry-Veicht Hurst exponent estimator.

$\hat{H}_{AV}$ can be calculated by doing a linear regression in the plot of the Abry-Veicht spectrum and taking the value of the regression's slope. Figure 2.7 shows two examples of Abry-Veicht spectra; in Figure 2.7a, one can see that the slope of the regression is negative, therefore $2\hat{H}_{AV} - 1 < 0 \Rightarrow \hat{H}_{AV} < 1/2$, i.e., the series presents SRD, while in Figure 2.7b, one can see that the slope of the regression is positive, therefore $2\hat{H}_{AV} - 1 > 0 \Rightarrow \hat{H}_{AV} > 1/2$, i.e., the series presents LRD.

It has been shown in several works, ranging from Internet to on-chip traffic analysis, that $\hat{H}_{AV}$ is an extremely accurate estimator for the Hurst exponent, given that it surpasses some significant biases presented by other estimators (BERAN et al., 2013). Due to this fact, $\hat{H}_{AV}$ was chosen to compose this work's methodology, as will be presented in Section 4.4.

2.3.4 Multifractal spectrum

As mentioned in Subsection 2.2.2, for multifractal processes, the scaling function $\tau(q)$ is non-linear in relation to the statistical order q and the value of its derivative,

Figure 2.7: Example of Abry-Veicht spectra

$d\tau(q)/dq$, shows small variations, normally lying in the interval $[1/2, 2]$. Due to these small variations, $\tau(q)$ can look as if it is almost linear, creating the necessity of representing it in a way that is able to make evident its non-linearity.

Firstly, $\tau(q)$ must be correctly estimated, what can be done in an unbiased fashion through a generalization of the Abry-Veicht spectrum (ABRY et al., 2002). This generalization relies in taking time averages of the Wavelet coefficients, $d_X(j, k)$, under a range of exponents (represented by q), instead of only a second order exponent, as it was done in Subsection 2.3.3. The output of this process is a function, $H(q)$, known as the *generalized Hurst exponent*, which is related to the scaling function in the following manner

$$\tau(q) = qH(q) - 1 \quad (2.21)$$

A way of evidencing the non-linearity of $\tau(q)$ is to compute its Legendre transform. It can be shown that the output of this computation is one of the simplest and most robust ways of estimating a series' multifractal spectrum (KRISHNA; GADRE; DESAI, 2003):

Definition 8. (*Legendre multifractal spectrum*) Let $\tau(q)$ be the scaling function of the time series $X(t)$. The Legendre spectrum of $X(t)$ is given by:

$$f_L(\xi) = \tau^*(\xi), \quad (2.22)$$

where $\tau^*(\xi)$ is the Legendre transform of the scaling function $\tau(q)$, given by

$$\tau^*(\xi) = \inf_q (q\xi - \tau(q)).$$

Figure 2.8: Example of multifractal spectra

The Legendre transform takes a function f to f^* , in a way that the first derivative of f is the inverse of the first derivative of f^* . If f is a linear function, then f^* degenerates into a single point and if f is non-linear f^* becomes an inverse parabola. Therefore, although no relevant parameters can be directly extracted from the multifractal spectrum, it is an important visualization tool to identify the presence of multifractal characteristics in a time series. An illustration of the difference between the multifractal spectrum of a mono and a multifractal time series is shown in Figure 2.8.

2.4 Time series synthesis models

In this section, time series synthesis models adopted in this work are presented; they work by fitting specific parameters to the time series being modeled, then, the parameters are simulated according to the model mechanics and synthetic time series are obtained. It is worth noting that the precision of these synthetic time series is tied to the qualities and objectives of the model, as will be made clear in the following subsections. The theory for this section was extracted from (SHUMWAY; STOFFER, 2010; RIEDI et al., 1999).

2.4.1 Auto-regressive models

The pure Auto-Regressive (AR) model can be seen as a Finite Impulse Response (FIR) filtering of a time series $X[k]$ (DAS; PAN, 2011) and is defined as

$$X[k] = w[k] + \sum_{j=1}^p \phi_j X[k-j] \quad (2.23)$$

where $w[k]$ is a Gaussian white noise, p is the order of the filter and the ϕ_j 's are the auto-regressive parameters; since the order of the filter is p , this model is denoted by $AR(p)$.

Related to the AR model, the Moving Average (MA) model can be seen as an Infinite Impulse Response (IIR) filtering of a Gaussian white noise $w[k]$ (DAS; PAN, 2011) and is defined as

$$X[k] = w[k] + \sum_{j=1}^q \theta_j w[k-j] \quad (2.24)$$

where q is the order of the filter and the θ_j 's are the moving average parameters; since the order of the filter is q , this model is denoted by $MA(q)$.

Equations 2.23 and 2.24 can be rewritten in terms of the backshift operator⁸ B as

$$\begin{cases} (1 - \sum_{j=1}^p \phi_j B^j)X[k] = w[k] \Rightarrow \phi(B)X[k] = w[k] \\ X[k] = (1 + \sum_{j=1}^q \theta_j B^j)w[k] \Rightarrow X[k] = \theta(B)w[k] \end{cases} \quad (2.25)$$

where $\phi(B)$ and $\theta(B)$ are the Auto-Regressive and Moving Average operators, respectively.

From all this, the following definition can be given:

Definition 9. (*Auto-Regressive Moving Average model*) The Auto-Regressive Moving Average model is an association of an $AR(p)$ model with a $MA(q)$ model, therefore denoted $ARMA(p, q)$, and is given by

$$\phi(B)X[k] = \theta(B)w[k] \quad (2.26)$$

The orders p and q of the ARMA model are determined by the shape of the

⁸The backshift operator is defined as $BX[k] = X[k-1] \Rightarrow B^n X[k] = X[k-n]$.

modeled series' autocorrelation function. Since, for practical reasons, p and q must be finite numbers, the ARMA model is only useful to model series whose autocorrelation function is "finite", or, more precisely, have an exponential decay. Because of this, ARMA models are suited to series without self-similarity, that is, series with SRD. However, the ARMA model can be extended to a more general one to handle LRD; this is done by inserting a fractional integration (FI) operator in the ARMA model, resulting in the ARFIMA model. The degree of this fractional integration is a fractional number d , with $|d| < 0.5$. Due to this, the ARFIMA model has one additional parameter, it is written as ARFIMA(p, d, q). All of this leads to the following definition.

Definition 10. *A time series $X[k]$ is ARFIMA(p, d, q) if it is stationary and obeys the relation*

$$\phi(B)(1 - B)^d X[k] = \theta(B)w[k] \quad (2.27)$$

where p and q are the orders (degree) of the auto-regressive ($\phi(B)$) and moving average ($\theta(B)$) operators, respectively, and d is the order of the fractional integration operator.

The fractional integration operator can be expressed as

$$(1 - B)^d = \sum_{j=0}^{\infty} \pi_j B^j \quad (2.28)$$

where

$$\pi_j = \frac{\Gamma(j - d)}{\Gamma(j + 1)\Gamma(-d)} \quad (2.29)$$

with $\Gamma(z) = \int_0^{\infty} x^{z-1} e^{-x} dx$ being the gamma function.

The d parameter is also estimated from the series' autocorrelation function.

2.4.2 Multifractal Wavelet Model

Although the ARFIMA model is a powerful and tractable model, self-similarity is treated in a strict form, what prevent the model to adequately characterize many types of signals. For instance, time series may possess significant LRD, but still display short term correlations and scaling behaviors inconsistent with strict self-similarity; moreover, in many time series, the scaling behavior of moments, as the series is aggregated, is a nonlinear function of the moment order; also many time series are

inherently positive and, hence, non-Gaussian.

Time series with the properties listed above fall naturally into the class of multifractal processes. Multifractal models are positive measures or distributions possessing self-similarity but non-homogeneous scaling. The goal of the Multifractal Wavelet Model is to propose a multifractal extension to traditional monofractal models suitable for analyzing, characterizing, and synthesizing positive processes with LRD.

The main concept of the MWM, which is completely based on Wavelet analysis, comes from the fact that one can derive a discrete time stochastic process (time series) $X^{(n)}[i]$ from a continuous time stochastic process $X(t)$, considering that the first one is an approximation of the last with resolution 2^{-n} . If the Haar scaling and Wavelet functions⁹ are used, $X^{(n)}[i]$ is related to the scaling parameters $a_X(n, k)$ (the same ones shown in Equations 2.14) according to the relation

$$X^{(n)}[i] = \int_{i2^{-n}}^{(i+1)2^{-n}} X(t)dt = 2^{-n/2}a_X(n, k) \quad (2.30)$$

for $i = 0, 1, \dots, 2^n - 1$.

From the relation presented in the equation above, one can see that a synthetic time series $\tilde{X}[i]$ can be derived from the generation of synthetic scaling coefficients $a_{\tilde{X}}(n, k)$. These synthetic coefficients can be obtained by recursive sums and differences of its own scaling and Wavelet coefficients, $a_{\tilde{X}}(j, k)$ and $d_{\tilde{X}}(j, k)$, as in the following equations:

$$\begin{cases} a_{\tilde{X}}(j, 2k) = 2^{-1/2}[a_{\tilde{X}}(j-1, k) + d_{\tilde{X}}(j-1, k)] \\ a_{\tilde{X}}(j, 2k+1) = 2^{-1/2}[a_{\tilde{X}}(j-1, k) - d_{\tilde{X}}(j-1, k)] \end{cases} \quad (2.31)$$

In order to the resulting synthetic series to have no negative values, the Wavelet coefficients $d_{\tilde{X}}(j, k)$ must be always smaller than the scaling coefficients $a_{\tilde{X}}(j, k)$; to guarantee this condition, one can tie multiplicative factors to the Wavelet coefficients, that is,

$$d_{\tilde{X}}(j, k) = a_{\tilde{X}}(j, k)A(j, k) \quad (2.32)$$

where $A(j, k)$ is a variable whose value lies in the interval $[-1, 1]$. Besides that, other conditions are imposed to the multipliers $A(j, k)$: they must be independent

⁹The Haar scaling and Wavelet functions are orthonormal functions and are the most basic Wavelet basis

and identically distributed (IID) within each scale, they must be independent of $a_X(j, k)$ and symmetrical around zero. These conditions are met by attributing a Beta distribution to the multipliers¹⁰.

If the conditions of the multipliers $A(j, k)$ are met and the Haar scaling and Wavelet functions are adopted, the relations in Equation 2.31 can be rewritten in order to completely eliminate the need for the Wavelet coefficients altogether, allowing the following relations to be established:

$$\begin{cases} a_{\tilde{X}}(j+1, 2k) = \left(\frac{1 + A(j, k)}{\sqrt{2}} \right) a_{\tilde{X}}(j, k) \\ a_{\tilde{X}}(j+1, 2k+1) = \left(\frac{1 - A(j, k)}{\sqrt{2}} \right) a_{\tilde{X}}(j, k) \end{cases} \quad (2.33)$$

Therefore, to generate a synthetic time series using the MWM, the parameters that must be extracted from the series under modeling are the parameters of the Beta distribution for the multipliers for every scale¹¹ and the value of $a_X(0, 0)$. The first parameter is extract from the correlation structure of the series and the later one is extracted from the discrete Wavelet transform of the series.

¹⁰The beta distribution is a family of continuous probability distributions defined on the interval $[0, 1]$ parametrized by two positive shape parameters, denoted by α and β , that appear as exponents of the random variable and control the shape of the distribution.

¹¹For a series with 2^N samples, N scales are going to be considered, therefore N different Beta distributions will have to be fit.

3 RELATED WORKS

In this chapter, the most relevant works related to the present dissertation are going to be exposed, showing the methods and shortcomings of each of them. The works are organized in the following sessions in a chronological order and they were selected for this exposition due to their intimate relation with simulation of SoCs and MPSoCs with LRD traffic generators. Besides, the related works also disposed in a logical order, given that more rigorous methods for LRD analysis and TLM modeling are introduced approximately following the chronological order. The exception for this rule, is the work by Bogdan et. al. presented in Section 3.4 which was the first to introduce multifractal analysis in the context of on-chip traffic modeling.

3.1 Varatkar et. al. proposition

In the context of performance analysis of on-chip communication structures, the work of Varatkar et al. (VARATKAR; MARCULESCU, 2004) was the first to adopt self-similar stochastic processes for the modeling and analysis of the data traffic. The authors proposed a methodology to generate synthetic traffic traces based on real ones, captured through the simulation of a MPEG2 video decoder implemented a NoC-based MPSoC. The simulation was performed by using the Stateflow tool from the MATLAB software (MATLAB, 2016b) and the traffic measurements were drawn at the input buffers of two processing modules executing two different tasks.

The synthetic traffic was generated with the fractional Gaussian noise (fGn) model, which depends on a single parameter, the Hurst exponent, as shown in Equation 2.2. In the work, the synthetic traffic was used to speed up performance estimation simulations, focusing in the impacts that the self-similarity present in the traffic can have in the formation of queues in the buffers. With such information, the authors analyzed how the Hurst exponent could be used to improve input buffer sizing in

modules.

Nevertheless, the Hurst exponent estimation methods used by the authors are related to the autocorrelation and periodogram estimators shown in Subsections 2.3.1 and 2.3.2, and they have been shown to be obsolete, with inaccurate results, as mentioned in Subsection 2.3.3. Another limiting factor of Varatkar's work is the Stateflow tool, which uses the StateCharts formalism for the MPSoC description. Although StateCharts semantics was later integrated with TLM (FINDENIG et al., **Proceedings**. New York: IEEE, 2010), the authors have not used this approach, opting for a chart based description and not a high level language description, which is not in conformity with the ESL methodology.

3.2 Soteriou et. al. proposition

In (SOTERIOU; WANG; PEH, **Proceedings**. New York: IEEE, 2006) the authors have proposed the Trident tool for traffic generation in Networks-on-Chip, which incorporates a comprehensive traffic model for NoCs that couples the spatio-temporal profiles of traffic. Three statistical parameters, the hop count, Hurst exponent and packet injection distribution, are used. This 3-tuple traffic model was empirically derived, based on NoC traces gathered from full-system simulations of thirty applications on three different chip architectures.

The hop count parameter was derived from the distribution of the probabilities of transmission and consumption of packets in NoC nodes taken with respect to the distance between them (the nodes). The packet injection distribution follows a Gaussian distribution, fitted to the measures of the packet injection rates of every node in the NoC. According to the authors, these two parameters, associated with a spatial traffic pattern¹, should be enough to characterize the traffic in the spatial domain.

The first disadvantage of this work comes from the characterization of the traffic in the time domain, with the Hurst exponent estimator being the same as the periodogram-based method, used in (VARATKAR; MARCULESCU, 2004), therefore having some significant shortcomings, as mentioned before; the traffic generation

¹The spatial traffic pattern accounts for how the destination of a packet is determined, given its origin, and it is independent of the hop count and packet injection rate.

method adopted by the authors is also similar to one in (VARATKAR; MARCULESCU, 2004), which is a single parameter model, risking oversimplification of the generated traffic. Another disadvantage of the work is the fact that all the traffic measurements were made through the use of cycle-accurate simulations, what required significant implementation work and high simulation times, given that thirty application were executed.

3.3 Scherrer et. al. proposition

In the works of Scherrer et. al. (SCHERRER; FRABOULET; RISSET, **Proceedings**. New York: IEEE, 2006a; SCHERRER; FRABOULET; RISSET, **Proceedings**. New York: IEEE, 2006b; SCHERRER; FRABOULET; RISSET, 2009), a methodology for traffic modeling was proposed based on cycle accurate simulations of multimedia applications being executed in a simple three element platform, composed of a processor, a cache and a main memory. The platform, built with components from the SoCLib (SOCLIB, 2016d) library, was conceived in this minimalistic way with the objective of studying the on-chip traffic characteristics and modeling, without the influence from other processing elements, as it would be in a larger NoC-based platform.

The architecture of the proposed platform is illustrated in Figure 3.1, where it can be seen that a MIPS, a three stage pipelined ISS model (SOCLIB, 2016c), is directly connected to an instruction and data cache which is, on its turn, directly connected to a RAM (Random Access Memory). The traffic is measured between the cache and the RAM, being the data transfer formalism defined under five elements: (1) destination address, (2) command type (read or write), (3) length of the data transfer, (4) transfer delay and (5) initial time of the transfer. The authors were able to measure these five elements in the simulation of the platform through the use of VCD (Value Change Dump) files and, from it, they were able to obtain the data throughput² associated with the transfers.

The data throughput is the measure of primary interest and this measurement is assembled as a time series. On top of this time series the authors have applied the *k-means* algorithm to identify one or more relevant traffic sequences, that follow similar

²The throughput of a data link is defined as the data (bits, bytes, flits or packets) counting by the time unit (ns, ms, s, etc.).

Figure 3.1: Architecture of the platform for traffic measurement

Adapted from (GONZALEZ, 2013)

statistical properties; they are called sequence phases of the traffic. An illustration of the results of this process applied to the traffic of three different applications is shown on Figure 3.2. Having identified these phases, the authors analyzed the LRD characteristics of each one of them through the use of the Abry-Veicht spectrum method, extracting the Hurst exponents; the Abry-Veicht spectra for the three applications from Figure 3.2 are shown in Figure 3.3.

These analyses were performed with the objective of generating synthetic traffic series, which was done by fitting ARMA models to the series which did not present LRD characteristics and ARFIMA models to the ones that presented them. In the work, the authors implemented a configurable (with the order and duration of the phases as some of the parameters) cycle-accurate traffic generator, named Multi-Phase Traffic Generator (MPTG), to insert the synthetic series in cycle accurate models of grid topology NoCs.

The MPTG was developed with the aim of inserting traffic in a NoC-based platform without the use of actual processors, looking to reduce simulation times. However, the simulations with the MPTG have shown no significant gain in time when compared with the simulation with the ones performed with actual processors (SCHERRER; FRABOULET; RISSET, **Proceedings**. New York: IEEE, 2006b), what

Figure 3.2: Applications traffic series with detected phase

Source: (SCHERRER; FRABOULET; RISSET, 2009)

Figure 3.3: Applications traffic series Abry-Veicht spectra

Source: (SCHERRER; FRABOULET; RISSET, 2009)

can be attributed to the use of cycle-accurate models in the simulation. This result reinforces the motivation of the present work for working with TLM based traffic generators.

3.4 Bogdan et. al. propositions

The works from Bogdan et. al. (BOGDAN; MARCULESCU, **Proceedings**. New York: IEEE, 2010; BOGDAN; MARCULESCU, 2011) were the first ones to highlight the presence of multifractality in simulations of NoC-based MPSoCs. Using cycle-accurate simulators, the authors assessed large MPSoCs (based on large mesh topology NoCs, ranging from 4×4 to 35×35 meshes) running several benchmark applications; besides that, they have proposed a methodology for quantifying the multifractal characteristics of the traffic as a function of packet injection rates of processing elements. On the development of this methodology, the authors have built an analytical model based on statistical physics, where a relation between the temperature of a gas and the packet injection rates of the processing elements was established. From this analytical model, the authors were able to extract performance measurements, such as the probability of buffer overflow or the missing a deadline, for applications with packet deadline requirements.

The main disadvantages of Bogdan's works are that they rely on cycle-accurate simulations, leaving the advantages of using TLM unexplored. The works have also limited its experiments to uniform NoC-based MPSoCs, where the same (processing) element is attached to every node of the NoC. This imposes a limitation since platforms with this uniform configuration relies on distributed memory schemes, what leaves platforms with centralized memories unexplored and disregards applications being executed in less complex platforms, such as mono-processed platforms.

3.5 Sepulveda et. al. proposition

In (SEPULVEDA et al., **Proceedings**. New York: IEEE, 2010), the authors have proposed a TLM framework for evaluating NoC-based SoCs, through the use of generators that could produce LRD traffic. To the extent of the research supporting the present work, the work in (SEPULVEDA et al., **Proceedings**. New York: IEEE,

2010) was the first, as far as we know, to unite TLM simulations with LRD traffic generators.

The synthetic traffic was generated using fractional Gaussian noise (fGn) with the Box-Müller method. As exposed in the previous chapter, the fGn has the Hurst exponent as the single parameter. The authors have simulated the platform with the traffic generators using three different values for the Hurst exponent and obtained results that indicated an increase in queue formation in the buffers with the increase in the degree of self-similarity in the synthetic traffic; this is coherent with results based in cycle-accurate simulations present in the literature.

A shortcoming of the work is the lack of a methodology for parameter extraction, relying on pure synthetic traffic. Both its temporal and spatial characteristics are synthetic; neither the Hurst exponents used for the generation of the synthetic traffic sequences nor the distribution of packets destinations were extracted from full system simulations, meaning these properties were not extracted from the execution of real applications.

3.6 Pekkarinen et. al. proposition

In (PEKKARINEN et al., **Proceedings**. New York: IEEE, 2011) the authors have proposed a methodology for NoC-based MPSoC performance evaluation through application benchmarking. The proposition relies on the Transaction Generator tool (KANGAS; HÄMÄLÄINEN; KUUSILINNA, 2006), which is written in SystemC and works with abstract high level application workload models based on task graphs described in Extensible Markup Language (XML) format. The Transaction Generator tool creates traffic to the NoC according to the description contained in the XML file and measures various performance metrics during simulation.

The concept of the Transaction Generator tool is illustrated in Figure 3.4, where it can be seen that the abstract high level application workload model, represented as a task graph, is mapped to the MPSoC processing elements (PEs) which, on their turn, are connected to a model of a NoC. The processing elements are simplified RISC processors models and the tasks on the diagram are not the actual application tasks, but code snippets that either mimics the computational functioning of an actual application task (e.g., a multimedia application) or represents a synthetic application.

Figure 3.4: Overview of the Transaction Generator tool

Source: (PEKKARINEN et al., **Proceedings**. New York: IEEE, 2011)

Hence, the traffic inserted into the communication structure is originated from the execution of these code snippets.

This approach presented by the authors have the advantage of allowing a significant exploration of the design space of the MPSoC given the simplicity of application modeling and mapping provided by the use of XML configuration files. However, the simulation of the processors is done in the cycle accurate level and the process of transforming an application into the high level workload model may not capture all the significant properties of the execution of the application, which is not the case for traffic generators based on formal time series statistics extracted from the actual execution of the application.

3.7 Gonzalez et. al. proposition

Based on the coherence found between the results of TLM and cycle-accurate simulations shown in (SEPULVEDA et al., **Proceedings**. New York: IEEE, 2010), the authors of (GONZALEZ et al., **Proceedings**. New York: IEEE, 2013; GONZALEZ,

2013) aimed to take the proposition further, by analyzing traffic series collected from the execution of real applications performed on a TLM platform. Besides the analysis, Gonzalez's work proposes a methodology for fitting auto-regressive models to the original series with the objective of generating synthetic traffic and, also, compares the accuracy of several Hurst exponent estimators in the context of on-chip traffic.

The authors took the methodology presented in (SCHERRER; FRABOULET; RISSET, 2009) as basis for the work, using the same minimalistic platform shown in Figure 3.1; its is also assembled with SoCLib components, with the difference that the platform was build on both cycle-accurate and TLM levels. As test applications, the authors used an Micro Four Thirds (MFT) algorithm, an image smoothing filter (SIGMALAB, 2016), and a Motion-JPEG (MJPEG) decoder (SOCLIB, 2016a).

On top of the traffic series captured from the simulations, the following statistics were computed: the autocorrelation function, the periodogram and the Abry-Veicht spectrum. Besides all these computations, the Hurst exponent was estimated for all the series using four different estimators presented on the literature.

The main result presented by the work was an exposition and a thorough statistical analysis, showing the LRD characteristics present in the cycle-accurate traffic, resulting from the execution of the MJPEG application, are maintained in the TLM traffic. This result was achieved by comparing the statistical computations mentioned in the previous paragraph for the two traffics.

Another significant result presented in the work was the exposition of an analysis showing that the ARFIMA model is suitable for modeling the on-chip LRD traffic, conclusion derived from the fact that the residuals³ of the model are quasi-normal and uncorrelated, what was shown through the use of histograms, QQ-plots and autocorrelation functions. It was also shown that the Abry-Veicht estimator for the Hurst exponent is the most accurate estimator when compared to the other tested estimators, what corroborates with previous results from Internet traffic analysis.

Despite all the results concerning the synthesis of LRD traffics through the ARFIMA model, it was not presented in the work a methodology for building and validating a traffic generator. It was also not considered the higher order statistics of the traffic sequences, what could point out to multifractal properties.

³The residuals of a model is a time series resulting from the difference between the original time series and the fitted time series.

4 SYNTHETIC TRAFFIC GENERATION METHODOLOGY

As mentioned earlier in Section 1.3, the general objective of this work is to develop a traffic capture methodology through the use of systems described in TLM and to propose a test packet switching traffic generator, also modeled in TLM, that can produce traffic with multifractal properties. In this chapter the actions needed to achieve this general objective will be presented, all grounded in the presented motivations and in the proposed theoretical foundations exposed in the previous chapters. All of the computations comprising the steps of the methodology were performed with the use of scripts written in R (CRAN, 2016), a widely utilized language for statistical computations, and MATLAB (MATLAB, 2016a).

4.1 Proposed methodology overview

The methodology can be summarized by the six steps of the diagram shown in Figure 4.1:

1. The designer must assemble a TLM simulation platform composed of a master and a slave element connected by a communication structure, to capture the data flow generated by their communication.
2. The traffic capture consists in interpreting the information (communication transactions) collected in the simulation performed in the previous step. This information is parsed to form a time series through an aggregation process and it is written to files.
3. The time series obtained in the previous step have their multifractal spectra computed, in order to assess the presence of multifractality. After this, the series

have their Hurst exponents estimated, in order to select the proper monofractal (auto-regressive) model to be fit to it.

4. After the traces' properties have been identified, the multifractal Wavelet model (MWM) is fit to the traces and the model's parameters are extracted. The simulation of the fitting models with their parameters provides new time series, which are denominated in this text as **MWM-fitted time series** or **MWM-fitted traffic profiles**. In parallel, for comparison purposes, auto-regressive (monofractal) models are also fit, whose simulation produces the **ARMA-fitted, or ARFIMA-fitted, time series or traffic profiles**. The model-fitted time series are written to files to feed the traffic generator.
5. The files obtained in the previous step supply information to the traffic generator, which transforms the time series information to communication transactions, feeding them to the system.
6. The traffic profiles produced by the traffic generator, obtained through different methods (referring to the different aggregation procedures and different fitting models) are recaptured and data are re-aggregated into the called **synthetic time series or synthetic traffic profiles**. These synthetic time series are compared against the time series converted from traces obtained from cycle-accurate simulations. This last step may not be, in fact, a part of the methodology for planning and building a traffic generator, but it is the part of our study to develop the methodology itself. The comparison between traffic profiles will allow to identify the best methods and aggregation window sizes, and validate it.

In the following sections more details of each step are going to be given.

4.2 Platform specification and simulation

4.2.1 SoCLib simulation platform

SoCLib (SOCLIB, 2016d) is a free and open source platform conceived for MPSoC prototyping, developed as a project in the French National Research Agency (ANR) and maintained by the Paris Informatics Laboratory (LIP6). SoCLib is basically

Figure 4.1: Methodology diagram

Table 4.1: SoCLib components

General purpose processors	MIPS32, SPARC V8, ARM V6-T, PowerPC 405, Lattice Micro32, Nios2, Microblaze
Digital signal processors	ST231, TMS320, C62
Buses and crossbars	Generic system bus, PiBus, Avalon, Token Ring, Local Crossbar
Bridges	PCI, USB, CAN, I2C, Wishbone
NoCs	VCI Generic Micro-Network, DSPIN Micro Network
Memories and controllers	Embedded SRAM, External SDRAM
Dedicated coprocessors	DWT, DEM, MOD, LDPC, TC1000, TC3000
General use components	TTY, Interrupt controller, Blocking device, Lock memories, DMA controller, MWMR controller, Frame Buffer controller, Disk usage controller

Source: (GREINER, 2007)

a library that incorporates modular descriptions of hardware components in both CA and TLM abstraction levels; these components allow the design of virtual prototypes for integrated systems.

The list of components present in SoCLib is given in Table 4.1, including some commercial ones. All of the listed components fall under the directives of the ESL design methodology, since they are modular hardware blocks described in high level non-synthesizable abstraction models, facilitating their reuse in the design flow. The description of the blocks is done with the SystemC language (IEEE. . . , 2012) together with its TLM library and all blocks work with the VCI (Virtual Component Interconnect) communication protocol (GROUP, 2001).

Figure 4.2: The TLM simulation platform

4.2.2 The TLM platform

In the present work, the SoCLib platform and its library of hardware modules were used to assemble a system formed by the triplet *<processor, cache memory, main memory>* with the objective of monitoring the amount of traffic related to the flow of data in the processor's output, by following the proposition of (SCHERRER; FRABOULET; RISSET, 2009). In order to monitor the execution of the application codes and assess the correct functioning of the system, a terminal emulator, denominated TTY, was also used. Since the TTY is a purely passive component it had no influence in the traffic flowing from the processor.

Figure 4.2 presents the TLM platform used in this work for the simulations. The processor is an Instruction Set Simulator (ISS) for a MIPS32 architecture; it lies inside a TLM wrapper that also encompasses a data and an instruction cache memories. The wrapper is also responsible for the processor interfacing under the VCI protocol, therefore it is called the VCI wrapper. The main memory is a random access memory (RAM), which is connected to the processor by a VCI Generic Micro Network (VGMN), shown in the figure simply as the command and response channels.

The codes of the applications are compiled and stored in the main memory and, at the beginning of the simulation, part of the instructions and data from the code are transported to the cache. The traffic that flows from the processor is, then, composed of read transactions (when cache misses occurs, for both data and instructions) and write transactions (when some result needs to be stored in the main memory).

Figure 4.3: Traffic modeling formalism

Source: (SCHERRER; FRABOULET; RISSET, 2009)

4.3 Traffic capture

Any communication action initiated by a processor in a TLM SoCLib platform generates an output transaction T_{out} . The minimum and maximum sizes of the transaction payload were set to 4 and 32 bytes, respectively, following the fashion presented by native examples in SoCLib. If the processor generates a transaction with payload size greater the 32 bytes, the exceeding amount is split up into sequential transactions with smaller payload lengths, generating bursts in the communication pattern.

Besides the payload size, the transaction carries other four properties (SOCLIB, 2016e), namely, (1) the command type (write or read), (2) the target address of the payload, (3) the payload data and (4) the estimated transaction time, which is based on parallel and distributed discrete event principles, presented in (VIAUD; PECHEUX; GREINER, **Proceedings**. New York: IEEE, 2006). Hence, in (SCHERRER; FRABOULET; RISSET, 2009), the transactions initiated by a processor were formalized by $T_{out}(k) = (A(k), C(k), S(k), D(k), I(k))$, where k is the transaction index, $A(k)$ is the payload target address, $C(k)$ is the transaction command type, $S(k)$ is the size of the transaction, $D(k)$ is the delay between transactions and $I(k)$ is inter-request time between transactions. This formalism is illustrated in Figure 4.3¹.

In defining this formal representation for transactions, the next step is to transform it into a time series representation. This transformation process was also retrieved from (SCHERRER; FRABOULET; RISSET, 2009), where the *aggregated throughput*

¹The usage of the 1 ns scale in TLM used by the authors was arbitrary, since the definition of an absolute time scale is mandatory in SystemC

Figure 4.4: Method for obtaining the aggregated throughput

of the transactions composes the time series, called the *traffic* associated to the transactions.

The first step to obtain the time series of the traffic (aggregated throughput) is illustrated in Figure 4.4a, showing a different point of view of the formalism presented in Figure 4.3, by plotting the sizes of the transactions against time. Since the cadence of transactions is related to the application-dependent data requests and responses, the time intervals in the plot are not constant. Over this plot, an observation (or aggregation) window T_{obs} is defined and all of the transactions that lie within this window have their sizes summed, generating the actual traffic plot shown in Figure 4.4b, where all samples are uniformly disposed in time. The aggregation window determine a traffic time series; all of the subsequent statistical analysis and model fitting are applied to its representation.

This aggregation procedure was implemented as an R language script, taking as input the transaction representation of the application's traffic and the size of the aggregation window (T_{obs}), presenting a time series (traffic profile) as output.

4.4 Statistical analysis

In this step of the methodology, all the time series (traffic profiles) obtained in the previous section are subjected to two different statistical analyses. The first is the computation of the series' multifractal spectrum, which allows to assess the

presence of multifractality in time series. It was performed with the use of the FracLab (FRACLAB, 2016) tool, a MATLAB based software for image and signal processing of fractals and multifractals. FracLab presents a Graphic User Interface (GUI) that allows the user to choose a method, among several ones, for the estimation of the multifractal spectrum of one dimensional signals (as the time series being handled); in this work the Legendre spectrum was used.

The second analysis applied to the time series is the estimation of its Hurst exponent, in order to identify the presence of SRD or LRD. This process, which has also been used in a monofractal context (BUENO; GONZALEZ; CHAU, **Proceedings**. New York: IEEE, 2016), is performed with the use of the Abry-Veicht estimator, since, as mentioned in Subsection 2.3.3, it provides an accurate and non-biased estimation of the Hurst exponent. The calculations for this end were performed with an R script, using of the `dwt` function R's *waveslim* package, which calculates the series' Discrete Wavelet Transform (DWT). The Wavelet type is user-specified and, in this present work, the Haar Wavelet was adopted. It allows the form proposed in Equation 2.20 to be implemented and the estimator, $\hat{H}_{AV}$, to be extracted from the slope of a linear regression applied to the plot of the DWT results.

4.5 Model fitting

After the inspection of the results from the analyses in the previous step, it is possible to determine, in a traditional monofractal approach, if the application's traffic series presents SRD or LRD; also, on top of the dependence characteristics, it is possible to assess if the series show multifractal properties. Since the multifractal model is supposedly general, the MWM will be used to fit all series; on the other hand, by properly identifying the SRD and LRD characteristics, an ARMA or ARFIMA model, respectively, will be used to fit to the series and used for assessing the multifractal model accuracy.

The fittings of the MWM were performed using MATLAB codes made available by the authors of (RIEDI et al., 1999; RIEDI, 2016); the used function was the `train_beta_mwm`. The fittings of the ARMA and ARFIMA models were performed with an R script, using the `auto.arima` function from the *forecast* package and the `arfima` function from the *fracdiff* package, respectively.

Figure 4.5: Model fitting process

The outputs of the aforementioned functions are the parameters of the models; the fitting of a traffic profile with its particular aggregation window leads to its specific parameters. In order to generate the model-fitted time series, these parameters must be used in model simulation. For the MWM, this is done with the `gen_beta_mwm` function; for the ARMA and ARFIMA models, the `simulate.Arima` and `fracdiff.sim` functions, respectively, are used.

As for the inputs of the fitting functions, they are the aggregated traffic profiles from the applications, with several different aggregation windows. Therefore, the outputs of the simulation functions (model-fitted traffic traffic generation) correspond also to the different corresponding aggregation windows. The whole process of model fitting and parameter simulation is illustrated in Figure 4.5.

4.6 Traffic generator simulation

After being obtained through the model simulation, each model-fitted (aggregated) traffic profile is parsed to a text file, which will be the input of the built traffic generator. The traffic generator consists of a simple initiator module, developed with the SoCLib TLM writing rules (SOCLIB, 2016e); the module is a black-box model of the TLM VCI wrapper, containing TLM VCI command and response interfaces. This has allowed the traffic generator module to be integrated to SoCLib and substitute the processor in the original TLM platform (Figure 4.2), resulting in the scheme presented in Figure

Figure 4.6: The traffic generator

4.6.

The main task of the traffic generator module is to convert the aggregated traffic (time series) information contained in the text file back into a sequence of transactions, meaning that the process shown in Figure 4.4 must be inverted, generating transactions with data assigned in irregular time intervals. In (MELLO, 2006), two methods for this inversion process were proposed:

- Conversion method 1: the values of the aggregated traffic time series (V) are divided by a predetermined amount known as Maximum Transfer Unit (MTU). V is split up into $\lfloor V/MTU \rfloor$ packets of size MTU , and the exceeding bytes (rest of the division) are put into a packet of a predetermined minimum size ($minTU$), even if the amount of exceeding bytes is smaller than $minTU$. These packets are disposed in fixed and small (in comparison with T_{obs}) equally spaced time intervals, as illustrated in Figure 4.7.
- Conversion method 2: this method is similar to the method 1, but the time intervals that the packages are disposed are no longer fixed, being uniformly disposed in T_{obs} , and the sizes of the packets no longer follow the MTU paradigm. An illustration of this method is shown in Figure 4.8.

In (MELLO, 2006), no significant differences between the two aforementioned methods were notice when their outputs are re-aggregated. This fact led to a conclusion that the transformation method does not exert much influence in the final results, therefore, in this present work a hybrid method for the conversion was

Figure 4.7: Conversion method 1

Adapted from (MELLO, 2006)

Figure 4.8: Conversion method 2

Adapted from (MELLO, 2006)

used, combining parts of the methods 1 and 2 above. The packets are transmitted in a uniform disposition in T_{obs} and follow the MTU paradigm, with $MTU = 32\text{bytes}$ and $\text{min}TU = 4\text{bytes}$. This was done with the objective of fitting the model-fitted transactions more accurately to the original observed transactions.

Due to the adopted conversion method, it is expected that, when using model-fitted traffic profiles resulting from the modeling of an original traffic, aggregated with wider windows, the traffic generator will end up producing faster simulations if compared to cases with narrower windows. This happens since, in the wider window cases, there will be less entrances in the code loop of the method implementation. However, also due to wider aggregations, the model-fitted traffic may lose some accuracy; this trade-off will be explored in the following methodology step.

4.7 Synthetic traffic validation

In order to validate the methodology for using the TLM traffic generator in a platform rather than the CA processor with the application, the traffic captured from their simulations may be compared. The capture of the comparison reference traffic,

from the simulation of a CA system, can be achieved by replicating the original TLM platform, executing equivalent applications. The validation is performed by computing the statistical characteristics of this CA reference traffic profile and comparing them to the synthetic traffic obtained from the simulations with the TLM traffic generators (for each aggregation window case); this should follow the methodology proposed in (BUENO; GONZALEZ; CHAU, **Proceedings**. New York: IEEE, 2016), which relies in all the time series analysis methods presented in Section 2.3. Some aspects of these different analyses, with respect to the implementation of the computations, are presented in the list below.

1. All the TLM traffic generator model-fitted transactions, for different fitting models and aggregation window sizes, must be re-converted and re-aggregated into synthetic time series in order to provide statistical characteristics for comparison. Their re-aggregation window size, as for the the aggregation window size of the real CA time series, obtained from the CA platform collected transactions, must be the one of a common validation aggregation window.
2. The autocorrelation function is computed with the use of the `acf` function present in the *stats* package of the R language. Through the autocorrelation function one is able to assess the presence of SRD or LRD by, respectively, observing over the function plot, an exponential or hyperbolic decay.
3. The periodogram is computed with the use of the `spectrum` function also from R's *stats* package. Since the output of this function can be very noisy, an autoregressive filter can be applied as an optional; in this present work, this filtering has been adopted, aiming to present a clearer and more readable periodogram. The presence of LRD can be verified in the periodogram as a pole near the frequency zero.
4. The Abry-Veicht spectrum is computed with the same R script used for the calculation of $\hat{H}_{AV}$, mentioned in Section 4.4, with the difference that here the focus is on plotting the spectrum and not only on calculating its slope.

Concerning the assembling of the CA platform, it resembles the one presented in Figure 4.2, having a MIPS32 processor with the same ISS used in the TLM simulation but attached to a CA cache memory and CA VCI wrapper; the main memory and the Generic Micro Network are also described in the CA level for this platform.

Figure 4.9: SoCLib's CA VCI signaling scheme

Source: (GREINER, 2007)

Differently from the TLM platform, where there is no presence of signals (due to the higher abstraction level), the CA platform follows the VCI signaling scheme, illustrated in Figure 4.9. The channels, through which the signals flow, are grouped under three sets:

- **COMP:** set shared bidirectional channels, comprising *command* (CMD), *address* (ADDRESS), *packet length* (PLEN), *source identification* (SRCID) and *thread identification* (TRID) signals.
- **CMD:** set of unidirectional command channels, comprising *valid* (CMD VAL), *acknowledge* (CMD ACK), *end of packet* (CMD EOP), *packet identification* (CMD PKTID) and *write data* (CMD WDATA) signals.
- **RSP:** set of unidirectional response channels, comprising *valid* (RSP VAL), *acknowledge* (RSP ACK), *end of packet* (RSP EOP), *packet identification* (RSP PKTID), *read data* (RSP RDATA) and *response error* (RSP ERROR) signals.

The most relevant signals for traffic capture are CMD, ADDR, PLEN and WDATA, which can be written to a VCD file during the platform simulation. This file can then be parsed to resemble the format of the output of the TLM platform, allowing it to be subjected to the proposed statistical analysis.

Since the described CA platform forms the validation system, the speed-up of the simulation with the TLM traffic generator is calculated against the CA platform

simulation time, i.e., the speed-up is given by the formula:

$$\text{Speed-up} = \frac{\text{CA platform simulation time}}{\text{TLM traffic generator simulation time}} \quad (4.1)$$

5 RESULTS

In this chapter the results obtained from the implementation of the proposed methodology are presented and discussed. All of the simulations and calculations were executed in a computer with an Intel® Core™ i7-2630QM CPU 2.00 GHz and a 7.7 GiB RAM memory running the Ubuntu 16.04 LTS operational system.

5.1 Test applications

In order to test the traffic generation methodology, several experiments have been proposed, being performed on the TLM and CA platforms described in Sections 4.2 and 4.7, respectively. Three application codes were used, all of them written in C language and cross-compiled to a MIPS32 target architecture. Since the used cross-compiler (suggested by the SoCLib installation notes (SOCLIB, 2016b)) cannot handle floating point variables, only fixed point applications were selected and simulated.

The utilized applications, listed in a crescent order of size (code lines), are:

- **Huffman**: its code, obtained from (ROSETTACODE, 2016), is an implementation of a priority queue based Huffman compression algorithm. The input of the program is a 500 words text (inserted directly in the code and, therefore, placed in the main memory), whose character frequency is computed and utilized for the compression.
- **MJPEG**: used in (SCHERRER; FRABOULET; RISSET, 2009) and in (GONZALEZ, 2013), this MJPEG (Motion JPEG) video decoder is a native SoCLib application. Figure 5.1 illustrates how the decoder works. The input is a stream of video composed of 24 frames, which are compressed 256×256 JPEG images. The stream of frames is delivered to the *demux* block, which transfers

Figure 5.1: Functioning of the MJPEG decoder

Adapted from (GONZALEZ, 2013)

Figure 5.2: Functioning of the MFT application

the flow of data to the other blocks; the *VLD* block performs a variable longitude Huffman decoding, passing the resulting coefficients to be reordered and inversely quantized in the *IQ-ZZ* block. The final block, *IDCT*, performs an inverse cosine transform and passes the information to the *LIBU* block, which allows the flow of data to be written to the main memory.

- **MFT**: this application, which was also used in (GONZALEZ, 2013), is available at (SIGMALAB, 2016); it is called MFT because it is part of the *Micro Four Thirds* system, present in DSLR (Digital Single Lens Reflex) digital photography cameras. The application works by moving a 3×3 kernel (matrix) across a gray-scale image (a 512×512 matrix placed in main memory); as the kernel moves, it replaces its center value by its median, resulting in a smoothed output image, as illustrated in Figure 5.2.

The simulation times of all the aforementioned applications are summarized in Table 5.1, respectively, in the second and third lines, where it can be seen that the

TLM simulations are faster than the CA ones, as expected. The maximum and minimum speedup were of 2.45 times and 1.95 times, as shown in the fourth line of the table.

Table 5.1: Simulation times

	Huffman	MJPEG	MFT
CA	3470	19400	34500
TLM	1440	9930	14100
Speedup	2.41	1.95	2.45

5.2 TLM Traffic capture and analysis

The communication transactions in the TLM platform were monitored in the aforementioned simulations and the transmitted data were captured according to the formalism presented in Section 4.3. Through the aggregation process illustrated in Figures 4.4a and 4.4b (on page 64), they were transformed into time series (traffic profiles). For the TLM simulations, the profiles were obtained by adopting five different aggregation windows, of 25, 50, 100, 200 and 500 nanoseconds.

The reason for applying different windowings is to verify the trade-off between the simulation speedup and the synthetic traffic profiles accuracy, this last one assessed by comparing them to the traffic profiles obtained from the CA simulations of the applications, using all of the time series analysis methods proposed in Section 2.3. It is expected that the first parameter will increase with the window size, due to the implementation of the generator, as will be presented in Subsection 5.3.2), while the second should decrease, since less sampling points would be used.

As a first step to verify the aforementioned trade-off, all of the obtained TLM traffic profiles (resulting from the five aggregation windows) must be analyzed, in order to justify the usage of the synthesis models. Therefore, the profiles had their multifractal spectra and Hurst exponents estimated, as detailed in the following subsections.

5.2.1 Multifractal spectra estimation

The multifractal spectra of the profiles are computed according to the formalism given in Subsection 2.3.4 and plotted, as presented in Figure 5.3.

Figure 5.3: Multifractal spectra of the TLM traffic profiles

(a) Huffman

(b) MJPEG

(c) MFT

It can be seen in Figure 5.3a that the Huffman application traffic profiles are monofractal for the 25ns and 50ns aggregation windows since they show single point spectra; for the other aggregation values, the profiles exhibit multifractality although for the case of 100ns window, the fractal dimension range is quite narrow (from 0.69 to 0.70), lying behind the spectrum of the 200ns window. For the MJPEG application, as seen in Figure 5.3b, all of the profiles exhibit significant multifractality, since all the spectra are wide, spanning at least 0.25 fractal dimensions. The MFT profiles are the ones that present the least amount of multifractality, with the exception for the 500ns aggregation window case, as seen in Figure 5.3c.

In this present work, we have tested the influence of the aggregation window size on the multifractal characteristics of the traffic profiles and have noted the amount of fractal dimensions (width of the multifractal spectrum) in the TLM simulations tends to increase with the enlargement of the aggregation window. For the Huffman and MFT cases, multifractal characteristics appear only in larger aggregation window cases. It is worth noticing that this behavior, derived from TLM traffic capture, is similar to what occurs in the CA simulations, as shown in the figures of Appendix A. This indicates the traffic capture performed at TLM level maintains the CA traffic profiles characteristics, concerning multifractal parameters, what extends the results presented in (BUENO; GONZALEZ; CHAU, **Proceedings**. New York: IEEE, 2016), which showed the validity in adopting TLM simulations for monofractal parameters.

5.2.2 Hurst exponent estimation

The Hurst exponents of the profiles were estimated through the Abry-Veicht spectrum (through the procedure presented in Subsection 2.3.3), being presented in Table 5.2. The Hurst exponent computation guides the proper selection of the auto-regressive model to be fit, i.e., ARMA for SRD profiles ($H < 0.5$) and ARFIMA for the LRD profiles ($H > 0.5$). As seen on Table 5.2, all of the profiles of the Huffman and MJPEG applications are LRD and all of the profiles of the MFT application are SRD, therefore the ARFIMA and ARMA models, respectively, will be fit to them. On the other hand, despite the type of fractality present in the traffic profiles, the MWM will be fit to all of them.

Table 5.2: Hurst exponents of the traffic profiles

	Huffman	MJPEG	MFT
25	0.857	0.661	0.343
50	0.887	0.673	0.349
100	0.910	0.670	0.364
200	0.926	0.654	0.386
500	0.902	0.623	0.487

5.3 Synthetic traffic obtainment and validation

5.3.1 MWM and auto-regressive model fitting

With the multifractal and dependence properties identified, it was possible to adjust the fitting models described in Section 2.4 (page 45) to all the traffic profiles (referring to all the aggregation windows used) obtained from all three applications: (1) Multifractal Wavelet Model, MWM; (2) auto-regressive models, either ARMA or ARFIMA. According to the results from Table 5.2, ARMA is used to be fit to the MFT traffic profile, while ARFIMA is adopted to the MJPEG and Huffman cases. Following the fitting procedure described in Figure 4.5 (page 66), there are 10 different model fittings for each application, resulting in 10 different model-fitted traffic profiles and, therefore, 10 synthetic traffic profiles.

5.3.2 Traffic generator implementation

As described in Figure 4.6 (in page 67), the traffic generator collects model-fitted traffic profiles, which are produced by the MWM and auto-regressive models, under the parameters obtained through the fitting process. These profiles are stored in text files.

Following the hybrid method proposed in Section 4.6, the 10 model-fitted traffic profiles are transformed back into sequences of transactions. The implementation of this method, presented in the code snippet below, forms part of the traffic generator; the remaining parts are responsible for the VCI interfacing and for the conformation of the developed component to the SoCLib rules.

```

1 std::fstream trace(m_trace_file_name);
2 unsigned int agg, acc_nb, acc_t, nb, t;
```

```

3
4 while(trace >> agg){
5     acc_nb = 0;
6     acc_t = 0;
7
8     while(acc_nb < agg){
9         if(agg - acc_nb < 32){
10             nb = agg - acc_nb;
11             t = m_window_length - acc_t;
12         }
13         else{
14             nb = 32;
15             t = (int) m_window_length/((agg/32) + 1);
16         }
17
18         /* sets the size of the transaction */
19         m_payload_ptr->set_byte_enable_length(nb);
20         m_payload_ptr->set_data_length(nb);
21
22         /*...*/
23
24         /* sends the transaction and increments time */
25         p_vci->nb_transport_fw(*m_payload_ptr, m_phase, m_time);
26         m_pdes_local_time->add(t*UNIT_TIME);
27
28         acc_nb += nb;
29         acc_t += t;
30     }
31 }

```

Basically, what the code above does is to read, line by line, the trace files (passed as a parameter to the component, through the `m_trace_file_name` variable), which contains the traffic profiles, and split each read value (attributed to the `agg` variable) in 32 bytes sized transactions¹, as stated in lines 19 and 20 of the code. The transaction is, in fact, sent at line 25. The corresponding aggregation window of the traffic profile

¹Usually, the last transaction is set to a smaller size, in order to the value of `agg` be correctly covered, as shown in line 10 of the code.

being used is also passed as a parameter (through the `m_window_length` variable) and is used to set the time increment between the transactions² in line 15; this increment is applied in line 26.

5.3.3 Computation of the synthetic traffic and statistical analysis

Since the direct output of the traffic generator is a sequence of transactions (the same as the output of the processors in both CA and TLM platforms) and not a time series, the process illustrated in Figures 4.4a and 4.4b (on page 64), must be reapplied to allow the time series analysis methods to be used. In this way, following the reasoning of aspect 1 in Section 4.7 (page 68) a validation aggregation window of 80ns was applied, *exclusively for analysis and comparison purposes*, to all 10 transaction cases at the output of the traffic generator, producing 10 synthetic time series, and also to the output of the CA platform. The value of 80ns was chosen with the objective of: (1) evidencing that the aggregation process used for comparing the synthetic traffic profiles is independent from the TLM capture aggregation processes used for generating the model-fitted traffic profiles, and (2) to provide a common basis for the comparisons.

Considering a direct comparison of the synthetic time series, related to the different aggregation cases, to the original CA traffic profiles is difficult, we compare their computed ACFs, periodograms, Abry-Veicht spectra and multifractal spectra, by superimposing and plotting the simulation results (errors with respect to the CA simulations are quantified and analyzed in the next section).

5.3.3.1 Plots visual inspection and accuracy analysis

Visual inspection on the plots may show the trends on accuracy for different applications (Huffman, MJPEG and MFT), TLM capture aggregation windows (25 to 500ns) and fitting models (MWM and any auto-regressive). In order to ease the perception and discussion of such trends, the quantity of plots are reduced and only the plots for the 25ns and 500ns TLM capture aggregation window cases are presented for illustration (the remaining plots are presented in Appendix B). Two sets

²The increment size depends on `agg`, therefore, usually, the last increment is set to a smaller size, in order the correct value of the aggregation window be covered, as shown in line 11 of the code.

Figure 5.4: Huffman application time series: synthetic (25ns TLM capture aggregation window) vs. CA

of curves are shown for each application; Huffman curves are presented in Figures 5.4 and 5.5, MJPEG curves in Figures 5.6 and 5.7, while the MFT ones are in Figures 5.8 and 5.9.

A first general observation on all figures for the computed multifractal spectra of the traffic profiles generated by the MWM and auto-regressive models points to the fact that, apparently, these models restrict the types of their derived traffic profiles. It has been mentioned in Section 5.2.1 that the captured traffic profile from the TLM platform shows, for some large aggregations and in both Huffman and MFT applications, multifractal characteristics that is not observed in the equivalent captured traffic from the CA platform. Regardless of this fact, all the synthetic traffic profiles obtained through the MWM model present multifractal characteristics for all applications, i.e., these characteristics are introduced to the Huffman and MFT applications traffic profiles, for which monofractal characteristics should be present.

Figure 5.5: Huffman application time series: synthetic (500ns TLM capture aggregation window) vs. CA

(a) Autocorrelation function

(b) Periodogram

(c) Abry-Veicht spectrum

(d) Multifractal spectrum

Figure 5.6: MJPEG application time series: synthetic (25ns TLM capture aggregation window) vs. CA

Figure 5.7: MJPEG application time series: synthetic (500ns TLM capture aggregation window) vs. CA

Figure 5.8: MFT application time series: synthetic (25ns TLM capture aggregation window) vs. CA

(a) Autocorrelation function

(b) Periodogram

(c) Abry-Veicht spectrum

(d) Multifractal spectrum

Figure 5.9: MFT application time series: synthetic (500ns TLM capture aggregation window) vs. CA

(a) Autocorrelation function

(b) Periodogram

(c) Abry-Veicht spectrum

(d) Multifractal spectrum

On the other hand, all the synthetic traffic profiles obtained through the ARFIMA or ARMA models present monofractal (on near monofractal) characteristics for all applications, i.e., these are introduced to the MJPEG application traffic profiles, for which multifractal characteristics should be present.

By visually inspecting the plots, it is possible to make some more detailed observations and draw some conclusions for each application.

- Huffman application cases:

- For both the 25 and 500ns TLM capture aggregation cases, it can be observed that the ACFs produced under the MWM fitting are much closer (almost matching) to the ACF of the original CA traffic when compared to the one obtained with auto-regressive model fitting, as seen on Figures 5.4a and 5.5a. The same can be said about the Abry-Veicht spectra, concerning their overall slopes (the ones produced under the MWM fitting are more accurate than the ones produced by the ARFIMA model), as seen on Figures 5.4c and 5.5c.
- For the periodograms, it can be concluded that both the MWM and the auto-regressive model presented similar accuracy, as seen on Figures 5.4b and 5.5b.
- Concerning the computed multifractal spectra, presented in Figures 5.4d and 5.5d, it can be observed that, for the 25ns case, the MWM fitting case spectrum shows to be much closer to the real (CA traffic) spectrum; for the 500ns case, the accuracy of both the synthetic traffic cases from MWM and ARFIMA fitting is similar, with a slight advantage of the first, since its spectrum shows to be closer to the real spectrum.
- The observations above lead to the suggestion that the traffic obtained through MWM fitting is more accurate than the ones obtained through ARFIMA model fitting.

- MJPEG application cases:

- For the ACFs, the accuracy of the synthetic traffic case from MWM fitting for the 25ns aggregation window is greater than the ARFIMA model's one, as seen on Figure 5.6a; for the 500ns case the accuracy of the traffic obtained through both models fitting are equivalent, since the shape of the

functions almost match completely, as it can be seen on Figure 5.7a. Both models also have similar accuracies concerning the Abry-Veicht spectra (similar overall slopes), as seen on Figures 5.6c and 5.7c.

- For the periodograms, a similar observation to the one in the previous item (concerning the ACFs) can be made, as it is seen on Figures 5.6b and 5.7b.
 - For the multifractal spectra, it can be observed that the accuracy of the synthetic traffic case from MWM fitting for the 500ns aggregation window is greater than the ARFIMA model's one, as it can be seen on Figure 5.7d; for the 25ns case, the accuracy of the traffic obtained through both models fitting is similar, with a slight advantage of the synthetic traffic case from MWM fitting, as it is seen on Figure 5.6d, which shows the spectrum of the last mentioned model to be closer to the real spectrum.
 - Similar to the Huffman application, for the MJPEG application, the observations above lead to the suggestion that the traffic obtained through MWM fitting is more accurate than the ones obtained through ARFIMA model fitting.
- MFT application cases:
 - For the 25ns case, considering all the ACF, periodogram and Abry-Veicht spectrum both models present similar accuracies, since they present almost matching behaviors. The same can be said about the multifractal spectra, since the proximity to the real spectrum is almost the same for both models, as seen on the plots of Figure 5.8.
 - For the 500ns case, the synthetic traffic case from MWM fitting presents a better accuracy for the ACF, periodogram and Abry-Veicht spectrum, as seen in Figures 5.9a through 5.9c. For the multifractal spectrum, the ARMA model presented a much better accuracy, almost matching the real spectrum, as it can be seen in Figure 5.9d.
 - The observations above lead to the suggestion that the traffic obtained through MWM fitting is more accurate than the ones obtained through ARMA model fitting; however, in case the original traffic is presents SRD characteristics as for the MFT application, we conclude that, although

Figure 5.10: Synthetic traffic profiles errors

suitable, the MWM fitting does not produce more accurate traffic compared to the synthetic traffic from ARMA fitting.

5.3.3.2 Error computation and accuracy analysis

In order to quantify the observations made in the previous subsection and to assess the synthetic traffic profiles accuracy, a simple mean square error calculation was applied to the ACF, periodogram and Abry-Veicht spectrum plots (for all aggregation windows and models), taking the CA results as reference.

The logarithms (base 10) of the obtained errors are summarized in the plot of Figure 5.10. The errors for the multifractal spectra were not calculated, since, usually, the spectra have different domains, that is, the spectra usually do not overlap each other, making it difficult to apply a mean square error calculation; however, for the same reason, the accuracy in the multifractal spectra can be verified visually as in Subsection 5.3.3.1, by taking in consideration the overall proximity of the spectra.

The plot in Figure 5.10 presents, numerically, the results that have been presented in section; it is possible to observe that synthetic traffic from MWM fitting produced, consistently, higher accuracy results, given that its errors are smaller than the ones produced by the auto-regressive models fitting, specially for the low aggregation cases. The only exception to this observation is the 500ns TLM capture aggregation

window case of the MFT application, where the auto-regressive model produced a slightly lower error. It is also possible to conclude that the synthetic traffic from MWM fitting produced higher accuracy results considering the multifractal spectra, given that it produced spectra with higher proximity to real traffic profiles spectra, with the exceptions of the 100ns, 200ns and 500ns aggregation cases for the MFT application, as seen in Figures B.8d, B.9d and 5.9d, respectively. This loss of accuracy of the synthetic traffic from MWM fitting for some cases in the MFT application may be due to its traffic short range dependence.

In order to better understand the error data illustrated in Figure 5.10, the proportions between the model's errors were calculated (taking the MWM errors as references, that is, the auto-regressive model's errors were divided by the MWM errors) and summarized in Table 5.3.

Table 5.3: Proportions between the errors (auto-regressive/MWM)

	25	50	100	200	500
Huffman	20.52	37.32	33.17	5.67	3.73
MJPEG	5.47	2.30	2.89	1.23	1.31
MFT	58.38	60.52	178.77	6.09	0.56

What Table 5.3 shows is, basically, how many times larger are the errors associated to the auto-regressive models with respect to the errors associated to the MWM model. It can be seen that the values in the table are predominantly greater than one (with the exception of the 500ns window case of the MFT application, in conformity to the observations made in the previous paragraphs), reaching values as high as 37.32 for the Huffman application, 5.47 for the MJPEG application and 178.77 for the MFT application.

From the results in Table 5.3, we draw the conclusion that the synthetic traffic produced by the MWM can be considered to have a higher accuracy than the produced by the other considered models. It is interesting to observe this occurs regardless to the fact that the MWM modeling has introduced multifractal characteristics to the Huffman and MFT applications, suggesting that the MWM can be used as a general fitter for any application, although excels for applications which are typically multifractal as the MJPEG.

5.4 Speedup-accuracy trade-off

The speedups of the simulation with the TLM traffic generator compared to the simulation in the original CA platform, calculated with the formula presented in Equation 4.1 (page 71), are summarized in Tables 5.4 through 5.6, for the Huffman, MJPEG and MFT, respectively.

Table 5.4: Traffic generator simulation times (Huffman application)

	25		50		100		200		500	
	ARFIMA	MWM	ARFIMA	MWM	ARFIMA	MWM	ARFIMA	MWM	ARFIMA	MWM
Simulation time	1200	1150	902	880	745	771	610	550	632	640
Speedup	2.89	3.01	3.85	3.94	4.66	4.50	5.70	6.31	5.50	5.42

Table 5.5: Traffic generator simulation times (MJPEG application)

	25		50		100		200		500	
	ARFIMA	MWM	ARFIMA	MWM	ARFIMA	MWM	ARFIMA	MWM	ARFIMA	MWM
Simulation time	2640	2650	1859	1844	2054	2073	1614	1610	1603	1603
Speedup	7.35	7.32	10.43	10.52	9.44	9.35	12.02	12.05	12.10	12.10

Table 5.6: Traffic generator simulation times (MFT application)

	25		50		100		200		500	
	ARMA	MWM	ARMA	MWM	ARMA	MWM	ARMA	MWM	ARMA	MWM
Simulation time	3987	4000	4017	4012	3356	3349	3280	3205	3093	3102
Speedup	8.65	8.62	8.59	9.00	10.28	10.30	10.52	10.76	11.15	11.12

From these tables, it is possible to see that, for each TLM capture aggregation window, the speedups by using either the MWM or the corresponding auto-regressive model are equivalent, as expected, since the size of the collected traffic profiles with both models are similar; it is also possible to observe that the speedup generally grows with the size of the aggregation windows, however, a saturation can be observed when this window gets larger than 200ns, for all the applications.

An interesting point to observe is the speedup of simulation with the traffic generator with respect to the original TLM platform simulation. If we consider the gain in simulation times shown in Table 5.1 (page 74), it is possible realize that for small TLM capture aggregation windows, there are almost no speedup. However, it is possible that this could be overcome by applying code optimizations or by implementing the synthetic time series generation mechanisms of the models directly inside the traffic generator.

As observed previously, the MWM presents a higher accuracy than the auto-regressive models for the vast majority of the considered cases. In order to quantify

the speedup-accuracy trade-off and to determine which windowing case present the better trade-off, the values presented in Table 5.7 were calculated by multiplying the error proportion values (Table 5.3) to the MWM speedup values; these values allow us to observe how they compensate each other (a high speedup against a low error proportion would give a low trade-off value, for example).

Table 5.7: Quantification of the speedup-accuracy trade-off

	25	50	100	200	500
Huffman	61.77	147.05	149.28	35.84	20.22
MJPEG	40.02	24.16	27.07	14.88	15.84
MFT	503.26	544.71	1841.38	65.54	6.25

It can be observed in Table 5.7 that the larger values are concentrated in the first three columns (25ns, 50ns and 100ns aggregation cases), indicating that the corresponding cases have a better speedup-accuracy trade-off. From this observation, it is reasonable to conclude that the usage of TLM capture aggregation windows in the surroundings of 100ns for the synthetic traffic generator is the best option for traffic generation.

6 CONCLUSIONS

In the present dissertation, a transaction level synthetic traffic generator that can reproduce multifractal characteristics was developed and analyzed, taking as main motivation the work proposed by (BOGDAN; MARCULESCU, 2011) and by (SCHERRER; FRABOULET; RISSET, 2009; GONZALEZ, 2013), whose proposed approaches on traffic modeling and analysis were fundamental for the validation and consolidation of the present work's methodology.

Following the proposed objectives, presented in Section 1.3, and the results, presented in Chapter 5, the following conclusions can be drawn:

- A methodology for capturing traffic from high abstraction level (TLM) simulations and for characterizing it under the multifractal paradigm was well established, following propositions from (SCHERRER; FRABOULET; RISSET, 2009) and (GONZALEZ, 2013), as presented in Sections 4.2 and 4.3. As exposed in those sections, the methodology is composed of five steps: (1) TLM platform simulation, (2) transaction capture and time series obtainment, (3) statistical analysis of the obtained time series, (4) model fitting to the obtained time series and, finally, (5) development of a traffic generator module; a sixth step was established to validate the methodology results.
- The implemented traffic generator was able to transfer the multifractal characteristics from the model-fitted time series (contained in input text files) to the generated transactions and, after the data re-aggregation through a reference validation aggregation window, to the synthetic time series. This affirmation is evidenced in the plots of Figures 5.4d through 5.9d, where its easily verified that the traffic profiles based on the MWM have wide multifractal spectra.
- The synthetic multifractal traffic profiles produced by the traffic generator were statistically compared to a reference CA traffic profile for each application, using

four different analysis (namely, the autocorrelation function, the periodogram, the Abry-Veicht spectrum and the multifractal spectrum); the same procedure was repeated to the monofractal traffic profiles (based on auto-regressive models), as presented in Figures 5.4 through 5.9. The accuracy of the synthetic traffic profiles were quantified for the the three first analyses, while for the multifractal spectra the accuracy was assessed qualitatively. Visual comparison led us to suppose that the multifractal traffic had a superior accuracy; the quantification of the errors confirmed the aforementioned supposition, the multifractal traffic presented lower errors for the vast majority of cases, what allowed us to conclude that the MWM is better suited for on-chip traffic modeling when compared to the monofractal auto-regressive models.

- The simulation speedup with the traffic generator was successfully explored by using several aggregation windows in the processing of the output of the TLM full system simulations. The simulation with traffic generator under MWM fitting showed a speedups ranging from 2.89x to 12.10x (similar values were obtained for the auto-regressive models), being larger for larger aggregation windows. Trading-off the speedup with accuracy, this analysis showed that the accuracy decreases with the increase in the speedup, as illustrated in Figure 5.10 and that the best speedup-accuracy balance would occur for aggregation windows close to 100 ns in our application-examples.

Perspectives and future works

To the extent of our knowledge, this is the first work on TLM on-chip traffic generators based on multifractal models and the first to propose an initial methodological approach for applying these multifractal models in the scope of integrated systems simulation. Several limitations exist and we see several extensions for this work; they are listed as follows:

- In this work, only a simple integrated system (composed of a processor, a memory and simple communication channels) model was used, leaving state-of-the-art systems, such as multi-processed NoC-based systems, unexplored.
- The traffic generator could have been optimized, either through code manipulation or through the usage of different time series to transactions conversion

methods, such as the ones mentioned in Section 4.6. The effects of these alterations could have been explored concerning the multifractality of the output traffic, which was not explored in (MELLO, 2006), and the resulting simulation speedup with the traffic generator. Experiments with different traffic generation methods, such as ones that do not rely on input text files and that can generate traffic in real time, could also have been made and have accuracy and speedup results could be compared to the ones obtained in the present work.

- Different mono and multifractal models could have been used, in order to verify if the advantages noted in this work for the MWM in relation to the auto-regressive models are general and encompasses other models.

Publications

Some preliminary results on which this work's methodology was based on were published in the paper "*Intra-chip traffic generation under autoregressive models based on time series obtained by TLM simulation*", which was presented in the *29th IEEE International System-on-Chip Conference (SOCC'2016)*, held in Seattle, USA, on the 9th of September of 2016.

REFERENCES

- ABRY, P. et al. The multiscale nature of network traffic: discovery, analysis, and modeling. **IEEE Signal Processing Magazine, special issue "Analysis and Modeling of High-Speed Data Network Traffic"**, v. 19, n. 3, p. 28–46, May 2002.
- ABRY, P.; GONCALVES, P.; FLANDRIN, P. Wavelet-based spectral analysis of 1/f processes. In: IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH, AND SIGNAL PROCESSING (ICASSP'93), 1993, Minneapolis, MN, USA. [S.l.: s.n.], **Proceedings**. New York: IEEE, 1993. v. 3, p. 237–240.
- ABRY, P.; GONCALVES, P.; VEHEL, J. L. *Scaling, Fractals and Wavelets*. [S.l.]: Wiley, 2009. ISBN 9781848210721.
- BAILEY, B.; MARTIN, G.; PIZIALI, A. *ESL design and verification: a prescription for electronic system-level methodology*. [S.l.]: Morgan Kaufmann Publishers, 2007. ISBN 978-0-12-373551-5.
- BERAN, J. et al. *Long-Memory Processes: probabilistic properties and statistical methods*. [S.l.]: Springer, 2013.
- BLACK, D. C. et al. *SystemC: From the Ground Up, 2nd ed.* [S.l.]: Springer, 2009. ISBN 9780387699585.
- BOGDAN, P.; MARCULESCU, R. Non-stationary traffic analysis and its implications on multicore platform design. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 30, n. 4, p. 508–519, 2011.
- _____. Quale - a quantum-leap inspired model for non-stationary analysis of noc traffic in chip multi-processors. In: NETWORKS-ON-CHIP (NOCS), 2010 FOURTH ACM/IEEE INTERNATIONAL SYMPOSIUM ON. [S.l.: s.n.], **Proceedings**. New York: IEEE, 2010. p. 241–248.
- BOURKE, P. *Fractal Dimension Calculator*. 2003. Available in: <http://paulbourke.net/fractals/fracdim/>. Accessed in: March 29, 2017.
- BUENO, J.; GONZALEZ, J.; CHAU, W. Intra-chip traffic generation under autoregressive models based on time series obtained by tlm simulation. In: IEEE INTERNATIONAL SYSTEM-ON-CHIP CONFERENCE (SOCC'2016), 29., 2016, Seattle, WA, USA. [S.l.: s.n.], **Proceedings**. New York: IEEE, 2016.
- COPPOLA, M. e. a. *Design of Cost-Efficient Interconnect Processing Units: Spidergon STNoC*. [S.l.]: CRC Press, Inc., 2008. ISBN 1420044710.
- CRAN. *The R Foundation*. 2016. Available in: <https://www.r-project.org/foundation/>. Accessed in: March 29, 2017.

DALLY, W. J.; TOWLES, B. P. *Principles and Practices of Interconnection Networks (The Morgan Kaufmann Series in Computer Architecture and Design)*. [S.l.]: Morgan Kaufmann, 2004. ISBN 9780122007514.

DAMARAJU, S. et al. A 22nm ia multi-cpu and gpu system-on-chip. In: IEEE INTERNATIONAL SOLID-STATE CIRCUITS CONFERENCE (ISSCC'12), 2012, San Francisco, CA, USA. [S.l.: s.n.], **Proceedings**. New York: IEEE, 2012. p. 56–57.

DAS, S.; PAN, I. *Fractional Order Signal Processing: introductory concepts and applications*. [S.l.]: Springer Briefs in Applied Sciences and Technology, 2011. ISBN 9783642231162.

FALCONER, K. *Fractal Geometry: mathematical foundations and applications*. [S.l.]: Wiley, 2003. ISBN 9780470848616.

FINDENIG, R. et al. Fast and accurate UML state chart modeling using TLM. In: IEEE INTERNATIONAL HIGH LEVEL DESIGN VALIDATION AND TEST WORKSHOP (HLDVT'10), 2010, Anaheim, CA. [S.l.: s.n.], **Proceedings**. New York: IEEE, 2010. p. 97–102.

FRACLAB. *FracLab - A fractal analysis toolbox for signal and image processing*. 2016. Available in: <<https://project.inria.fr/fraclab/>>. Accessed in: March 29, 2017.

FREEDMAN, D.; PISANI, R.; PURVES, R. *Statistics (Fourth Edition)*. [S.l.]: W. W. Norton & Company, 2007. ISBN 9780393929720.

GARZON, J. S. E. *Estimativas de desempenho da estrutura de comunicação de SoC a partir de modelos de transações*. Thesis (PhD in Electrical Engineering) — Departamento de Engenharia de Sistemas Eletrônicos, Universidade de São Paulo, 2008.

GHENASSIA, F. *Transaction-Level Modeling with SystemC: TLM concepts and applications for embedded systems*. [S.l.]: Springer, 2005. ISBN 9780387262321.

GONZALEZ, J. et al. Long range dependence in intrachip transaction level traffic. In: IEEE LATIN AMERICAN SYMPOSIUM ON CIRCUITS AND SYSTEMS (LASCAS), 4., 2013, Cusco, Peru. [S.l.: s.n.], **Proceedings**. New York: IEEE, 2013. p. 1–4.

GONZALEZ, J. L. *Ajuste de tráfego intrachip obtido por simulação no nível de transações a modelos de séries autossimilares*. Dissertation (Masters in Electrical Engineering) — Departamento de Engenharia de Sistemas Eletrônicos, Universidade de São Paulo, 2013.

GRANGER, C. W. J.; JOYEUX, R. An introduction to long-memory time series models and fractional differencing. **Journal of Time Series Analysis**, Blackwell Publishing Ltd, v. 1, n. 1, p. 15–29, 1980.

GREINER, A. Soclib : Une plate-forme de prototypage virtuel pour systemes multi-processeurs integres sur puce. 2007. Available in: <<http://leom.ec-lyon.fr/fetch/programme/greiner.pdf>>. Accessed in: March 29, 2017.

GROUP, O.-C. B. D. W. *VSI Alliance™ Virtual Component Interface Standard*. 2001. Available in: http://home.mit.bme.hu/~feher/MS_C_RA/VCI/VCI.pdf. Accessed in: March 29, 2017.

IEEE Standard for Standard SystemC Language Reference Manual. *IEEE Std 1666-2011 (Revision of IEEE Std 1666-2005)*, p. 1–638, Jan 2012.

JIANG, N. et al. A detailed and flexible cycle-accurate network-on-chip simulator. In: IEEE INTERNATIONAL SYMPOSIUM ON PERFORMANCE ANALYSIS OF SYSTEMS AND SOFTWARE (ISPASS'13), 2013, Austin, TX. [S.l.: s.n.], **Proceedings**. New York: IEEE, 2013. p. 86–96.

KANGAS, T.; HÄMÄLÄINEN, T. D.; KUUSILINNA, K. Scalable architecture for soc video encoders. **Journal of VLSI signal processing systems for signal, image and video technology**, v. 44, n. 1, p. 79–95, 2006.

KEUTZER, K. et al. System-level design: orthogonalization of concerns and platform-based design. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 19, n. 12, p. 1523–1543, 2000.

KIASARI, A. et al. A markovian performance model for networks-on-chip. In: EUROMICRO CONFERENCE ON PARALLEL, DISTRIBUTED AND NETWORK-BASED PROCESSING (PDP'08), 16., 2008, Toulouse, France. [S.l.: s.n.], **Proceedings**. New York: IEEE, 2008. p. 157–164.

KRISHNA, M.; GADRE, V.; DESAI, U. B. *Multifractal Based Network Traffic Modeling*. [S.l.]: Springer, 2003. ISBN 9781402075667.

LELAND, W. et al. On the self-similar nature of ethernet traffic (extended version). **IEEE/ACM Transactions on Networking**, v. 2, n. 1, p. 1–15, 1994.

LEUPERS, R.; TEMAM, O. *Processor and System-on-Chip Simulation*. [S.l.]: Springer, 2010. ISBN 9781441961754.

LIMA, A. B. *Contribuições à modelagem de teletráfego fractal*. Dissertation (PhD in Telecommunication Engineering) — Departamento de Engenharia de Telecomunicações, Universidade de São Paulo, 2008.

LU, Z.; YAO, Y.; JIANG, Y. Towards stochastic delay bound analysis for network-on-chip. In: IEEE/ACM INTERNATIONAL SYMPOSIUM ON NETWORKS-ON-CHIP (NOCS'14), 8., 2014, Ferrara, Italy. [S.l.: s.n.], **Proceedings**. New York: IEEE, 2014. p. 64–71.

MANDELBROT, B. B. *The Fractal Geometry of Nature*. [S.l.]: W. H. Freeman and Co, 1983. ISBN 0716711869.

MARCULESCU, R. On-chip networks: two sides of the same coin. **IEEE Transactions on Design and Test of Computers**, 2010.

MATLAB. *The Language of Technical Computing*. 2016. Available in: <https://www.mathworks.com/products/matlab/>. Accessed in: March 29, 2017.

_____. *Stateflow - Model and simulate decision logic using state machines and flow charts*. 2016. Available in: <https://www.mathworks.com/products/stateflow/>. Accessed in: March 29, 2017.

MELLO, A. et al. Parallel simulation of SystemC TLM 2.0 compliant MPSoC on SMP workstations. In: CONFERENCE ON DESIGN, AUTOMATION AND TEST IN EUROPE (DATE'10), 2010, Dresden, Germany. [S.l.: s.n.], **Proceedings**. New York: IEEE, 2010. p. 606–609.

MELLO, F. L. *Estudo e implementação de um gerador de tráfego com dependência de longa duração*. Dissertation (Masters in Electrical Engineering) — Departamento de Engenharia de Telecomunicações, Universidade de São Paulo, 2006.

MICHELI, G. D.; BENINI, L. *Networks on Chips: technology and tools (systems on silicon)*. [S.l.]: Morgan Kaufmann, 2006. ISBN 0123705215.

NICOPOULOS, C.; NARAYANAN, V.; DAS, C. R. *Network-on-Chip Architectures: a holistic design exploration: 45 (lecture notes in electrical engineering)*. [S.l.]: Springer, 2009. ISBN 978-90-481-3031-3.

PARISCHA, S.; DUTT, N. *On-Chip Communication Architectures: system on chip interconnect*. [S.l.]: Morgan Kaufmann Publishers Inc., 2008. ISBN 012373892X.

PAXSON, V.; FLOYD, S. Wide area traffic: the failure of poisson modeling. **IEEE/ACM Transactions on Networks**, IEEE Press, v. 3, n. 3, p. 226–244, 1995.

PEH, L.-S.; JERGER, N. E. *On-Chip Networks*. 1st. ed. [S.l.]: Morgan and Claypool Publishers, 2009. ISBN 9781598295849.

PEKKARINEN, E. et al. A set of traffic models for network-on-chip benchmarking. In: INTERNATIONAL SYMPOSIUM ON SYSTEM ON CHIP (ISSOC'13), 13., 2011, Tampere, Finland. [S.l.: s.n.], **Proceedings**. New York: IEEE, 2011. p. 78–81.

QIAN, Z. et al. A comprehensive and accurate latency model for network-on-chip performance analysis. In: ASIA AND SOUTH PACIFIC DESIGN AUTOMATION CONFERENCE (ASP-DAC'14), 19., 2014, Sun Tec, Singapore. [S.l.: s.n.], **Proceedings**. New York: IEEE, 2014. p. 323–328.

RIEDI, R. *Multifractal Wavelet Model source code*. 2016. Available in: <http://www.stat.rice.edu/~riedi/>. Accessed in: March 29, 2017.

RIEDI, R. H. et al. A multifractal wavelet model with application to network traffic. **IEEE Transactions on Information Theory**, v. 45, n. 3, p. 992–1018, 1999.

ROSETTACODE. *Huffman coding*. 2016. Available in: https://rosettacode.org/wiki/Huffman_coding#C. Accessed in: March 29, 2017.

SCHERRER, A.; FRABOULET, A.; RISSET, T. Long-range dependence and on-chip processor traffic. **Microprocessors and Microsystems**, v. 33, n. 1, p. 72–80, 2009.

_____. Automatic phase detection for stochastic on-chip traffic generation. In: INTERNATIONAL CONFERENCE ON HARDWARE/SOFTWARE CODESIGN AND SYSTEM SYNTHESIS (CODES+ISSS '06), 4., 2006, Seoul, Republic of Korea. [S.l.: s.n.], **Proceedings**. New York: IEEE, 2006. p. 88–93.

_____. A generic multi-phase on-chip traffic generation environment. In: IEEE INTERNATIONAL CONFERENCE ON APPLICATION-SPECIFIC SYSTEMS, ARCHITECTURES AND PROCESSORS (ASAP'06), 17., 2006, Samos, Greece. [S.l.: s.n.], **Proceedings**. New York: IEEE, 2006. p. 23–27.

SEPULVEDA, J. et al. The LRD traffic impact on the NoC-based SoCs. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEM DESIGN (SBCCI'10), 23., 2010, São Paulo, Brazil. [S.l.: s.n.], **Proceedings**. New York: IEEE, 2010. p. 97–102.

SHUMWAY, R. H.; STOFFER, D. S. *Time Series Analysis and Its Applications: with R examples*. [S.l.]: Springer Texts in Statistics, 2010. ISBN 9781441978653.

SIGMALAB. *MFT Application Code*. 2016. Available in: <<http://code.google.com/p/sigmav/>>. Accessed in: March 29, 2017.

SOCLIB. *The DSX/MJPEG platform*. 2016. Available in: <<https://www.soclib.fr/appliance/docs/dsx-mjpeg/index.html>>. Accessed in: March 29, 2017.

_____. *Installation Notes*. 2016. Available in: <<http://www.soclib.fr/trac/dev/wiki/InstallationNotes>>. Accessed in: March 29, 2017.

_____. *Mips Processor Functional Description*. 2016. Available in: <<http://www.soclib.fr/trac/dev/wiki/Component/Mips>>. Accessed in: March 29, 2017.

_____. *SoCLib Simulation Platform*. 2016. Available in: <<http://www.soclib.fr/>>. Accessed in: March 29, 2017.

_____. *Writing TLM2.0-compliant timed SystemC simulation models for SoCLib*. 2016. Available in: <<http://www.soclib.fr/trac/dev/wiki/WritingRules/Tlmt>>. Accessed in: March 29, 2017.

SOTERIOU, V.; WANG, H.; PEH, L. A statistical traffic model for on-chip interconnection networks. In: IEEE INTERNATIONAL SYMPOSIUM ON MODELING, ANALYSIS, AND SIMULATION (MASCOTS'14), 14., 2006, Monterey, CA, USA. [S.l.: s.n.], **Proceedings**. New York: IEEE, 2006. p. 104–116.

STENICO, J. W. de G. *Um novo modelo de tráfego multifractal aplicado a análise de desempenho de redes*. Dissertation (PhD in Telecommunication Engineering) — Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, 2014.

TAQQU, M. S.; TEVEROVSKY, V.; WILLINGER, W. Is network traffic self-similar or multifractal? **Fractals**, v. 05, n. 01, p. 63–73, 1997.

VARATKAR, G.; MARCULESCU, R. On-chip traffic modeling and synthesis for mpeg-2 video applications. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, v. 12, n. 1, p. 108–119, 2004.

VIAUD, E.; PECHEUX, F.; GREINER, A. An efficient TLM/T modeling and simulation environment based on conservative parallel discrete event principles. In: DESIGN, AUTOMATION AND TEST IN EUROPE CONFERENCE (DATE'06), 2006, Munich, Germany. [S.l.: s.n.], **Proceedings**. New York: IEEE, 2006. v. 1, p. 1–6.

WIKLUND, D.; SATHE, S.; LIU, D. Network on chip simulations for benchmarking. In: IEEE INTERNATIONAL WORKSHOP ON SYSTEM-ON-CHIP FOR REAL-TIME APPLICATIONS (IWSOC'04), 4., 2004, Banff, AB, Canada. [S.l.: s.n.], **Proceedings**. New York: IEEE, 2004. p. 269–274.

APPENDIX A – MULTIFRACTAL SPECTRA OF THE CA TRAFFIC PROFILES

Figure A.1: Multifractal spectra of the CA traffic profiles

(a) Huffman

(b) MJPEG

(c) MFT

APPENDIX B – COMPLIMENTARY PLOTS

Figure B.1: Huffman application time series: synthetic (50ns TLM capture aggregation window) vs. CA

Figure B.2: Huffman application time series: synthetic (100ns TLM capture aggregation window) vs. CA

Figure B.3: Huffman application time series: synthetic (200ns TLM capture aggregation window) vs. CA

(a) Autocorrelation function

(b) Periodogram

(c) Abry-Veicht spectrum

(d) Multifractal spectrum

Figure B.4: MJPEG application time series: synthetic (50ns TLM capture aggregation window) vs. CA

Figure B.5: MJPEG application time series: synthetic (100ns TLM capture aggregation window) vs. CA

Figure B.6: MJPEG application time series: synthetic (200ns TLM capture aggregation window) vs. CA

(a) Autocorrelation function

(b) Periodogram

(c) Abry-Veicht spectrum

(d) Multifractal spectrum

Figure B.7: MFT application time series: synthetic (50ns TLM capture aggregation window) vs. CA

Figure B.8: MFT application time series: synthetic (100ns TLM capture aggregation window) vs. CA

(a) Autocorrelation function

(b) Periodogram

(c) Abry-Veicht spectrum

(d) Multifractal spectrum

Figure B.9: MFT application time series: synthetic (200ns TLM capture aggregation window) vs. CA