

ESL Design Methodology for Architecture Exploration using OSCI TLM-2.0 Standard

Mahshid Sedghi, Fatemeh Javaheri, Nastaran Nemati, and Zainalabedin Navabi

Electrical and Computer Engineering

School of Engineering Colleges, Campus 2, University of Tehran, 1450 North Amirabad, 14395-515 Tehran, Iran

{mahshid, n-javaheri, nastaran, navabi}@cad.ece.ut.ac.ir

Abstract— We present a methodology for design of complex digital systems at system level through a training video. After the introduction to SystemC and TLM-2.0, the video explains a number of design sub-levels and guidelines for design at each sub-level. A test data compression system is implemented as a case study using OSCI TLM 2.0 standard.

I. INTRODUCTION

Recent advances in semiconductor technology have enabled designers to integrate hundreds of embedded processors on a single chip. This fast-paced increasing complexity of digital systems, on the other hand, has introduced new challenges to the design community. An important challenge is to handle the complexity of designing such systems with increased productivity and decreased time-to-market. Electronic System Level (ESL) design methodologies solve this problem by starting the design from higher abstraction levels than RTL. Transaction Level Modeling (TLM) has been proposed as the key step toward the ESL methodology. In the TLM point of view, a system is divided into computation and communication parts. The focus of the TLM is on communication and this is performed through passing high level data structures called transactions between computation parts.

An ESL design methodology is presented in this work. We refer to the design methodology as the set of design guidelines and rules which help the designer in implementation of the design. While there are widely used well established RTL design methodologies, for design at ESL there is no solid design methodology. For example, for an RT level system, the straightforward approach is a datapath/controller partitioning and implementing each part separately. In contrast to RTL, an ESL designer has a high degree of freedom in design choices. A variety of architectures can be devised for a single system at ESL. The lack of solid ESL design methodologies results in increased time-to-market, decreased performance and difficulty of IP-reuse. In this work, we will present an ESL design methodology which contains several sub-levels. A Test Data Compression system (TDC) is used as the case study for this methodology and is implemented at each sub-level using OSCI TLM 2.0 standard.

II. ESL DESIGN METHODOLOGY

Our ESL design methodology consists of three design sub-levels. The test data compression used as a case study consists of four processing elements (PE). The first PE (Writer) reads test data from an ATE memory and passes it on to the next PE. The next two PEs perform MTF and Run-Length compression on the data received. Finally, Reader reads the compressed test data and writes it to the

ATE memory. We describe three levels for any ESL design and implementation methodology. These levels are discussed below.

A. Level Zero

Figure 1 shows the system architecture at level zero. This architecture is sequential and is specifically appropriate for verification purposes. Issues like memory architecture and optimization are not the concern of the designer. This architecture is implemented using TLM 2.0 blocking transport interface.

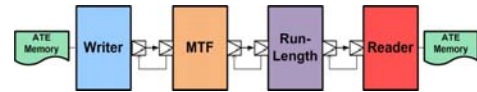


Figure 1 TDC architecture 1

B. Level One

Figure 2 and 3 show two different architectures at level one. The most important characteristic of this level is the memory architecture. In architecture 2, a central memory is accessed by different PEs through a single port, while in architecture 3 each PE has individual access to the central memory. These architectures are implemented using TLM 2.0 blocking transport interface and are sequential.

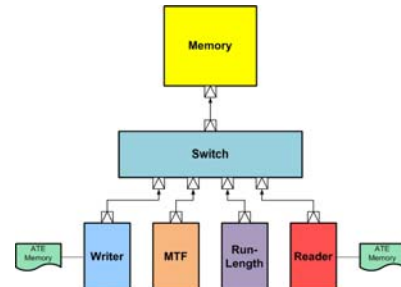


Figure 2 TDC architecture 2

C. Level Two

In level two, the main focus is on design optimization. Concurrency as an optimization paradigm is considered in this level. In order to exploit concurrency, we implement pipelining using TLM 2.0 non-blocking transport interface. Pipelined versions of architectures 2 and 3 are implemented in this level.

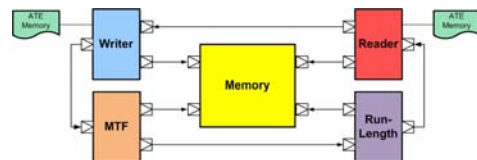


Figure 3 TDC architecture 3