Modeling of Communication Protocols in VHDL

Ali Assi and Bozena Kaminska
Ecole Polytechnique of the University of Montreal
P.O. Box 6079, Station “Centre-ville”, Montreal, PQ, Canada, H3C 3A7

Abstract
As synthesis tools become more advanced and reliable, the entry point for the designer in the design process is moving towards higher levels of specification. In this paper, the feasibility of using VHDL language to model communication protocols is examined, and a modeling methodology for such systems using VHDL is presented. We demonstrate our methodology on the transport protocol ISO/CCITT class 0. The resulted model is an appropriate model for the high level design [1], which mainly includes the steps shown by figure 1. The paper concludes with a discussion of the importance and future trends in modeling communication protocols using HDLs.

1. Introduction

Rapid advances in transmission technology and networking have resulted in development of hardware implementations of several communication protocols [2]. Communication processing has to be done faster than at any time in the past. Such high speeds can be achieved by the use of specialized VLSI circuits [3]. If communication processing consumes a significant part of processing power of a large host computer, then it is very expensive. If it can be performed by a relatively inexpensive VLSI circuit, then the overall system cost will be lower. One of the potential problems of implementing protocols in hardware is the difficulty in changing its design when the underlying protocol is changed. Most of the new VLSI circuits for communication processing are at least partially programmable [3-4].

Hardware description languages like VHDL are attracting more and more interest from the hardware design community. The most interesting application fields are simulation and synthesis based on VHDL. In combination with synthesis tools, VHDL allows hardware to be designed at a high level abstraction. The rich descriptive capabilities of VHDL [5-6-7-8] and the algorithmic power of communication protocols [9-10] complement each other. Telecommunications has been gaining in importance recently. At the same time, VHDL has become an international standard for the design of digital systems. In spite of this, the literature lacks reports on the use of VHDL to model communication protocols. This paper surveys some of the techniques VHDL can provide for communication protocols implementers.

Here, we mainly define a modeling methodology for communication protocols, aiming to have a synthesizable model, which leads, using the synthesis tools, to a hardware implementation of the protocol. We encountered no reference in literature regarding the behavior modeling of the communication protocols using the VHDL language. The specification found in literature are written in languages dedicated to the specification of communication protocols, we name here three of them: Estelle, LOTOS and SDL. These languages are usually used for the development and testing of software protocol implementations [4].

![Figure 1. High-Level Design](image)

Note however, that to achieve the possibility of efficient VLSI implementation, the development of protocol modeling methodology using HDLs (VHDL in particular) is essential, in order to increase the overall quality of the final products. The presented methodology is mainly founded on the expressive power of VHDL.

The remainder of this paper is organized as follows. Section 2 provides an overview of the description techniques in VHDL, which explore the match between VHDL language and communication protocols rules. Next, in section 3 the proposed modeling methodology is described. Section 4 contains the modeling of the ISO/CCITT Transport protocol class 0 (TP0) as a prototype example. The paper concludes by identifying the main results of our work thus far and trends which might guide future work in this area.

2. Description Techniques in VHDL

VHDL is an IEEE standard intended to be used for the development, synthesis, verification and documentation of hardware designs. It has the capability of specifying a system in terms of behavior, dataflow or structure or any combination thereof [8].

Communication circuits are often easily thought of as a set of concurrent communicating modules. Consequently, a specification language must have constructs to represent notions as concurrency, communication, parallelism, non-determinism and other characteristics of the communication protocols [9-10]. To specify the OSI communication protocols, the VHDL language must have the capability of expressing these notions. The description method in VHDL is based on the concepts of “component” and “port”. A component is an entity of specification which executes a given task, and which includes several concurrent activities. A port is used by a component to
communicate with its environment (the other components of the system). In this section we discuss the ability of the VHDL language to describe the communication protocols.

Parallelism
Parallelism refers to the specification of concurrency. In VHDL it is expressed implicitly through concurrent instructions which are executed within an architecture. Figure 2 illustrates this notion. The set of instructions describing the functionality of this architecture is being executed in parallel.

```
ENTITY entity_name IS
  PORT ( ... );
END entity_name;
ARCHITECTURE architecture_name OF entity_name IS
BEGIN
  ・ concurrente_instruction1;
  ・ concurrente_instruction2;
  ・ concurrente_instruction3;
END architecture_name;
```

Figure 2: Expression of the parallelism in VHDL

Non-Determinism
One of the fundamental concepts of communication protocols is the Non-Determinism, which refers to the specification of the Non-Determined comportment of the protocol. This notion in VHDL is expressed through conditional assignments, and figure 3 illustrates it.

```
ENTITY entity_name IS
  PORT ( ... );
END entity_name;
ARCHITECTURE architecture_name OF entity_name IS
BEGIN
  ・ comportment1 WHEN condition1 ELSE
  ・ comportment2 WHEN condition2 ELSE
  ・ comportment3 WHEN condition3 ELSE
  ・ comportment4;
END architecture_name;
```

Figure 3: Expression of the non-determinism in VHDL

Communication
The notion of communication in VHDL is expressed through `WAIT ON` instruction within a process. Figure 4 illustrates the communication between two VHDL entities: entity_name1 and entity_name2. These two entities communicate through a port “signal” which is declared within each entity. Thanks to the instruction `WAIT ON`, the entity_name2 wait for a change on “signal” produced by entity_name1, to execute the process instructions.

Timing
In VHDL, timing specification takes one of two forms:
1. After clauses which specify the time into the future when the value of the signal is to be updated with the new value (e.g. `X <= value AFTER delay`).
2. Timeout clauses which specify the maximum time to be spent at the wait statement (e.g. `WAIT ON X FOR Delay`).

3. Modeling Methodology
Having introduced the description techniques in VHDL language, and since those techniques cover most of the communication protocol concepts and functions [9], we now examine in details the proposed methodology for modeling communication protocols in VHDL. The main steps involved in this methodology are:

```
ENTITY entity_name1 IS
  PORT ( ... );
  ・ signal : OUT signal_type;
END entity_name1;
ARCHITECTURE architecture_name1 OF entity_name1 IS
BEGIN
  ・ signal <= expression;
END architecture_name1;
```

```
ENTITY entity_name2 IS
  PORT ( ... );
  ・ signal : IN signal_type;
END entity_name2;
ARCHITECTURE architecture_name2 OF entity_name2 IS
BEGIN
  PROCESS process_name;
  BEGIN
    WAIT ON signal FOR delay;
    X <= signal AFTER delay;
  END process_name;
END architecture_name2;
```

Figure 4: Expression of the communication in VHDL

Partitioning of the protocol into Communicating Entities (Step 1)
Using an English language specification of the protocol [5-9], we define the interfaces which are known as the service access points, and represent either service requests from the higher layer and responses to these requests, or services furnished by the lower layer and service requests to this layer. In VHDL these service access points correspond to the “port” notion of the language, and are declared within an entity. A thorough analysis of those functions that the protocol must realize, will allows us to define the entities (figure 5) it constitute. These functions correspond to the notion “entity” of the VHDL language.

```
ENTITY entity_name1 IS
  PORT ( typed ports list);
END entity_name1;
ENTITY entity_name2 IS
  PORT ( typed ports list);
END entity_name2;
ENTITY entity_name3 IS
  PORT ( typed ports list);
END entity_name3;
```

Figure 5: Partitioning of the protocol into communicating entities

Programming of Communicating Entities (Step 2)
In VHDL, the programming procedure of communicating
component is done within the body of the entity (architecture) previously declared (figure 6). We use PROCESS notion to describe the sequential behavior of the component. We define the specific variables of the protocol (informations to be emitted on the medium) and other variables that are specific to the functionality of the entity. The architecture reveals both the structure and the functionality of the entity.

4. Example: Transport Protocol ISO/CCITT class 0 (TP0)

In order to illustrate the usefulness of our modeling methodology we will present a modeling example: the ISO/CCITT transport protocol class 0 (TP0) [5-9].

![Figure 8. Seven-layer OSI architecture](image)

Overview of the TP0

Among the seven-layer OSI architecture illustrated in figure 8, the purpose of the transport layer is to provide a transparent, reliable, end-to-end data transfer mechanism for the session layer and other layers (the users) above [9]. It uses the services of the network layer below to help in this function but shields the upper layers from the details of the network connections and types of networks used (figure 9). To handle both the different types of data transfer that might be expected over a transport connection and the wide variety of networks that might be available to provide network services, five classes have been defined for the ISO transport protocol. These are labeled classes 0, 1, 2, 3, and 4 [9]. Class 0, for example, is the simplest type of transport connection, with a minimum of functions defined. In this class the transport protocol is required only to set up a simple end-to-end transport connection and, in the data-transfer phase, to have the capability of segmenting data messages if necessary. It has no provision for recovering from errors and cannot multiplex several transport connections onto a single network connection.

The TP0 may be considered as a set of functions accomplished by nine entities. These entities communicate with each others and with the related layers using the appropriate ports (figure 10).

Specification of module1

The VHDL entity module1 communicates with its environment using the ports shown by figure 11.
The VHDL code of the nine modules with the global program, are obtained and simulated using Mentor Graphics VHDL simulator [12].

Protocol Functions Not Described
The verification of: TCONreq primitive validity, network connection availability and the possibility of using the current network connection are not specified in this paper.

5. Conclusion
We have discussed a new methodology to model the communication protocols, using VHDL language. A complete VHDL model of the ISO/CCITT transport protocol class 0 (TP0) has been presented. Simulation results [12] demonstrate the feasibility and usefulness of our methodology. This work could be expanded in the future to cover more of the concepts involved in developing communication protocols. In addition, work in the area of hardware/software co-development. Our experience here has given us confidence that further work in the area will benefit communication protocols engineering and extend the application domain of VHDL.

Our next job, will be the synthesis of the resulting model with the commercial synthesis tools like synopsys or Mentor Graphics, to obtain a hardware implementation of the protocol, or, in order to enhance the protocol efficiency, implement only some modules of the protocol in hardware and keep the rest in software (Codeign).

References