Università degli Studi di Verona, Facoltà di Scienze MM. FF. NN.
Dipartimento di Informatica
EDALab: Embedded System Design Center

Ca' Vignal, Strada Le Grazie 15
37134 Verona, Italia
Tel. +39 045 8027069
Fax +39 045 8027068

Verona, 01/03/2007

Modeling and Simulating FSMDs in VHDL

Nicola Bombieri

Ca' Vignal, Strada Le Grazie 15
37134 Verona, Italia
Tel. +39 045 8027069
Fax +39 045 8027068

1 Required Background

Students interested in learning HDL modeling and simulation are required to know the fundamentals of Finite State Machine (FSM) and Finite State Machine with Datapath (FSMD) model of computations. Moreover, it is advisable that students are familiar with at least one Hardware Description Language such as VHDL or Verilog.

2 Goal

The goal of this lecture consists of describing the basic concepts related to the use of two of the main important commercial tools to model and simulate electronic circuits.

Students will learn to:

- modeling a digital device by using the HDL Designer starting from a set of high level specifications;
- simulate the behavior of the digital device in order to dynamically check the implementation correctness.

3 Introduction

This lecture shows how to create digital devices at Register Transfer Level (RTL) by using block diagrams, control blocks described as a hierarchical state machine and re-used component described by a HDL text view. This implementation task is a fundamental step into an embedded system design flow as depicted in Figure 1. In facts, moving from Transaction Level (TL) towards RTL requires designers to translate the high level system design to a more refined one, by adding implementation details dependent on the architectural choices [1]. Since the refinement process can not be automatic it is duty of designers so far.

Ca' Vignal, Strada Le Grazie 15
37134 Verona, Italia
Tel. +39 045 8027069
Fax +39 045 8027068

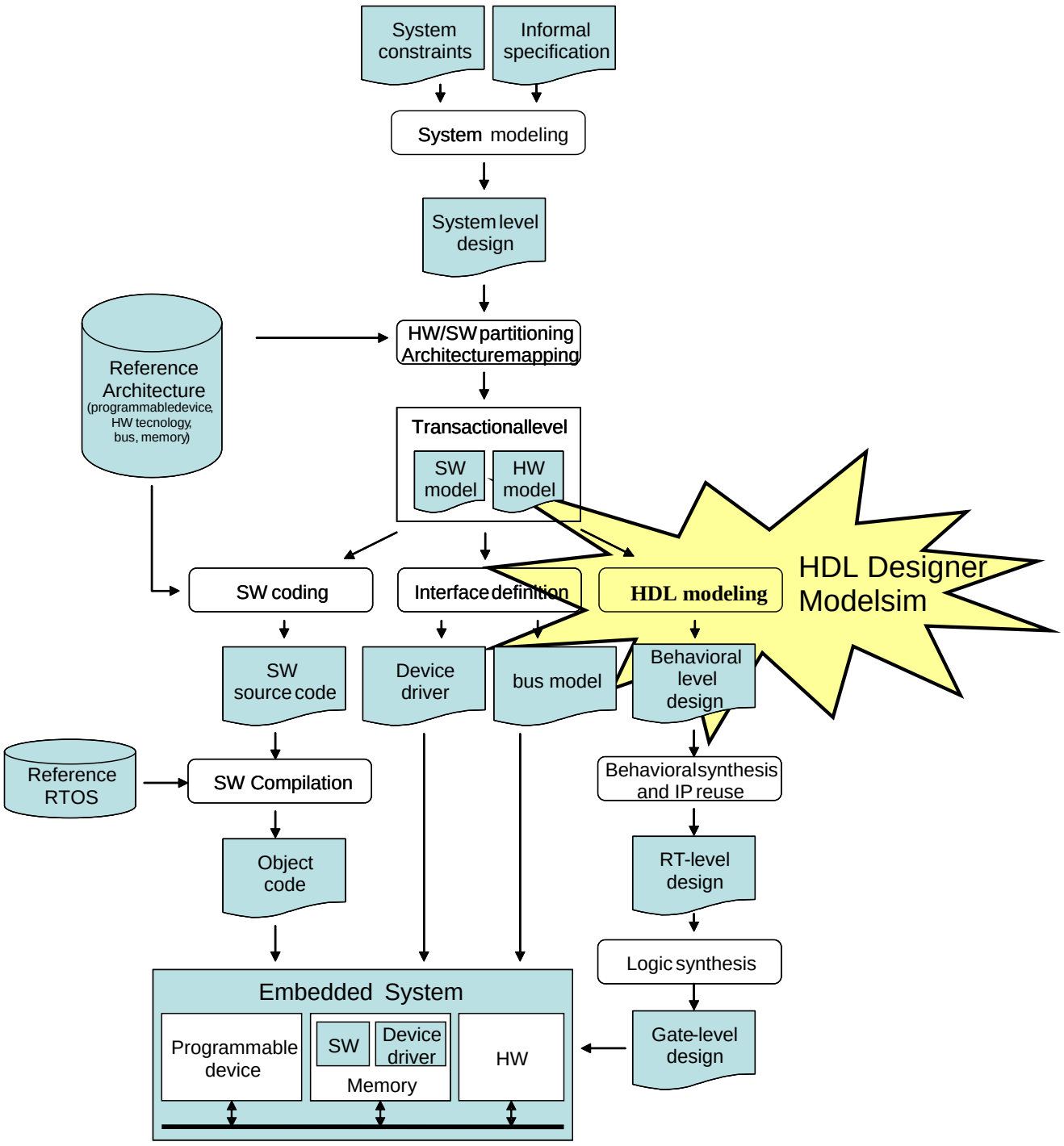


Figure 1: Embedded system design flow.

Several commercial tools aid designers in this task. HDL Designer by Mentor Graphics is one of the most widespread tool and it will be presented in this lecture.

Besides the implementation purpose, designers exploit HDL Designer to create test benches by using a flow chart. Then, the flow chart can be used as a test harness to simulate the generated VHDL for the digital device. Finally, the simulation results can be displayed as animation on the flow chart and state machine to assist in debugging the design. The design, hence, can be considered ready to the synthesis phase.

4 HDL Designer

A practical and complete tutorial for the use of HDL Designer is available in [2]. Only the key concepts of its use will be presented in this paper.

HDL Designer allows to implement the system design by using different description models, such as block diagrams, state diagrams, truth tables, component reuse, and flow charts. Block and state diagrams will be introduced as the most important model to represent FSMDs.

4.1 Block Diagram

The block diagram model is used to represent the system device by a set of communicating components. Figure 2 shows an example. The system is represented as a top level module, defining I/O ports and every block composing the device.

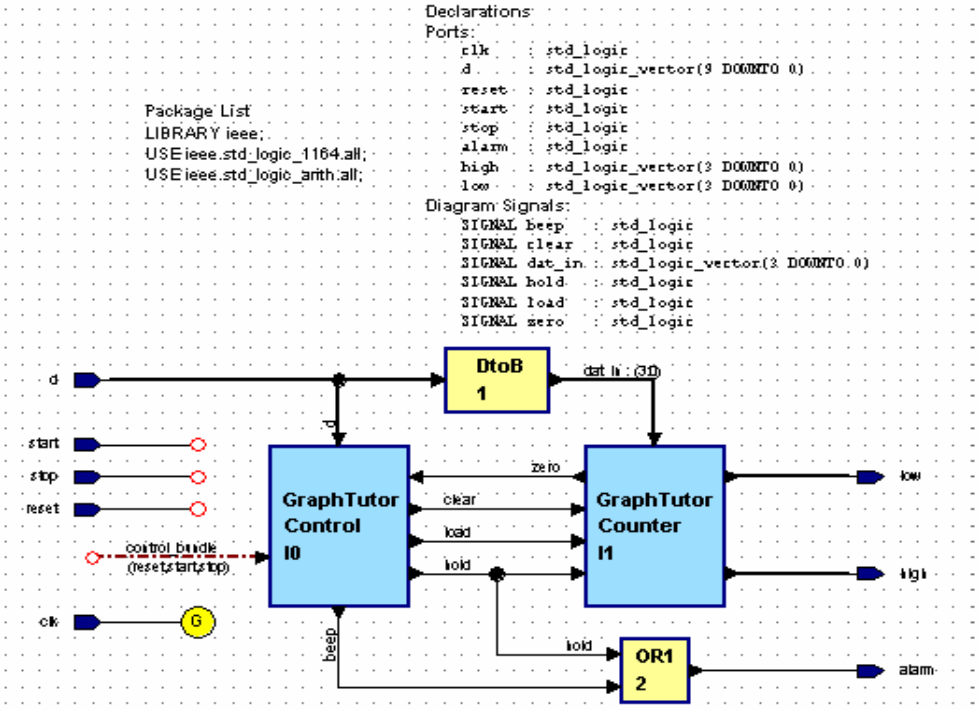


Figure 2: Example of block diagram.

Ca' Vignal, Strada Le Grazie 15
37134 Verona, Italia
Tel. +39 045 8027069
Fax +39 045 8027068

All the components appear into the diagram as black boxes. At this level, the designer just takes care about the components connection.

4.2 State Diagram

State diagram is the most important design model in our context since it allows to graphically implement the device behaviour by using the Finite State Machine model. The diagram is created by adding state elements (specifying the start state), transitions between states, and defining both conditions and actions for each transition. An example of state diagram created by HDL Designer is showed in Figure 3.

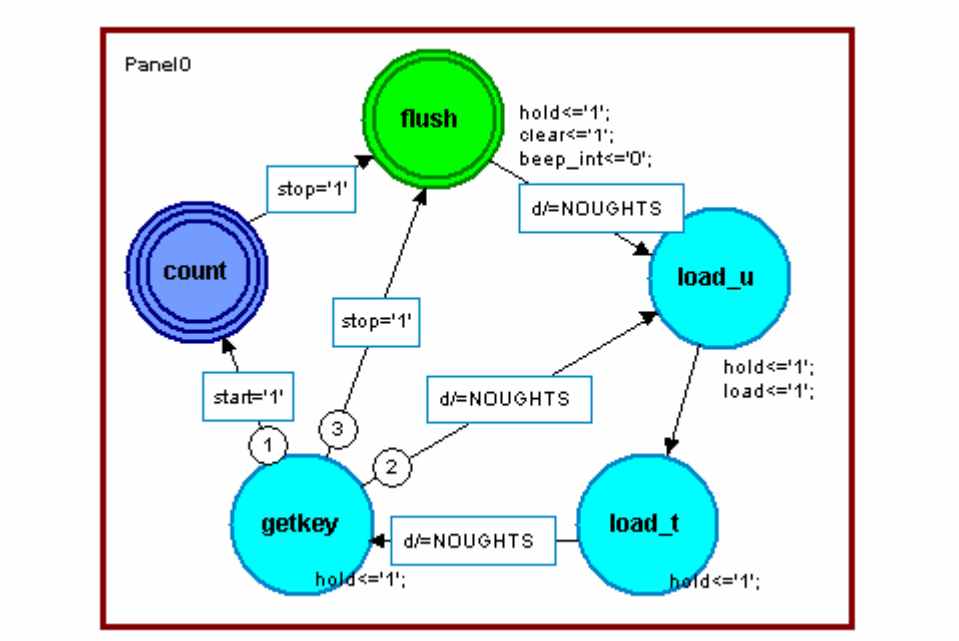


Figure 3: Example of state diagram.

The *State Machine Properties* feature provides designers to set HDL generation characteristics and the state machine encoding. For example, once the FSM is completed, designers can choose between synchronous and asynchronous behaviour, clock and reset types, and the recovery state. Finally the designed FSM can be translated into a HDL code by choosing the HDL style (i.e., 2 or 3 processes, IF or CASE statements).

The HDL code generation is monitored in a log window which includes any error and warning issued during the generation process. If any error occurs, the designer can display the source graphics corresponding to the errors. If there are no errors, the designer can proceed to the test bench generation.

4.3 Test bench generation

Once the RTL representation has been generated, the designer can create a test bench module to check the behaviour correctness of the device. HDL Designer firstly generates the

Ca' Vignal, Strada Le Grazie 15
37134 Verona, Italia
Tel. +39 045 8027069
Fax +39 045 8027068

tester block and then it connects the tester to the device, generating an ad-hoc test bench. Figure 4 shows an example of test bench generated for a timer component.

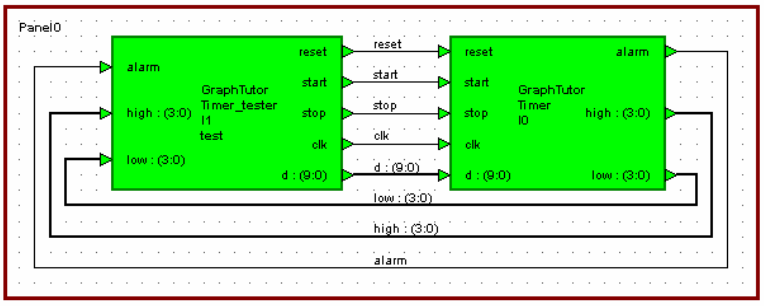


Figure 4: Example of test bench component.

The matching nets on the two components are implicitly connected by name. However, the designer may like to connect them explicitly by choosing the *Autoconnect* process.

The created test bench containing the device and the stimuli generator modules can finally be translated into HDL code, in order to undergo the analysis phase by using Modelsim.

5 ModelSim

Modelsim belongs to the suite of Mentor Graphics, with together HDL Designer and other tools for the synthesis task (i.e., Leonardo). During the generation task of the HDL description, all the necessary to simulate and visualize the device is automatically set by HDL Designer. Thus, the designer can easily analyse how the device under test react to the stimuli on the input port, by checking the waves of the output ports and signals. Figure 5 shows an example of visualization of any port and signal values throughout the time scale.

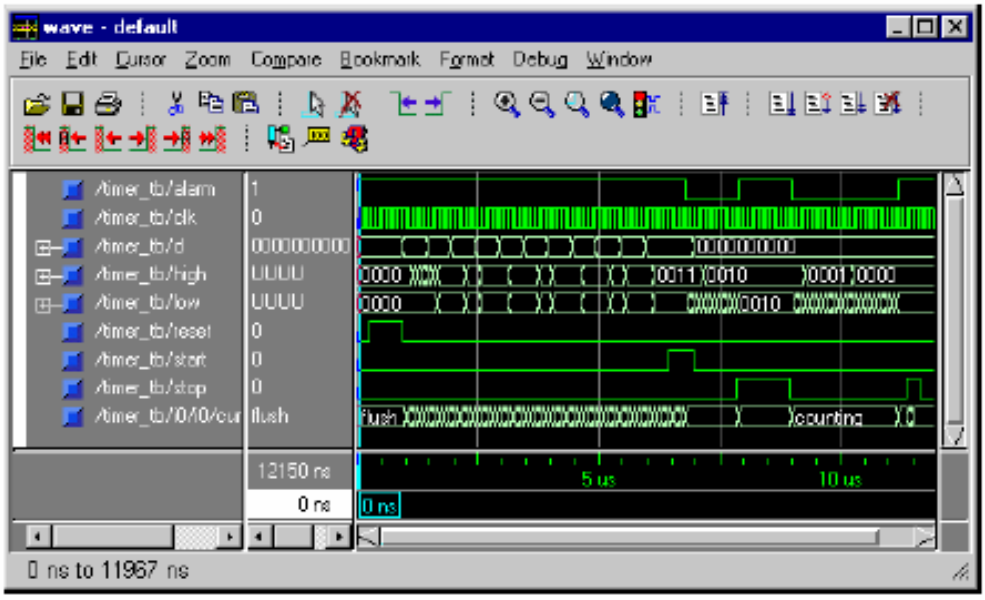


Figure 5: Example of waveforms visualization.

Ca' Vignal, Strada Le Grazie 15
37134 Verona, Italia
Tel. +39 045 8027069
Fax +39 045 8027068

5.1 Manual stimuli generation

When the test bench generation is not feasible, the designer can analyse the device by manually generating the input stimuli. That is, he can force arbitrary values on the input ports, observing the waveforms of the output ports over time. The necessary steps before proceeding are the following:

1. *Work folder creation*. The simulator compiles the VHDL models and holds them into the folder named *work*. For any Entity, a sub folder is created into *work*, wich contains the compiled version of the device. To create the folder (once for all), run:

   ```
   qhlib work
   ```

2. *VHDL files compilation.* To compile the device under test (*namefile.vhd* or *namefile.vhdl*) run:

   ```
   qvhcom -93 namefile.vhdl
   ```

   If the execution produces the following result, then the compilation phase has been successfully completed:

   ```
   Model Technology ModelSim SE vcom 6.1b Compiler 2005.09 Sep  8
   2005
   -- Loading package standard
   -- Loading package std_logic_1164
   -- Loading package std_logic_arith
   -- Loading package std_logic_signed
   -- Compiling entity adpcm_vhdl
   -- Compiling architecture rtl of adpcm_vhdl
   ```

3. *Modelsim run*. By editing:

   ```
   qhsim
   ```

To manually force the stimuli on the input ports, use the `force` command with the following syntax:

```
force <port_name/signal> <value> <time>
```

For example, the following statement will force to 4 the input *x* in 10 simulation time units:

```
force x 00000100 10
```

In order to automate the application of the input stimuli, it is possible to insert the `force` statements into a script file (i.e., *file.do*). Then, to execute the script, run:

```
force <file.do>
```

Finally, the simulation can start by editing the command:

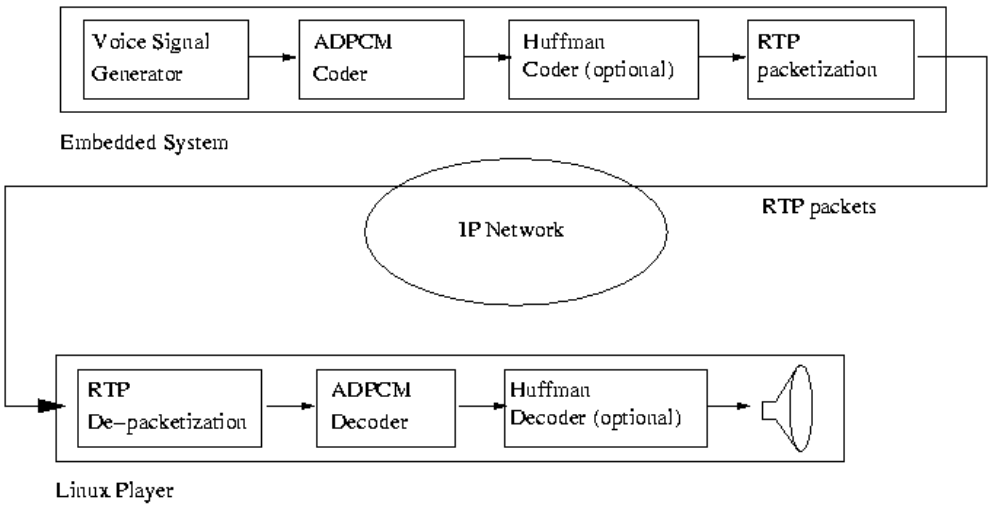
```
run <simulation_time>
```

Ca' Vignal, Strada Le Grazie 15
37134 Verona, Italia
Tel. +39 045 8027069
Fax +39 045 8027068

6 Case Study: V-CLIP

V-CLIP (Voice Client Over IP) is a multimedia embedded system for transmitting voice over IP (VOIP) [1]. It consists of:

- Voice Signal Generator: it is used to generate voice data to verify the correctness of the application.
- ADPCM Coder: it performs 4:1 data compression by exploiting a simple Adaptive Differential Pulse Code Modulation algorithm.
- Entropy Coder: it further improves data compression by encoding the most frequent input symbols with fewer bits.
- RTP Packet Generator: it performs data packeting in according to the Real-Time Transport Protocol and it sends data over the network.



As the ADPCM coder is the module that will become hardware, it gains the central focus of this lecture. Students will implement it in VHDL code, by exploiting the tools previously presented, in the following steps:

1. FSMD realization by means of HDL Designer, starting from the high level C code as specification (http://profs.sci.univr.it/~bombieri/V-CLIP).
2. Automatic testbench generation.
3. Dynamic verification by simulation, by means of Simulink.

Ca' Vignal, Strada Le Grazie 15
37134 Verona, Italia
Tel. +39 045 8027069
Fax +39 045 8027068

7 References

[1] EDALab "V-CLIP: A Voice-Client over IP", http://www.edalab.net.
[2] Mentor Graphics "*Graphical Design Tutorial for HDL Author and HDL Designer*", Software Version 2004.1, 12 March 2004.
(http://profs.sci.univr.it/~bombieri/HDL/tutorial.pdf)