SYNOPSYS DESIGN SYSTEM IMPLEMENTED IN EDUCATION

Peter Trifonov Goranov, Associated Professor, Ph.D.,
Marin Hristov Hristov, Associated Professor, Ph.D.,
Mariana Evstatieva Goranova, Assistant Professor, Ph.D.,
Anthony Trifinov Trifonov, Ph.D. Student
Technical University of Sofia, Bulgaria

This paper provides student to complete IC design process. SYNOPSYS Design System offers very powerful tools to teach the general digital system design principles required as functional design. A simple example is included. The design is expressed at the register-transfer level (RTL) and then automatically is converted to a gate-level. Students can explore a wide range of architectural and functional alternatives and choose the best one. Thus students use the general design principles and can apply them to real industry.

1. INTRODUCTION

For the past few years CAD systems have occupied a central position in the VLSI production. Universities include courses covering the analysis, design, application, and fabrication of IC. The engineering education has to be supported by well-developed design tools.

The objective of this paper is to describe the complete design aspects of a full custom design. The simple example contains common used blocks as: memory, comparator, counter and finite state machine. The overall emphasis has been to perform mixed-level simulation - some blocks (comparator, counter, finite state machine) are realised at the gate level, but the memory is remained at the RTL. For the RTL description we use the hardware description language VHDL which is the leader in the available commercial systems.

2. FUNCTIONAL DESIGN PROCESS

The typical high-level design process starts with the specification level of the IC and continues with design entry, validation, implementation, and verification.[1]

2.1. SPECIFICATION

A simple example illustrates the main steps in a typical functional design process and integrates counting and control logic. The circuit reads from a 16x8 ROM. The 8-bit word has two control bits and six data bits. The combination 11 of the control bits reverses the counting, otherwise the counting continues in the same direction. The counting has to be reversed in case of overloading in positive and negative counting direction. This synchronous circuit has a clock signal of 200 ns.

2.2. DESIGN ENTRY

This step involves the design description as schematic diagrams and register-transfer level HDL code. RTL description uses the VHDL language.

Fig. 1 shows one possible decision of the design hierarchy. The example is organised in a two-level hierarchy. The top-level entity, **EXAM**, consists of three blocks. The block **CONTROL** is partitioned into two subblocks. SYNOPSYS creates three files for each block: `.sch` (schematic description), `.sym` (symbol description), `.vhd` (VHDL description) and a file `.tre` describing design hierarchy. Fig. 2 displays the block diagram of the example.

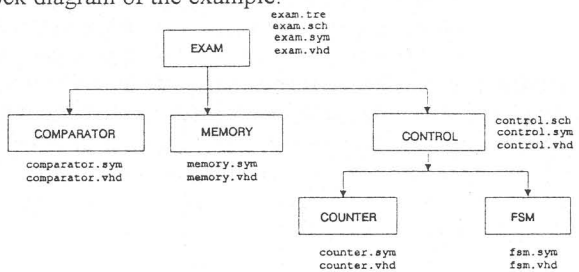

Fig. 1. Circuit design hierarchy

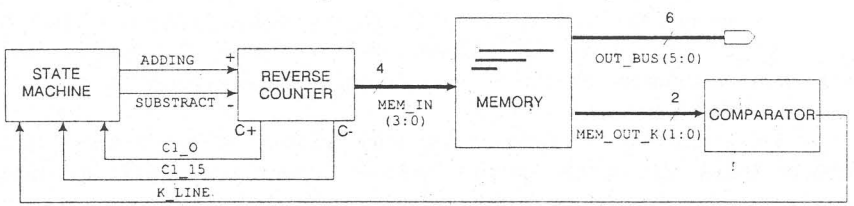

Fig. 2. Circuit block diagram

The first step in the Design Entry phase is to create a schematic **exam.sch** for the top-level design entity using *Schematic Editor*. Students create symbols for the functional blocks **COMPARATOR**, **MEMORY**, **CONTROL** with the *Symbol Editor*; connect symbols with wires; add I/O markers for the ports; and put attributes for the pins, wires, and symbols. For this top-level entity they create a symbol for **EXAM**.

The second step is to create a schematic for the lower level schematic. For the example, students create a schematic **control.sch** for the block **CONTROL** which consists of symbols **COUNTER** and **FSM** (Finite State Machine). When the schematic is completed they use the *Hierarchy Navigator* to link schematics and symbols into a single hierarchy **exam.tre** for **EXAM**.

The next step is to generate structural VHDL code using the *VHDL Netlister*. This step includes the following parts:

- Library and use statements
- An entity declaration

- Architecture statements
  - Complete architecture named `SCHEMATIC` for symbols with schematics associated for high-level design
  - Architecture template named `BEHAVIORAL` for symbols without schematics for the bottom of the design
  - Testbench architecture template
- Configuration statement

Students add VHDL code in the user defined sections with the `BEHAVIORAL` description for each symbol for the bottom of the design (Fig. 3). The file `tb_exam.vhd` is a testbench file which contains the entire design as a unit under test (`UUT`) and a block `TB` where students can put process to drive the input ports and read the output ports (Fig. 3a). SYNOPSYS permits very flexible way to input test signals. In this file students define the clock of the circuit. `for_train` package has to be available in the testbench because the package contains declaration and required functions definitions.

```
  TB : block
  begin
   CLOCK_CLK : PROCESS
     begin
CLK <= '1';
CLK <= transport '0' after CLK1_TIME/2;
wait for CLK1_TIME;
   end process;
  end block;
-- *** End Test Bench - User Defined Section
***
```

Fig.3a. The Part of the VHDL `Test Bench`

```
architecture BEHAVIORAL of COMPARATOR is
  begin
work:process(K_IN)
begin
case K_IN is
 when "11"  => K_OUT <='1';
 when others => K_OUT <='0';
end case;
end process;
end BEHAVIORAL;
```

Fig.3b. The Part of the VHDL `COMPARATOR`

```
architecture BEHAVIORAL of FSM is
type STATE_TYPE is (S0,S1);
signal CURRENT_STATE, NEXT_STATE:STATE_TYPE;
  begin
WORK:
 process(CURRENT_STATE, CAR_0, CAR_15, K_IN)
begin
NEXT_STATE<=CURRENT_STATE;
 case CURRENT_STATE is
  when S0 => C_UP<='1'; C_DO<='0';
     if K_IN='1' or CAR_15='1' then
        C_UP<='0'; C_DO<='1'; NEXT_STATE<=S1;
     end if;
  when S1 => C_UP<='0'; C_DO<='1';
     if K_IN='1' or CAR_0='1' then
        C_UP<='1'; C_DO<='0'; NEXT_STATE<=S0;
     end if;
 end case;
end process;
SYNCH:process
begin
wait until CLK'event and CLK='1';
CURRENT_STATE<=NEXT_STATE;
end process;
end BEHAVIORAL;
```

Fig.3c. The Part of the VHDL `FSM`

```
architecture BEHAVIORAL of COUNTER is
  begin
SYNCH : PROCESS
variable countup,countdo:bit_vector(3 downto 0);
begin
wait until CLK'event and CLK='1';
   if (ADD='1' and SUB='0') then
   countup := f_ADD(countup);countdo := "1111";
   CAR_0 <= '0';CAR_15 <= '0';
   if countup="1111" then
    C_OUT<="1111"; CAR_0 <= '0';
    CAR_15 <= '1';
   end if;
   C_OUT<=  countup;
   elsif (ADD='0' and SUB='1') then
     countdo := f_SUB(countdo);
     countup := "0000";
     CAR_0 <= '0';CAR_15 <= '0';
    if countdo="0000" then
      C_OUT<="0000"; CAR_0 <= '1';
      CAR_15 <= '0';
    end if;
   C_OUT<=  countdo;
   end if; end process;
end BEHAVIORAL;
```

Fig.3d. The Part of the VHDL COUNTER

```
architecture BEHAVIORAL of MEMORY is
type MEMORY_ARRAY is array (0 to 15)
        of bit_vector(7 downto 0);
  begin
READ:process(MEM_IN)
variable mem : MEMORY_ARRAY :=("00000000",
"00000101","00010010","00100110","00111001",
"01001010","01011001","01100110","01110001",
"00000000","11111101","10010010","00100110",
"10111001","01111010","01011001");
variable count : INTEGER;
variable r_vect : bit_vector(7 downto 0);
begin
  count :=Bit_vector_to_integer(MEM_IN);
  r_vect := mem(count);
  MEM_OUT    <= r_vect(7 downto 2);
  MEM_OUT_K <= r_vect(1 downto 0);
end process;
end BEHAVIORAL;
```

Fig.3e. The Part of the VHDL MEMORY

2.3. VALIDATE THE DESIGN

The generated VHDL description is simulated using the VSS (*VHDL System Simulator*) Family.

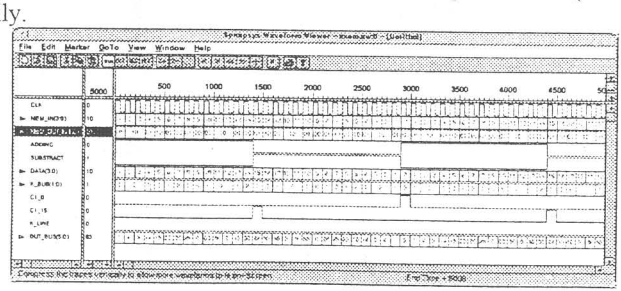

Fig.4. Signals in the Waveform Viewer

This phase includes two steps: translating the VHDL code to an intermediate representation using *VHDL Analyzer*, and carrying out the simulation using *VHDL Simulator*. First students analyse the testbench package `for_train` manually because it is not part of the design hierarchy. After that they analyse the design. SYNOPSYS provides a tool to simplify later re-analysis using *simdependens* which creates a *makefile*. When students simulate the design, they use the *VHDL Debugger*

to write command files that execute the process, put break points, trace signals. The simulator generates signal waveforms in the *Waveform Viewer* (Fig. 4). If the design doesn't work, students can modify VHDL code, re-analyse and restart the simulation, or return to Design Entry.

2.4. IMPLEMENTATION

The synthesis is the process of translating the RTL design description into an optimised gate level description for a partial technology. At this stage the designer has to use the technology library which contains information for the technology process and cells, such as *OR* and *AND*. For the example we use the library `lsi_10k` for a 2μ CMOS process.

First students use the *Design Analyzer* program to analyse the VHDL code. The result of this program is an intermediate format performing syntax checking and checking for synthesizable logic. The next elaborate stage translates the intermediate format to SYNOPSYS database format `db`. The results of the RTL synthesis is a set of registers and combinational logic. Some modules are generated directly at blocks with specified arithmetic functions. That's why the logic optimisation procedure is required to implement these functions.

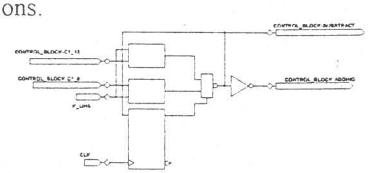

Fig. 5a. Schematic for FSM

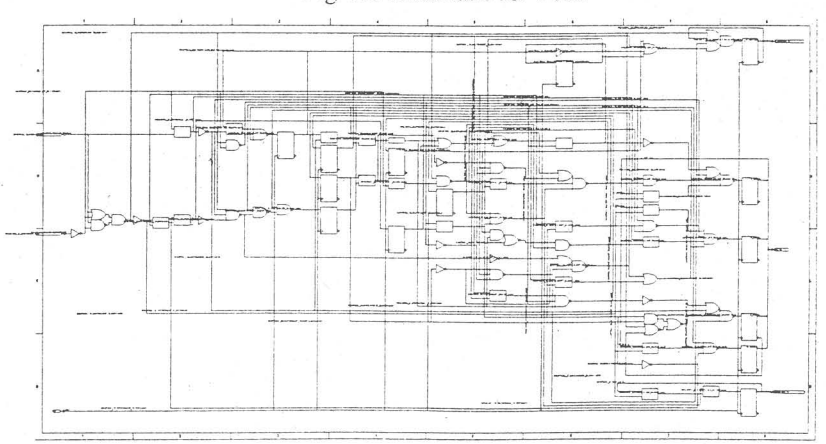

Fig. 5b. Schematic for COUNTER

The designer has to include information such as drive strength on the ports, input and output delays, capacitive load on the output ports, fanout, operating

conditions, wire load model, realistic timing and area goals for the design. The next stages are optimising the design and generating the schematics (Fig. 5). After checking the synthesised design students can generate reports which provide information on the defined attributes, constraines, operating conditions etc.

If the goals are not met, students have to change constraints, modify compile attributes and reoptimize the design. Otherwise they can save results in formats for the *SGE Schematic*, layout application program, SYNOPSYS internal database format `db`.

2.5. VERIFY THE DESIGN

This final phase verifies the gate-level description of the design from Fig. 5. To simulate the description the designer can use the VHDL versions of the silicon vendors cell libraries or to translate the synthesis library to a VHDL simulation library. We use the second approach for our example.

The *VSS simulator* performs mixed-level simulation. For the example, the blocks `COMPARATOR`, `COUNTER` and `FSM` are implemented at the gate-level, but the `MEMORY` block is described at the register-transfer level.

If after simulation the performance goals are not met, the designer has to return to the Implementation or Design Entry phase.

The following step is creating an output netlist to the layout programs. SYNOPSYS provides an interface to the design systems such as CADENCE and Mentor Graphics.

3. CONCLUSIONS

Circuit design using the SYNOPSYS Design System has proven to be an effective tool for learning the functional behaviour of IC. Once behaviour is defined, the corresponding logic is then synthesised. This optimised gate-level description may be automatically converted to a layout. CADENCE Design System has been used by students in their projects to layout design. Finally, the extracted structure from the physical layout is compared with the original schematic. Thus students in Electronics use extensively the design systems in VLSI Design and CAD in Microelctronics courses in the ECAD Laboratory, Technical University of Sofia, to determine if circuit goals are met.

REFERENCES

1. VSS Family Tutorial, SYNOPSYS Tutorial, 1995.
2. Design Compiler Tutorial, SYNOPSYS Tutorial, 1995.
3. Neil H.E. Weste, Kamran Eshraghiam, Principles of CMOS VLSI Design, A Systems Perspective, Second Edition, Addison-Westley Publishing Company, 1993.
4. J. Armstrong, Chip Level Modelling in VHDL, Prentice Hall, 1988.