DAT094

Lab1: Combinational and sequential logic

Chen Wu **September 12, 2024**

Abstract

1. Introduction

2. Experiment Process

3. Results and Discussion

Question 1

What is the purpose of the entity and the architecture sections, respectively?

The concepts of entity and architecture contribute to managing hardware structure designs for engineers. As their name suggests, the *"Entity"* can be seen as a unit from outside perspective, and the *"Architecture"* sections indicate specific realization methods of such an entity unit.

In a declaration of entity, uses must confirm the external properties, ranging from entity and port names, their input and output setting, and data types. Thus, the purpose of setting entities is to allow this unit to be used as a part of the overall design, and user can also personalize its parameter when instantiate an entity.

Architectures contain the details to realization of an entity. It also gives users a more clear display and design way of an entity design, where designers can divide an entity into different parts. In a complex module, the standard of segmentation is generally by their function. Generally, an architecture section only implements one function, so that this is a more efficient method to design function block and debug. Besides, these two definitions provide more convenient solutions for design's reusability.

Question 2

Describe the std_logic type. What are the possible values that a signal of this type can take? Can you motivate this set of values from a simulation perspective?

According to IEEE1164, std_logic supports a lot of different types of signal. There is a table to illustrate them respectively:

Value	Description	Motivation from Simulation Perspective
′U′	Uninitialized	Indicates that the signal has not been initialized, helping detect uninitialized signals during simulation.
′X′	Forcing Unknown	Represents an unknown state due to signal conflicts or undefined values, useful for detecting driver conflicts.
′0′	Logic 0	Represents logical low (false), typically used for normal binary operation.
′1′	Logic 1	Represents logical high (true), typically used for normal binary operation.
′Z′	High Impedance	Represents a tri-state condition, useful for modeling buses and tri-state buffers where signals are not driven.
′W′	Weak Unknown	Indicates a weakly driven unknown state, useful for modeling situations where the drive strength is insufficient.
′L′	Weak 0	Represents a weakly driven logic 0, commonly used for weak pull-down configurations.
′H′	Weak 1	Represents a weakly driven logic 1, commonly used for weak pull-up configurations.
′-′	Don′t Care	Represents a "don′t care" condition, useful for synthesis optimization where the exact value is irrelevant.

Table 1: Possible Values of the `std_logic` Type and Their Simulation Motivations

However, there are four types we used most frequently: $'1', '0', 'Z', 'X'$. Those values can be set for a signal, and it is also the only four meaningful and potential occurred values in simulation. So we can, to some extent, simplify above types into the four ones.

We have no necessary to discuss the value ′0′ and ′1′, which are the basic of any digital system. $'X'$ will appear when you forget to initialize (reset) circuit when simulating, especially there exists some registers. It means that the signal state can be high or low because, without initialization, register values determined by physical circuit situation randomly when power is on. $'Z'$ represents high-impedance mode of a tri-state gate.

Question 3

What is the default length of the input and output vectors in the module convert_data_format?

```
1 entity convert_data_format is
2   generic(SIGNAL_WIDTH : integer := 4);
3   port(input   : in  std_logic_vector(SIGNAL_WIDTH-1 downto 0);
4        convert: in  std_logic;
5        output : out std_logic_vector(SIGNAL_WIDTH-1 downto 0));
6 end entity convert_data_format;
```

Listing 1: Entity declaration in "convert_data_format.vhdl"

According to the second line in Listing 1, we can easily find the default bit width input and output vectors is 4.

Question 4

What is the meaning of the symbol <= as used in the file?

In VHDL, " <= " is an operator to realize the assignment of a signal. The effect has a little difference with the assignment behaviors in software development. As in verilog, the value would not be directly conveyed to the signal, but developers can consider this assignment will happen when the architecture's function has been done.

Note 1

Noticeably, in VHDL, there is no a conception of blocking assignment rather than in Verilog. The non-blocking assignment (" <= ") that all assignments happen in parallel, which is so important in sequential logic designs. The singularity of assignment operator helps us reduce the mistakes caused by misunderstandings on blocking and non-blocking assignment in Verilog.

Question 5

What is the meaning of the symbol & as used in the file?

```
-- In .vhdl file
output <= not(input(SIGNAL_WIDTH-1)) & input(SIGNAL_WIDTH-2 downto 0) when (convert = '1');
```

Listing 2: "&" symbol in "convert_data_format.vhdl"

```
-- In testbench file:
assert (output = "0110")
  report "Error for input " & to_string(input) &
  ": The output value is " & to_string(output) &
  ", but it should be " & "0110"
  severity warning;
```

```vhdl
wait for 100 ns;                             -- 150 ns 1110

assert (output = "1110")
  report "Error for input " & to_string(input) &
  ": The output value is " & to_string(output) &
  ", but it should be " & "1110"
  severity warning;
wait for 100 ns;                             -- 250 ns 0110

assert (output = "1110")
  report "Error for input " & to_string(input) &
  ": The output value is " & to_string(output) &
  ", but it should be " & "1110"
  severity warning;
wait for 100 ns;                             -- 350 ns 1110

assert (output = "0110")
  report "Error for input " & to_string(input) &
  ": The output value is " & to_string(output) &
  ", but it should be " & "0110"
  severity warning;
wait for 50 ns;

-- Force testbench to stop after 400 ns
report "Testbench finished!" severity failure;
```

Listing 3: "&" symbol in "convert_data_format_tb.vhdl"

In Listing 2, the symbol of ″&″ is a concatenation operator in VHDL which can gather different signals into a new port or an internal sigal. In testbench file like Listing 3, this notation has more useful and flexible usage. When setting assertion part in a testbench, users can use this to concatenate strings similar to the ″ + ″ operator in high-level program languages.

Question 6

Describe in your own words how this module computes output values from the input values.

```vhdl
entity convert_data_format is
  generic(SIGNAL_WIDTH : integer := 4);
  port(input   : in  std_logic_vector(SIGNAL_WIDTH-1 downto 0);
       convert: in  std_logic;
       output : out std_logic_vector(SIGNAL_WIDTH-1 downto 0))
  ;
end entity convert_data_format;

architecture convert_data_format_arch of convert_data_format
   is
```

```
begin
  output <= not(input(SIGNAL_WIDTH-1)) & input(SIGNAL_WIDTH-2
   downto 0) when (convert = '1')
            else input;
end architecture convert_data_format_arch;
```

Listing 4: Entity declaration in "convert_data_format.vhdl"

Listing 4 is the specific architecture in a converter entity. This entity design has two input ports and an output port.

The single bit input port ”convert” takes the responsibility of controlling converting function part work or not. When input port ”convert” is enabled (high), the ”output” port is assigned the inverted value of the $(\text{SIGNAL\_WIDTH} - 1)^{\text{th}}$ bit of the input, followed by the remaining bits. It will directly output the input value.

Question 7

Using the QuestaSim Help system, briefly describe in your own words what each command in the do file does.

```
restart -f -nowave
config wave -signalnamewidth 1

add wave convert
add wave -divider "Binary Representation"
add wave -radix binary input output
add wave -divider "Signed Representation"
add wave -radix decimal input output
add wave -divider "Unsigned Representation"
add wave -radix unsigned input output

run 400ns

view signals wave
```

Listing 5: convert_data_format_tb.do File

With the help of official guide files [1], we can get the meaning of each line in the do file (Listing 5). A table summarized keywords meaning in a do file is following:

Based on the Table 2, the Listing 5 can be interpreted into those steps:

1. Restart Simulation:
   - `restart -f -nowave` – Reset the simulation without previous waveforms.
2. Configure Waveform:
   - `config wave -signalnamewidth 1` – Set signal name width in the waveform display.
3. Add Signals:

Keyword	**Meaning**
`restart`	Restarts the simulation environment and clears any previous waveforms.
`config wave`	Configures waveform display parameters, such as the signal name width.
`add wave`	Adds a signal to the waveform window for observation during the simulation.
`add wave -divider`	Adds a divider in the waveform window to organize and distinguish between different sections of signals.
`add wave -radix`	Sets the radix (number format) for signal display, such as binary, decimal, or unsigned.
`run`	Runs the simulation for a specified amount of time (e.g., 400 ns).
`view signals`	Opens the waveform window to display signals and their waveforms.

Table 2: Summary of do file keywords

- `add wave convert` – Add signal `convert` to the waveform.

4. Set Up Dividers and Radix:
   - Binary Representation:
     - Add a "Binary Representation" divider and display input and output in binary.
   - Signed Representation:
     - Add a "Signed Representation" divider and display input and output in signed decimal.
   - Unsigned Representation:
     - Add an "Unsigned Representation" divider and display input and output in unsigned decimal.
5. Run Simulation:
   - `run 400ns` – Run the simulation for 400 nanoseconds.
6. View Waveforms:
   - `view signals wave` – Open the waveform viewer to observe signals.

Note 2

Compare my previous experience of using Modelsim, just clicking the right button to add all ports into a waveform window, wrting and executing a do file is so convenient and has a better portability.
Of which, adding some dividers and changing different signals into appropriate data formats are the most impressive on me.

Question 8

Describe in your report the malfunction caused by the change in signal range in convert data format.vhdl, and how the test bench indicated that something was wrong.

```

** Warning: (vsim-WLF-5000) WLF file currently in use: vsim.wlf
          File in use by: root  Hostname: knuffodrag.ita.chalmers.se  ProcessID: 217792
          Attempting to use alternate WLF file "./wlftEs6x53".
** Warning: (vsim-WLF-5001) Could not open WLF file: vsim.wlf
          Using alternate file: ./wlftEs6x53
** Warning: Error for input 0110: The output value is 1011, but it should be 1110
   Time: 250 ns  Iteration: 0  Instance: /convert_data_format_tb
** Warning: Error for input 1110: The output value is 0111, but it should be 0110
   Time: 350 ns  Iteration: 0  Instance: /convert_data_format_tb
** Failure: Testbech finished!
```

Listing 6: Errors and warnings show in terminal

Benefiting from those assertion commands set in testbench file, we can quickly locate the unexpected value. Besides, observing and comparing waveform (Figure 1 and Figure 2) before and after changing is another method to detect, but non-intuitively.

Question 9

Make a few other small changes, one by one; for each one, study the pronouncements of the test bench. Did the bench catch all errors? If not, why not?

Question 10

Describe in your own words what the new elements in the sequential logic section represent and accomplish.

Question 11

What is the significance of the list of signal names following the keyword process?

Question 12

Briefly describe, in your own words, the idea behind a finite state machine (FSM).

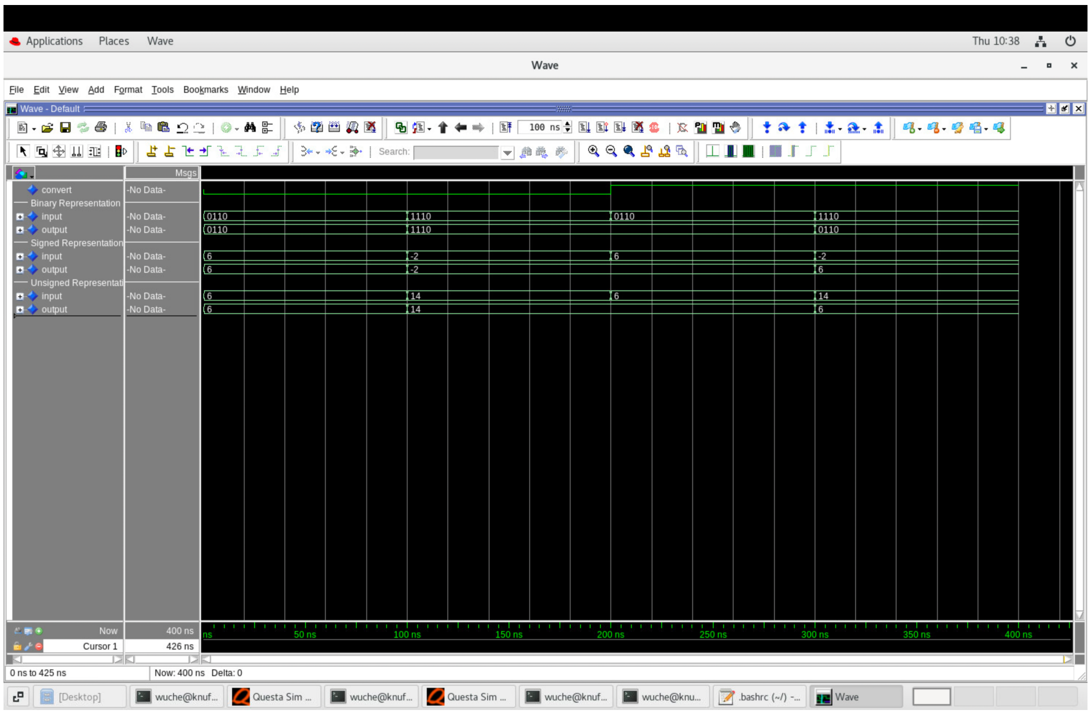

Figure 1: Before

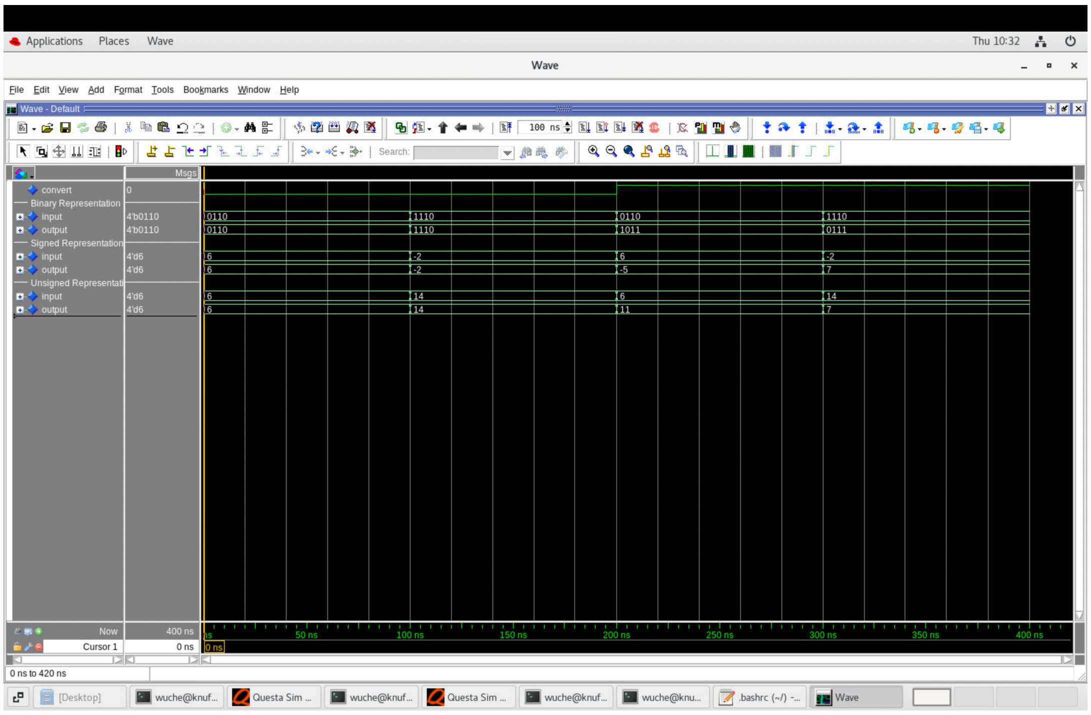

Figure 2: After

Question 13

Each of the process clauses in the example file implements one aspect of the FSM behavior. What are these?

Question 14

The clk and rst signals do not occur in all the process clauses. Why is this?

Question 15

Compare the port lists of the component with that of the entity in dac controller.vhdl. Are there any significant differences?

Question 16

Describe in your report the malfunction caused by the change in state transition in dac controller.vhdl, and how the test bench indicated that something was wrong.

Question 17

Make a few other small changes; for each one, study the pronouncements of the test bench. Did the bench catch all errors? If not, why not?

Appendix: Challenges and Solutions

References

[1] "Help: Tcl and do files." (2024), [Online]. Available: `file:///usr/local/cad/questa-2024.1.2/questasim/docs/htmldocs/mgchelp.htm#context=questa_sim_user&id=1357&ihub=questa_sim_ih&Sword=do` (visited on 09/09/2024).