ECE 3270 - Digital Computer Design - Dr. Ligon

JAN 12 2024

ASIC: application specific integrated circuit

FPGA: Field programmable gate array

VHDL: hardware description language

textbook: digital design using vhdl a systems approach

room 309 room code: 327468

logical and physical specification

1. design entry types:
   - truth tables, waveforms, state diagrams.
   - Schematic capture.
   - hardware description language.

netlist mixed-mode synthesis

JAN 17 2024

hierarchical design methodology involves creating segments of a product that are designed with different tools at differeing levels of specificity.

synthesizer cad tool takes specification/hardware description language and creates design from the specification.

Lab portion of the class involves using synthesizer on HDL code.

schematic capture (drawing schematics) becomes untenable with larger and more complex designs

during process of synthesis, technology mapping involves putting specific hardware into design.

1. EDIF: Electronic Design interchange format.

2. Functional Simulation is final step that verifies design.
   - functional simulators assume the time needed for signals to propagate through the logic gates

   is nebligible.

   - timing simulators must be used in tandem with functional simulators in order

   to obtain a complete test of the design.

   - add timing bounds to portions of circuit: best, worst, typical

   this allows for testing with all possible operation timings.

   - Use spice for specific timing measurements/edge cases. Spice uses

   numerical methods and has higher precision than simple digital logic simulators

   1. Event-driven simulation:

      - Zero delay simulators will not detect race and hazard conditions.
      - Race condition: when two or more signals are changis simultaneously in a circuit which may resuelt in an ein correct state when a condition is assumed to be stable.

      1. Possible states in logic simulation:
      - 0

    - 1
    - U (unknown)
    - Z (high impedance) Relevant with TSB (tri-state buffer) and TG (transmission gate)

    use reset or preset signal to set unknown back into a known state.

  Mixed-mode simulation: logic and spice simulation together

JAN 22 2024

1. Design – Place and route – Schematic Capture

  - pin data:
    1. placement
    2. electrical resistance
    3. signal name
    4. special flags
  - Routing
    1. traditional : by gate
    2. floor planning : by structure
  - Simulations to estimate real-world timing
    - identify critical paths
    - full logical simulation
    - full electrical simulation
    - leads to re-routing, or re-design

Lecture 1: The Digital Abstraction, Combinational Logic.

Associated reading: chapter 1, 3, 6

- The Digital Abstraction
  - representation
  - noise
- Low voltage CMOS logic:

[Damage] – −0.3V – 0.0V – 0.7V – 1.7V – 2.5V – 2.8V [Damage]

- 0.7 to 1.7 V is transition region for 2.5V CMOS logic.
- As opposed to analog systems, digital systems can:
  - Process, transport, and store info without noise distorion.
  - Possible because the signals are discrete
  - No loss of informmation with added noise until the noise becomes large enough to push the signal our of the valid range
- Digital signals are periodically restored to keep them in the vaoid range using a buffer.
- In analog systems, since all voltages are valid signals there is no way to restore the signal to a noiseless state between operations.
- Analog systems also limited in precision
  - Accuracy is limited by the background noise.
- All resotring logic devices guarantee that the outputs fall into a range that is narrower than the input range

- Larger noise margins are not necessarily better.

$V_{NMH} = V_{OH} - V_{IH}$

Jan 24 2024G

- Combinational logic means that logical outputs are based soley on inputs, not state or memory.
  - closed under acyclic composition (as long as a feedback loop is not created when composing combinational logic circuits, then the composition is still combinational)
  - combinational, not combinatorial
- Sequential logic depends on memory and state.
- More than one possible logic equation for a given truth table.
- Demorgans law:

$$\neg(x \land y) = \neg x \lor \neg y \to \neg(x \lor y) = \neg x \land \neg y$$

  - Hazards:
    - Hazards occur when changing from one implicant to another
    - Internal timing delays may cause undefined behavior.
    - Prime implicants on k-map.
    - Make circuits hazard-free by adding redundant implicants to cover transitions.
    - Types of hazards: static, dynamic, functional
    - Hazards also occur in sequential circuits.
  - Chapter 7 + appendix A+B: VHDL:

Jan 29 2024 – VHDL syntax

- Explicit declaration
- CONSTANT bus_width : INTEGER := 32;
- CONSTANT rise_delay : TIME := 20ns;
- VARIABLE data_val : STD_LOGIC_VECTOR(7 DOWNTO 0);
- VARIABLE sum : INTEGER RANGE 0 to 100;
- VARIABLE done : BOOLEAN;
- SIGNAL clock : STD_LOGIC;
- SIGNAL addr_bus : STD_LOGIC_VECTOR (31 DOWNTO 0);

Component Instantiation:
- introduces a relationship to a component declaration.
- port map maybe either named or positional.

named: reg5 : fill_reg port map (clk => clk, rst => rst, write => write_comp, read => readout, data_in => dataout_comp, FFout => FFOut, data_out => datareg5_out);

positional:

port map (clk, rst, write_comp, readout, dataout_comp, FFout, datareg5_out);

logical operators do not bind or have precedence except for the not operator.

& - Concatenation operator for strings. String is any sequence of characters.

VHDL identifier rules:
- letters, digits, underscores only
- The last character cannot be an underscore

- two underscores in succession are not allowed
- using reserved words is not allowed.

1. Combinational design entities use only
   1. concurent assignment statements
   2. Case or Case? statements (with “when others =>”)
   3. if statements - only if all signalsa have a default assignment
   4. instantiations of other combinational modules.
2. Sequential desing entitites use only
   1. Combinational logic
   2. Explicitly declared register (flip-flops)
3. Do not use
   1. Loops
   2. Provess except for case, casex, or if
4. Do use
   1. Signal slices e.g., a(7 downto 1) = b(6 downto 0);
5. Logic is organized into small design entities.
   1. Leaf design entities not more than 40 lines
   2. If it could be made two design entities, if should be.
6. Use lots of comments
   1. Comments
   2. Meaningful signal names - tempHigh, not th
   3. Meaningful module names - DaysInMonth not mod3
7. Constants
   1. All constants explicitly defined if used nore than once
8. Signals
   1. Bused (multi-bit signals) are numbered high to low
      - e.g., bus(31 downto 0)
   2. All signals should be high-true (except primary inputs and outputs)
9. Visualize the logic your VHDL will generate.
   1. if you can’t visualize it, the result will not be pretty

January 31 2024

1. What is a VHDL process?
   1. proecesses are either awake or asleep
   2. A process normally has a sensitivity list
      1. When a signal in that sensitivity list changes value, the process wakes up and all of the sequentil statments are “executed”
      2. for example, a process witha clock signal in its sensitivity list will become active on changes of the clock signal
      3. At the end of the process, all outputs are assigned and the process goes back to sleep until the next time a signal changes in the sensitivity list.

2. Process
   1. If no sensitivity list is given, then wait statements must be used in the process

3. Multiple statements can execute concurrently

4. The statements describing the behavior are executed sequentially
   1. This is true from a simulation standpoint

   2. From a synthesized hardware point-of-view, multiple assignments to a single signal (variable) generally implies multiplexing of the assignments to produce a signal output

5. Assignments made inside the process are not visivle outside of it.

6. process

```
begin
  wait for 15 ns;
  clk <= not(clk);
end process;

process(clk, rst)
begin
  if rst = '1'; then
    readout <= '0';
  elsif (clk'event and clk = '1') then
    if(fout = '1') then
      readout <= '1';
    else
      readout <= '0';
    end if;
  end if;
end process;
```

1. clk'event is attribute of clk
2. when writing a <= b and b <= c updates happen at end of process
3. concurrent means nonprocedural (no in a process statement)
   1. The order in which the CSA statements appear textually has nothing to do with the order in which they execute
   2. They execute at the same time essentially
   3. Concurrency is fundamental to hardware and VHDL, think in terms of parallel signal transforms

```
with s select
  x <= a when "00",
    b when "01",
    c when "10",
    d when "11";

with int_value select
  x <= a when 0 to 3,
    b when 4 | 6 | 8,
    c when 10,
    d when others;
```

Feb 2 2024

```
signal_name <= value_1 WHEN condition1 ELSE
  value_2 WHEN condition2 ELSE
  ...
  value_n WHEN conditionN ELSE
  value_x;
```

```
x <= a when (s = "00") else
  b when (s = "01") else
  c when (s = "10") else
  d;
```

Feb 5 2024 – Testbench examples – end of first midterm material

```
// & operator is string concatenation
//
  report "input = " & to_string(to_integer(unsigned(input))) &
    " isprime = " & to_string(isprime);

    // empty testbench
    entity testbench is
    end testbench

architecture test+adder of testbench is
  signal clk : std_logic := '0';
  signal rst : std_logic := '0';

  signal a : std_logic_vector(4 downto 0);
  signal b : std_logic_vector(4 downto 0);
  signal c : std_logic_vector(4 downto 0);

  begin
  dut : entity adder
    port map( in1 => a, in2 => b, out1 => c);

  //process for simulating the clock
  //
  process
  begin
    clk <= not(clk);
    wait for 20ns;
    end process;

  //This process does the RESET

  process
  begin
    rst <= '1';
    wait for 53ns;
    rst <= '0';
    wait until(rst'event and rst = '1');
    //stops this process from happening again (this is an initial)
  end process;
```

- VHDL supports generics
- An aggregate is a collection of items that are gathere together to form a total quantity

```
v <= (others => '0');
v <= ('1', '0', others => '0'); // "00000000"
v <= (4 => '1', others => '0'); // "00010000"
v <= ( 3 DOWNTO 0 => '0', others => '1'); // "11110000"

//example generics

ENTITY counters IS
GENERIC ( WIDTH: INTEGER := 8);
  PORT(
    d : IN STD_LOGIC_VECTOR(WIDTH-1 DOWNTO 0);
    clk: IN STD_LOGIC;
    clear: IN STD_LOGIC;
    load: IN STD_LOGIC;
    up_down : IN STD_LOGIC;
    qd : OUT STD_LOGIC_VECTOR(WIDTH-1 DOWNTO 0);
  )
```

- For any instances of a component, use FOR GENERATE
- This instantiates an object in each loop, which can help you create many numbers of similar components
- no breaks, not actual loop

Feb 7 2024 – Review Day

-Multi state logic: 0, 1, U Z+, Z-

Z+ input comes from floating output closer to a 1, Z- floatint output closer to a zero weak 1 vs weak 0

Tristate buffer:

VHDL overall layout

entity - parts specification architecture - actual logic description. process is piece that goes into architecture. behavioral, structural, dataflow. process is behavioral. switch level simulation - hybrid electrical and event driven simulation. mixed mode - is another hybrid simulation but ususally has 2 seperate simulations running in tandem. rise/fall time vs rise/fall delay – testbench variables are not signals, cannot be assigned to ports or sensitivity list, etc. do as much as you can without variables. := assignment, <= continuous assignment

```
process(clk, rst)
  if(rst) then q <= 0;
  else if(clk'event = 1 and clk = 1) then q <= d;
  end;
end;
```