EITF35: Introduction to Structured VLSI Design

VHDL-5: Coding Style

Liang Liu
liang.liu@eit.lth.se

Traditional PL v.s. VHDL

Traditional PL

- Operations performed in a **sequential** order
- Help human's thinking process to develop an algorithm step by step (dangous in HDL)
- Resemble the operations of a basic **computer model**

HDL – Characteristics of digital VLSI

- Connectiosn of parts
- Concurrent operations
- Concept of propagation delay and timing

Example

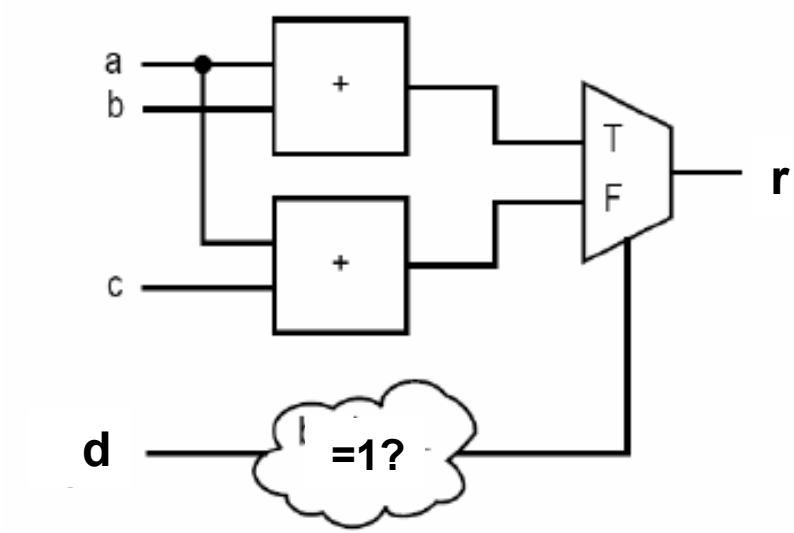
- HDL code:

```
Process (a,b,c,d)
Begin
  if (d='1')
    r <= a+b;
  else
    r <= a+c;
  endif
End process
```

- Traditional PL

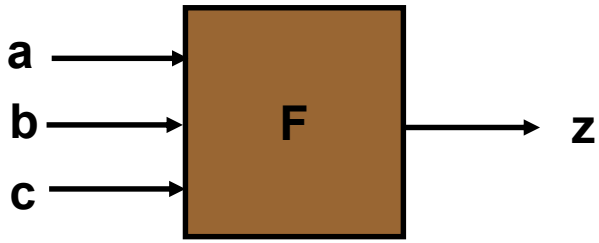
```
if (d='1')
  r <= a+b;
else
  r <= a+c;
endif
```

HDL



Two Basic Digital Components (What)

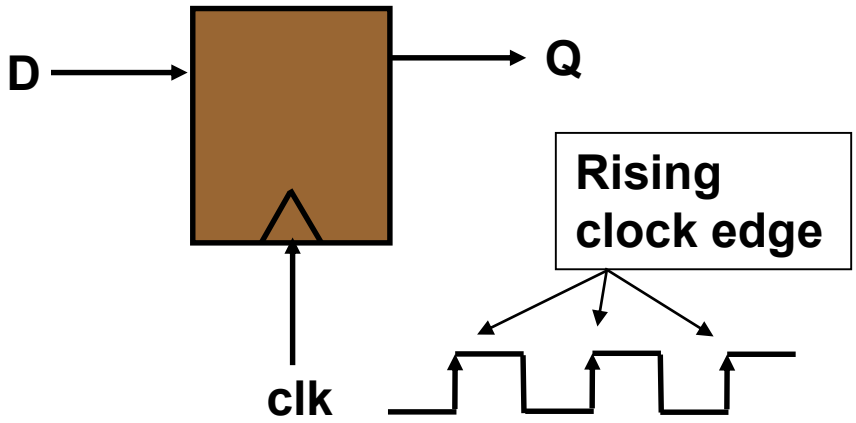
Combinational Logic



Always:
z <= F(a, b, c);

i.e. a function that is always evaluated when an input changes.
Can be expressed by a truth table.
Outputs only depend on PRESENT inputs (no memory)

Register



if clk' event and clk= '1' then
Q <= D;

i.e. a stored variable,
Edge triggered D Flip-Flop with enable.
Can memorize input (for at least one clock cycle)

Two-Segment Coding

- Separate combinational circuits and registers

Go ahead, two-segment works!

Comments "--": may be "more important"

```
-------------------------------------------------
-- Design : CARRIER SENSE
-- File Name : CARRIER_SENSE.vhdl
-- Purpose : Model of CARRIER SENSE process in PCS (IEEE Std 802.3)
-- Limitation : none
-- Errors : none known
-- Include Files: none
-- Author  : Liang Liu, liang.liu@eit.lth.se, Lund University
-- Simulator  : ModelSim 6.5
-- ------------------------------------------------
-- Revision List:
-- +----------+------------------+---------------+--------------------------------+
-- | Version  | Author           | Date          | Changes                        |
-- +----------+------------------+---------------+--------------------------------+
-- |1.0       |      Liang Liu   | 2001/08/03  | original created               |
-- |1.1       |      Liang Liu   | 2002/01/04  |disable TX_EN to CRS    |
-- |          |                  |               |in repeater mode                |
-- +----------+------------------+------ ----------+--------------------------------+
```

Make your code readable!!!
Maintenance and re-usability

Indent the code properly

```
architecture one_seg_arch of dff_en is
begin
  process (clk,reset)
    begin
      if (reset='1') then
        q < = '0' ;
      elsif (clk'event and clk='1') then
        if (en='1') then
          q <= d;
        end if;
      end if ;
  end process;
end one_seg_arch;
```

Use two space instead of 'tab' for indenting

some tools like vim and emacs treat 'tab' differently

Names

Make your file name and signal name meaningful that others can read your code conveniently

- **Suggestion for signal name: direction_function_feature**

  For example: *i_clk_r*, means an input clock with rising edge trige

  Mark register output and combinational logic output

- **One entity per file and make the file name the same as entity name**

Data types

- **Integer: only for CONSTANT, NOT for SIGNALS**
- **std_logic/std_logic_vector: often for interface (I/O) and logic signals**
- **signed/unsigned: signals for arithmetic operation**

Complete Sensitivity List

3-input and circuit

```
bad: process(a)
begin
    y <= a and b and c;
end process;
```

```
Good: process(a,b,c)
begin
    y <= a and b and c;
end process;
```

For a combinational circuit, all inputs need to be included in the sensitivity list!!!

Complete value assignment

- For purely combinational processes:

Every output must be assigned a value for every possible combination of the inputs

```
process (SEL, A, B)
begin
if (SEL = '0') then
Y <= A;
end if;
end process;
```

```
process (SEL, A, B)
begin
if (SEL = '0') then
Y <= A;
else
Y <= B;
end if;
end process;
```

Avoid latch in your design

No For Loops (at least for beginners)

- "For" may make your design process last forever ….

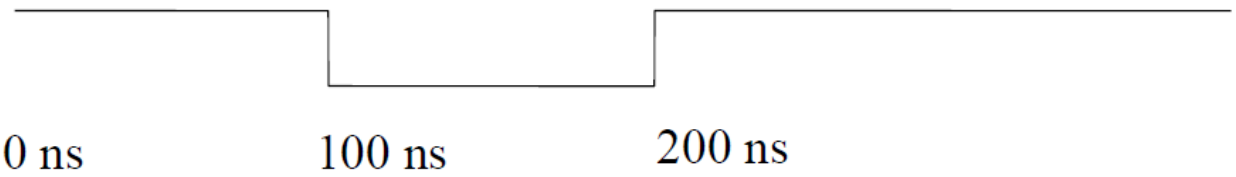
Don’t Wait

- The ‘wait’ or ‘after’ statement can delay for a certain amount of time, e.g., “wait 10ns;”
- Only use it in test benches that are not meant to become hardware
- Do not use them in the design of your hardware
- EDA Tools will insert delay automatically by adding buffer

Example:

```
A <= '1',
     '0' after 100 ns,
     '1' after 200 ns;
```

Not synthesizable



DO NOT introduce uncertainty

- ‘X’ and ‘U’ are legal VHDL, but the synthesized circuit won’t behave like you expect.

```
architecture behv of ALU
is
begin
process (A,B,Sel) begin
case Sel is
when "00" =>
Res <= A + B;
when "01" =>
Res <= A + (not B) + 1;
when "1X" =>
Res <= A and B;
when "1U" =>
Res <= A or B;
when others =>
Res <= A;
end case;
end process;
end behv;
```

```
architecture behv of
ALU is
begin
process(A,B,Sel) begin
case Sel is
when "00" =>
Res <= A + B;
when "01" =>
Res <= A + (not B) + 1;
when "10" =>
Res <= A and B;
when "11" =>
Res <= A or B;
when others =>
Res <=A;
end case;
end process;
end behv;
```

Feedbacks

- Don’t use a combinational feedback
- You never really need them.

```
process(a,b)
begin
      c<=c+a+b;
end process;
```

Multiple Source

- Drive every signal from exactly one process or concurrent assignment.

```
Architecture arc_ad of ad is
begin
      e<=a+b;
      e<=c+d;
end arc_ad;
```

Initial value

- Do NOT assign initial values to signals

```
signal bad_signal : std_logic := '0';
```

Infer registers

```
process (clk)
begin
  if rising_edge(clk) then
    q0 <= d0;
  else
    q0 <= d1;
  end if;
end process;
```

```
process (clk)
begin
  if rising_edge(clk) then
    q0 <= d0;
  end if;
end process;
```

“Rules” for coding sequential circuits

- Strictly follow the synchronous design methodology; i.e., all registers in a system should be *synchronized by a common global clock signal* (otherwise special circuits are needed)

- The memory components should be coded clearly so that a predesigned cell can be *inferred from the device library*.

- Isolate the memory components from the VHDL description and code them in a *separate segment*. One-segment coding style is not advisable.

- *Asynchronous reset*, if used, should be only for system initialization. It should not be used to clear the registers during regular operation

- Unless there is a compelling reason, a *variable should not be used* to infer a memory component.

Do NOT mix

- Combinational logic with registers
- Asynchronous reset with synchronous reset
- Mealy machine with Moore machine

Lecture

- **Oct. 1st Tuesday (13.15-15.00)**

Design for Test (DFT)

Erik Larsson

Professor

- **Oct. 7th Monday (13.15-15.00)**

Torsten Larsson

ERICSSON

Lecture

- **Oct. 8th Tuesday (13.15-15.00)**

Stefan Lundberg AXIS COMMUNICATIONS