ECE381(CAD), Lecture 7:

Dataflow modeling in VHDL

Mehdi Modarressi

Department of Electrical and Computer Engineering,

University of Tehran

Pictures and examples are taken from the slides of "VHDL: Analysis & Modeling of Digital Systems" and also from "VHDL by Example"

Dataflow modeling

- Readings:
  1. "VHDL: Analysis & Modeling of Digital Systems": Chapter 8.1.1 to 8.1.4
  2. "VHDL by Example": Chapter 2
  3. "Designers Guide To VHDL": parts of chapters 5 and 13

! A difference in taxonomy in the book "VHDL programming by example"

- The name Chapter 2 is "Behavioral modeling", but I classify the techniques discussed in this chapter as dataflow
- The name of chapter 3 is "Sequential modeling", but I classify the techniques discussed in that chapter as behavioral

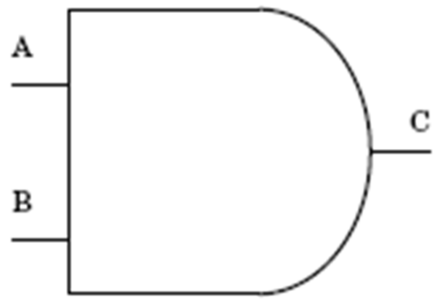
Dataflow modeling

- Specifies the flow of information
- Assigning values to signals
  - In the concurrent body of the architecture
- Signal: an object that holds data
  - Equivalent to a wire in real hardware
- Recall: four classes of objects that can hold data in VHDL:
  - Signal
  - Variable
  - Constant
  - File

Concurrent signal assignment

- Basic structure: concurrent signal assignment statements

  a<=b after 10 ns;



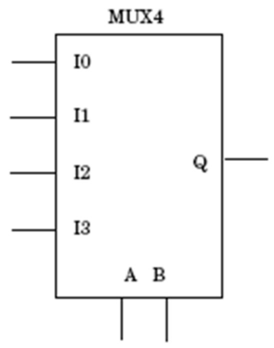
```
ENTITY and2 IS
  PORT ( a, b : IN BIT;
           c : OUT BIT );
END and2;

ARCHITECTURE and2_behav OF and2 IS
BEGIN
  c <= a AND b AFTER 5 ns;
END and2_behav;
```

Dataflow description in VHDL

- Concurrent signal assignment is the basic dataflow construct
- Three assignment constructs with more abilities:
  - Selected signal assignment
  - Conditional signal assignment
  - Guarded signal assignment

Selected signal assignment



- A data flow multiplexer
- The conditions are executed one at a time in sequential order until the conditions of a statement are met
- The first statement that matches the conditions assigns the value to the target signal

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;

ENTITY mux4 IS
PORT ( i0, i1, i2, i3, a, b : IN std_logic;
                          q : OUT std_logic);
END mux4;

ARCHITECTURE mux4 OF mux4 IS
SIGNAL sel: INTEGER;
BEGIN
WITH sel SELECT
 q <= i0 AFTER 10 ns WHEN 0,
      i1 AFTER 10 ns WHEN 1,
      i2 AFTER 10 ns WHEN 2,
      i3 AFTER 10 ns WHEN 3,
      'X' AFTER 10 ns WHEN OTHERS;

sel <= 0 WHEN a = '0' AND b = '0' ELSE
       1 WHEN a = '1' AND b = '0' ELSE
       2 WHEN a = '0' AND b = '1' ELSE
       3 WHEN a = '1' AND b = '1' ELSE
       4 ;
END mux4;
```

Conditional signal assignment

- Condition waveform has a series of waveforms with condition

```
o1 <= a WHEN cond ='1' ELSE o1;
```

- Can use keyword UNAFFECTED

```
o1 <= a WHEN cond ='1' ELSE UNAFFECTED;
```

- Can use multiple conditions

```
Z <= a AFTER 5 NS WHEN d = '1' ELSE
        UNAFFECTED WHEN e = '1' ELSE
        b AFTER 5 NS  WHEN f = '1' ELSE
        c AFTER 5 NS;
```

Conditional signal assignment

```
ARCHITECTURE better OF mux IS
BEGIN
  q <= i0 WHEN a = '0' AND b = '0' ELSE
       i1 WHEN a = '1' AND b = '0' ELSE
       i2 WHEN a = '0' AND b = '1' ELSE
     i3 WHEN a = '1' AND b = '1' ELSE
     'X';       --- unknown
END better;
```

- A multiplexer in dataflow with conditional signal assignment

Dataflow DFF

```
ENTITY d_flipflop IS
  GENERIC (delay1 : TIME := 4 NS; delay2 : TIME := 5 NS);
  PORT (d, c : IN BIT; q, qb : OUT BIT);
END d_flipflop;
 --
ARCHITECTURE assigning OF d_flipflop IS
  SIGNAL internal_state : BIT;
BEGIN
 internal_state <= d WHEN (c ='1' AND NOT c'STABLE) ELSE internal_state;
  q <= internal_state AFTER delay1;
  qb <= NOT internal_state AFTER delay2;
END assigning;
```

- *internal_state* keeps the current FF state
- GENERIC is used to pass design parameters to instances
  - Will be introduced soon

Take care of multiple drivers!

```
USE WORK.std_logic_1164.ALL;
ENTITY mux IS
PORT (i0, i1, i2, i3, a, b: IN std_logic;
      q : OUT std_logic);
END mux;

ARCHITECTURE bad OF mux IS
BEGIN
  q <= i0 WHEN a = ‘0’ AND b = ‘0’ ELSE ‘0’;
  q <= i1 WHEN a = ‘1’ AND b = ‘0’ ELSE ‘0’;
  q <= i2 WHEN a = ‘0’ AND b = ‘1’ ELSE ‘0’;
  q <= i3 WHEN a = ‘1’ AND b = ‘1’ ELSE ‘0’;
END BAD;


 ARCHITECTURE better OF mux IS
 BEGIN
   q <= i0 WHEN a = ‘0’ AND b = ‘0’ ELSE
        i1 WHEN a = ‘1’ AND b = ‘0’ ELSE
        i2 WHEN a = ‘0’ AND b = ‘1’ ELSE
     i3 WHEN a = ‘1’ AND b = ‘1’ ELSE
     ‘X’;       --- unknown
 END better;
```

- The first architecture assigns multiple values to q when an event occurs on a or b
- Multiple drivers for q
  - Must be resolved
- The second architecture is correct

Guarded signal assignment

- The concept of blocks in VHDL
- Blocks are a partitioning mechanism within VHDL
  - Allow the designer to logically group areas of the model
- Each block represents a self-contained area of the model.
- Can declare any object that can be declared in the architecture declaration section:
  - local signals, types, constants, and so on.

Guarded blocks in VHDL

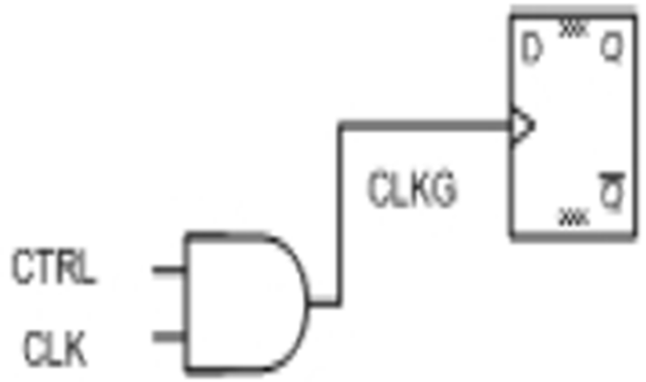
- A *guarded block* contains a *guard expression* that can enable and disable drivers inside the block
- Used for clock gating
- The guard expression is a boolean expression
  - When true, drivers contained in the block are enabled

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
ENTITY latch IS
PORT( d, clk : IN std_logic;
      q, qb : OUT std_logic);
END latch;

ARCHITECTURE latch_guard OF latch IS
  BEGIN
  G1 : BLOCK( clk = ‘1’)
  BEGIN
   q <= GUARDED d AFTER 5 ns;
   qb <= GUARDED NOT(d) AFTER 7 ns;
  END BLOCK G1;
END latch_guard;
```

Guarded signal assignment

- Guarded signal assignment is often used for *clock gating* in digital systems
- Clock gating: one of the most effective mechanisms to reduce power consumption of digital systems
- Disables or suppresses the clock of a part of the chip, when the part is idle



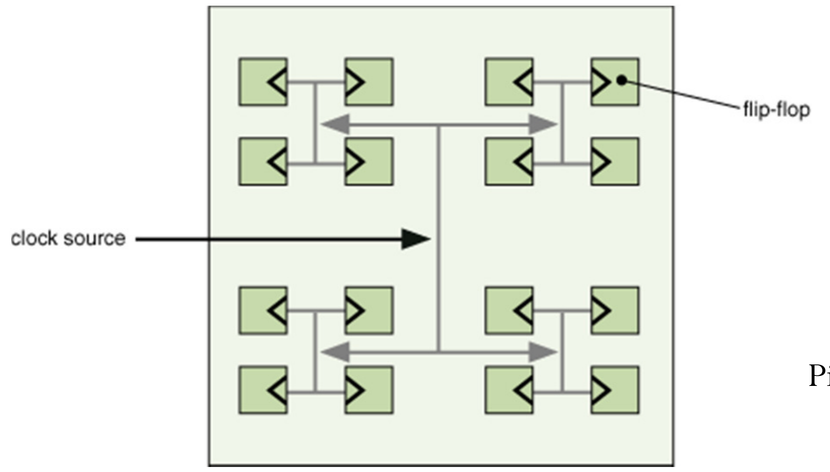
Why low-power design?

- Low power design is important from three different reasons
- Device temperature
    - To meet chip's *TDP*, Cooling and packaging costs
    - What is *TDP*?
- Battery-life
    - In battery-operated hand-held devices
- Environment
    - Overall energy consumption

Why clock gating is effective?

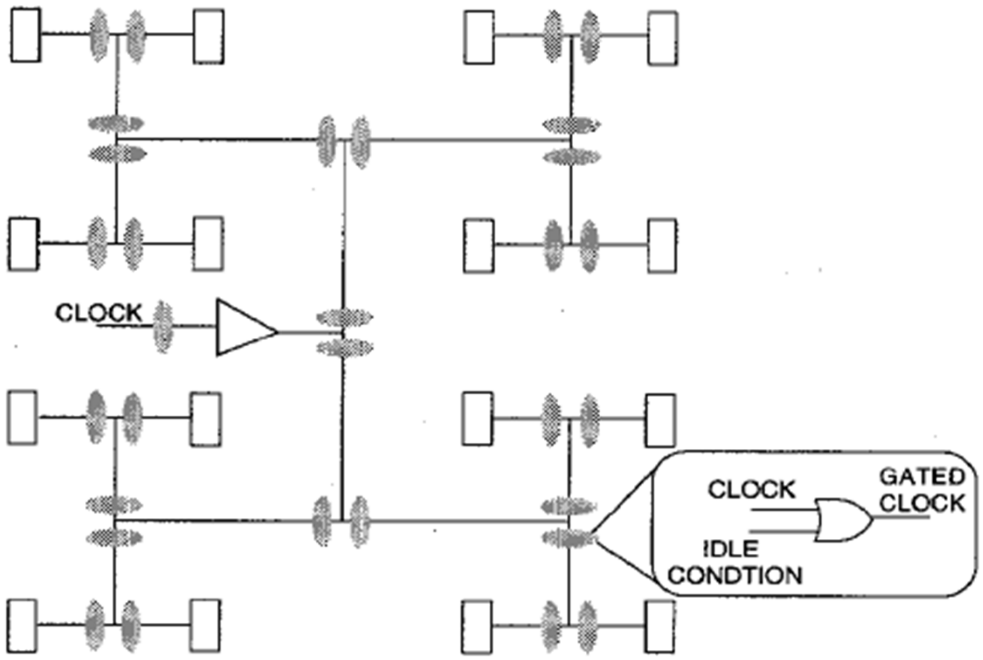
- Clock signal is the most active part of any digital system
  - One 1→0 and one 0→1 transition in each cycle
- Clock signal is fed to all memory elements of the chip (flipflops, registry file,…)
- The entire clock tree is charged and discharged per cycle

→ considerable power consumption (sometimes around 30% of the total chip power)



Picture source: http://www.sanyosemi.com

Clock gating



- The dark parts are the gating logic

  Can disable the clock of a part of the tree

Picture source: LP VLSI course at UT Delft

Generics

- Used for design parameterization
- Why would a designer want to pass information to an entity?
- Instance-specific information
  - Such as delay times
  - Parameters such as datapath widths, signal widths, and so on
- Example: what if we need two and gates with two different delays?
  - Write 2 different architectures
  - Write one architecture and pass the delay as the parameter
- The latter is done by *Generics*

Generics

```
ENTITY and2 IS
  GENERIC(rise, fall : TIME; load : INTEGER);
PORT( a, b : IN BIT;
      c : OUT BIT);
END AND2;

ARCHITECTURE load_dependent OF and2 IS
  SIGNAL internal : BIT;
BEGIN
internal <= a AND b;
c <= internal AFTER (rise + (load * 2 ns)) WHEN internal = '1'
 ELSE internal AFTER (fall + (load * 3 ns));

END load_dependent;
```

- Generic usage in VHDL

How to use generics

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
ENTITY test IS
  GENERIC(rise, fall : TIME; load : INTEGER);
  PORT ( ina, inb, inc, ind : IN std_logic;
         out1, out2 : OUT std_logic);
END test;

ARCHITECTURE test_arch OF test IS
  COMPONENT AND2
   GENERIC(rise, fall : TIME; load : INTEGER);
   PORT ( a, b : IN std_logic;
          c : OUT std_logic);
  END COMPONENT;
BEGIN
  U1: AND2 GENERIC MAP(10 ns, 12 ns, 3 )
   PORT MAP (ina, inb, out1 );

  U2: AND2 GENERIC MAP(9 ns, 11 ns, 5 )
   PORT MAP (inc, ind, out2 );
END test_arch;
```

- Generics should be passed to the instances when the entity is instantiated
- In the same manner as ports