Digital IC Design

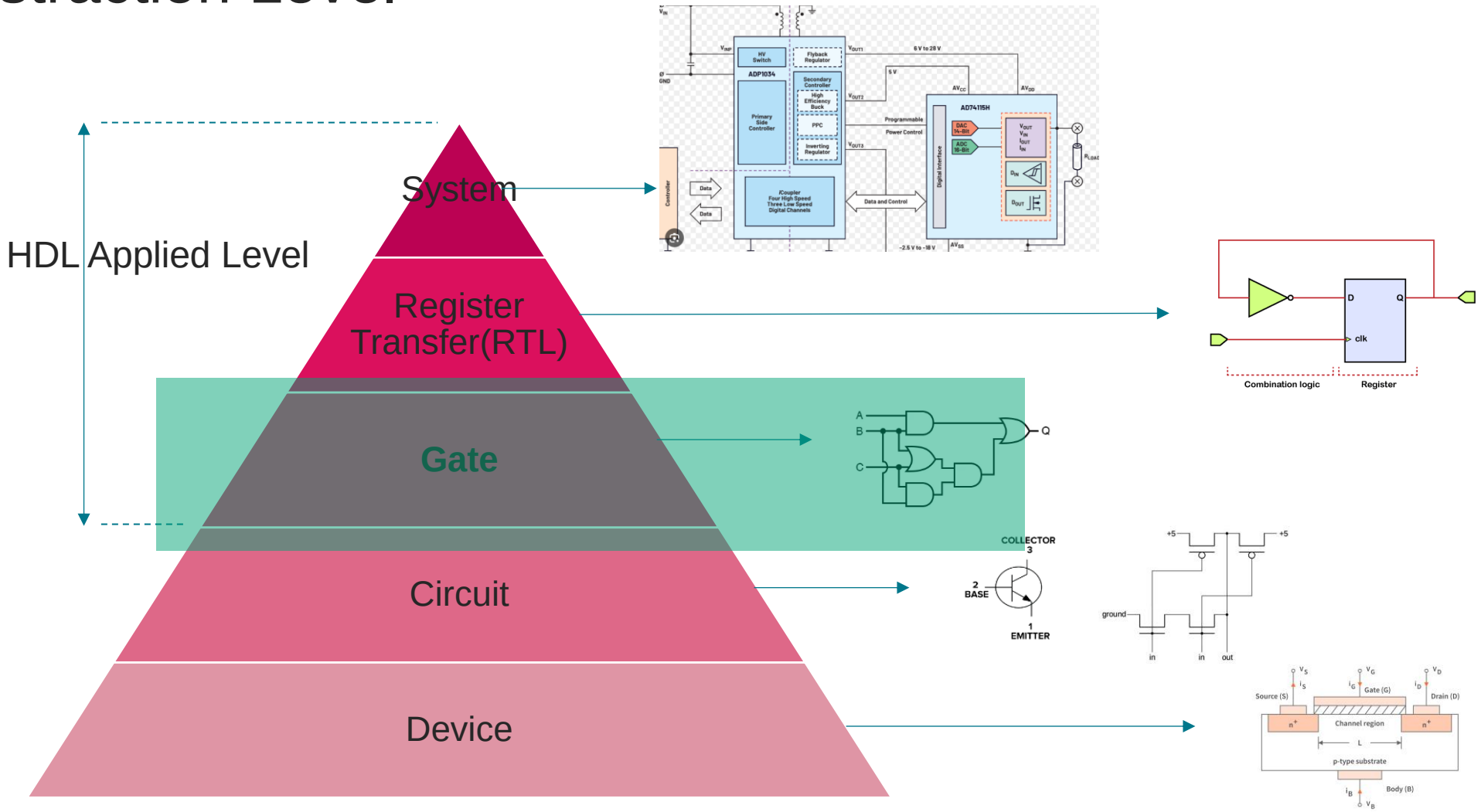
Week3: Gate Level Modeling 1(Basic Overview)

2024 | Nineplus Infotech X Anseong Polytechnic University
Sr Research Engineer, William Woo

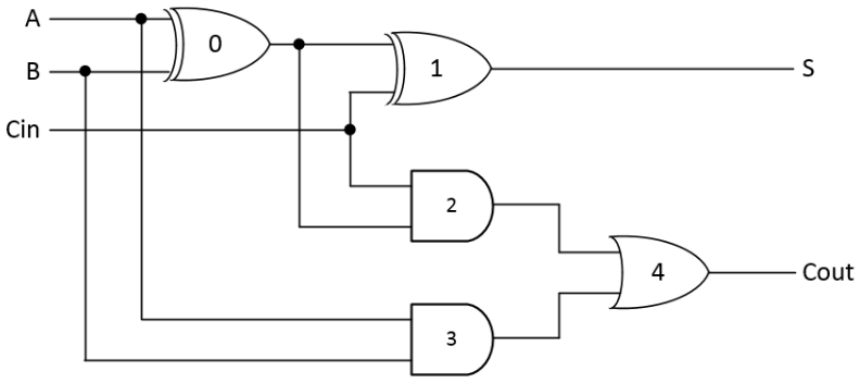
cadence

1. Gate Level Modeling

Abstraction Level



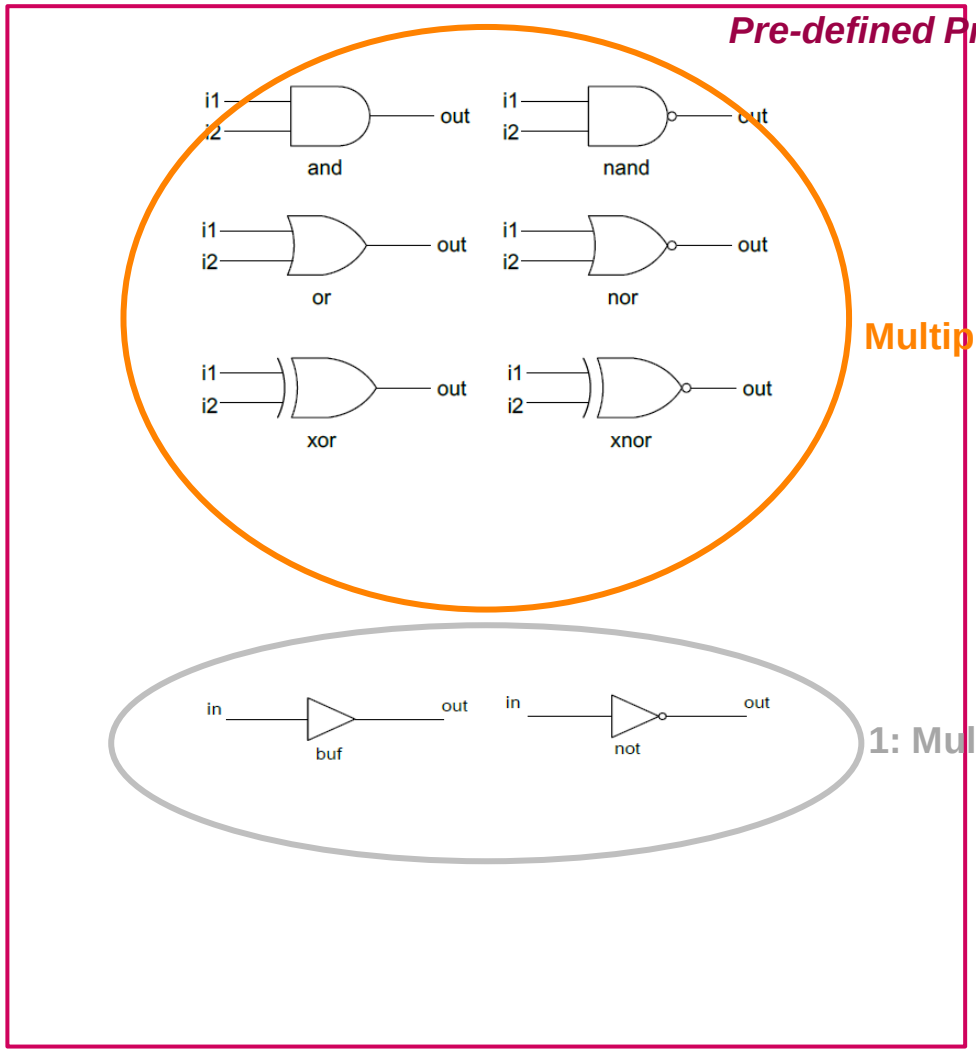
Gate-level Modeling




```
1
2  module full_adder_gatelevel_module (a, b, cin, sum, cout);
3      input a, b, cin;
4      output sum, cout;
5
6
7      // xor_out_1 and xnor_out_2 is not used
8      // wire xor_out_1;
9      wire xnor_out_1;
10     // xnor_out_2;
11     // wire not_out_1;
12
13     wire and_out_1, and_out_2, and_out_3;
14     wire or_out_1;
15
16     //sum = (a⊙b)⊙(cin)
17     //Fill this out
18     xnor_gatelevel_gate xnor_1 (.a(a), .b(b), .out(xnor_out_1));
19     xnor_gatelevel_gate xnor_2 (.a(xnor_out_1), .b(cin), .out(sum));
20
21     //cout = b(cin) + a(cin) + ab
22     //Fill this out
23     and_gate and_1 (.a(b), .b(cin), .out(and_out_1));
24     and_gate and_2 (.a(a), .b(cin), .out(and_out_2));
25     and_gate and_3 (.a(a), .b(b), .out(and_out_3));
26
27     or_gate or_1 (.a(and_out_1), .b(and_out_2), .out(or_out_1));
28     or_gate or_2 (.a(or_out_1), .b(and_out_3), .out(cout));
29
30 endmodule
```

- Gate-level modeling
  - 게이트 및 연결관계를 직관적으로 표현
  - Gate Schematic과 HDL module이 1:1로 대응

Types of Gate: Overview



Pre-defined Primitive Gate

- **Instantiation like a module**
- **Module define (X)**
- **Minimal units for logic circuits**

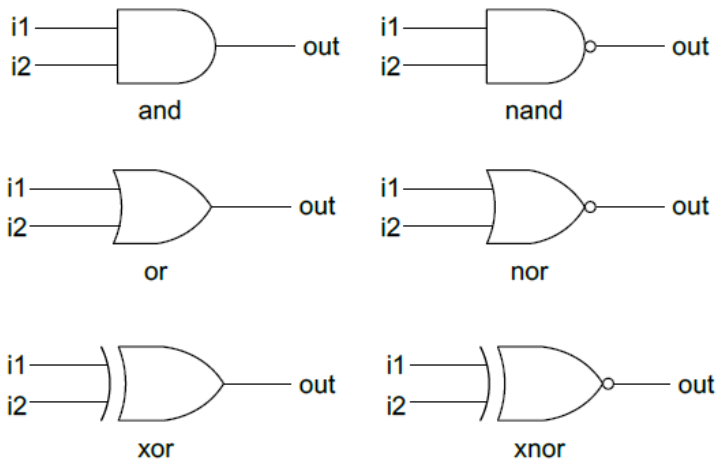
Multiple : 1 Gate

- **Input: Multiple Scalar**
- **Output: one Scalar**

1: Multiple Gate

- **Input: One Scalar**
- **Output: Multiple Scalar**

Types of Gate: Multiple to 1 Gate

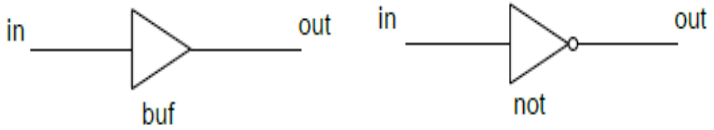


=

```
1   wire out, i1, i2, i3;
2
3
4   // formal gate instantiation
5   and and1    (out, i1, i2);
6   nand nand1  (out, i1, i2);
7   or or1      (out, i1, i2);
8   nor nor2    (out, i1, i2);
9   xor xor2    (out, i1, i2);
10  xnor xnor2  (out, i1, i2);
11
12
13  // n-input gate
14  and and3    (out, in1, in2, in3);
15
16
17  // instantiation without name
18  or          (out, in1, in2, in3);
```

Gate Instance_Name (OUTPUT, INPUT1, INPUT2, ...)

Types of Gate: 1 to Multiple




```
1   wire out1, out2, in;
2
3
4   // formal gate instantiation
5   buf buf1    (out1, in);
6   not not1    (out1, in);
7
8   // N-output gate
9   buf buf2    (out1, out2, in);
10
11  // instantiation without name
12  not         (out2, in);
13
```

Gate Instance_Name (OUTPUT1, OUTPUT2, … , INPUT)

2. Module & Instance

Module: Definition

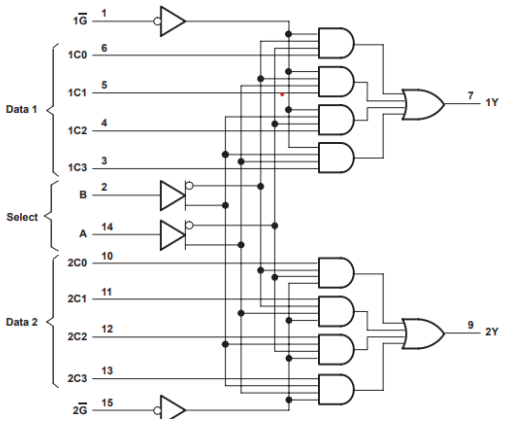
- Module
  - Verilog에서 기본적인 설계 블록
  - 상위 블록에 필요한 기능들을 제공

→ Hierarchical Structure | Design Methodology

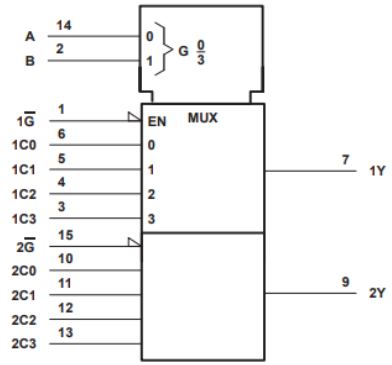
  - In/output의 포트 인터페이스를 제공하고 내부 상황은 은닉시킴

→ abstraction / encapsulation

  - 특정한 object가 만들어지는데 template을 제공함(module invocation)



Symbolization / Encapsulation

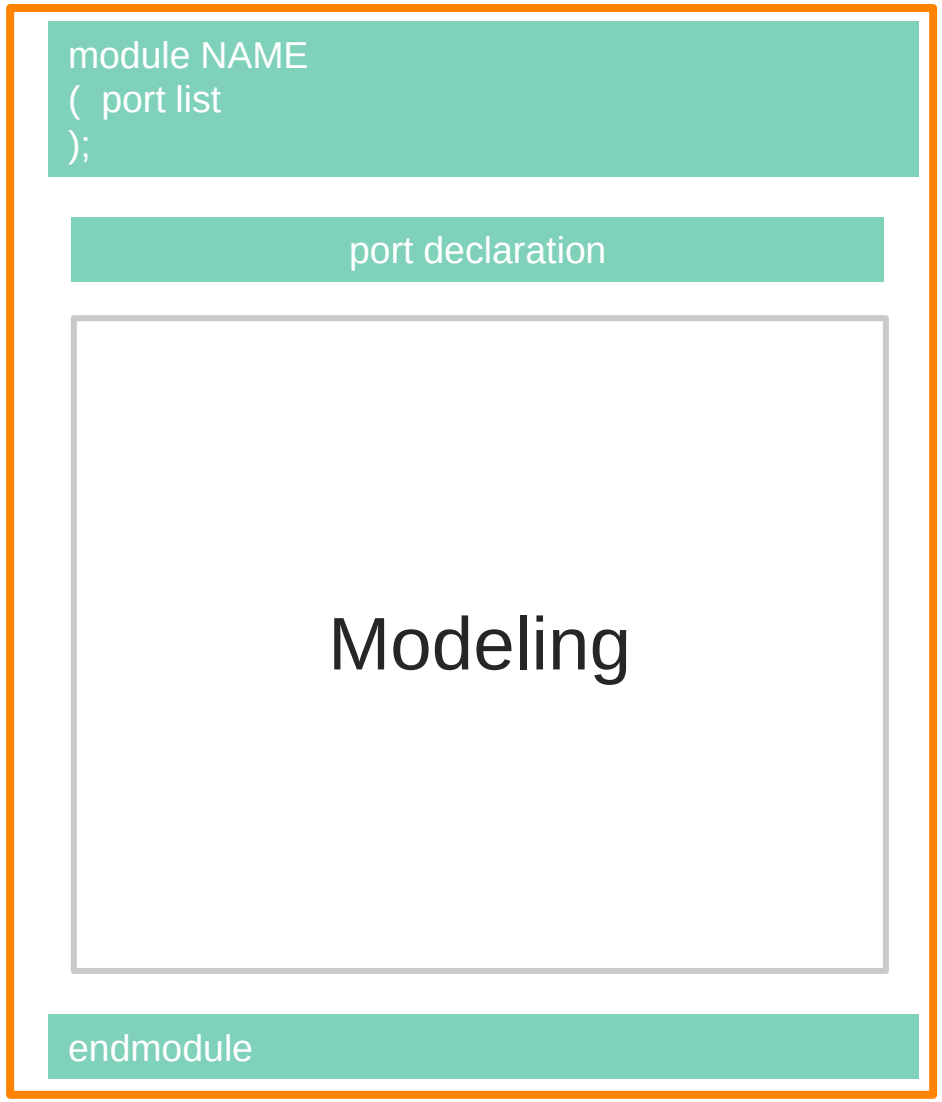


Source: TI, SN54ALS153, Dual 1-of-4 Data Selectors

Modularization / Hardware Descripting

```
/*
Module Name: 4 X 1 multiplexor
Date Created: 20-Aug-2023
Author: William Woo
Logic: out = (~sel1 & ~sel0 & in1) | (~sel1 & sel0 & in2) |
            (sel1 & ~sel0 & in3) | (sel1 & sel0 & in4);
Revision: v0.1
*/

module mux4_to_1 // 4 to 1 multiplexor
(out, // 1bit output
in1,  // 1bit input1
in2,  // 1bit input2
in3,  // 1bit input3
in4,  // 1bit input4
sel1, // 1bit selection1
sel0); // 1bit selection2

// port declaration
output wire out;
input wire in1, in2, in3, in4;
input wire sel0, sel1;

// data flow statement
assign out = (~sel1 & ~sel0 & in1) | (~sel1 & sel0 & in2) |
            (sel1 & ~sel0 & in3) | (sel1 & sel0 & in4);


endmodule // finish describing the module
```

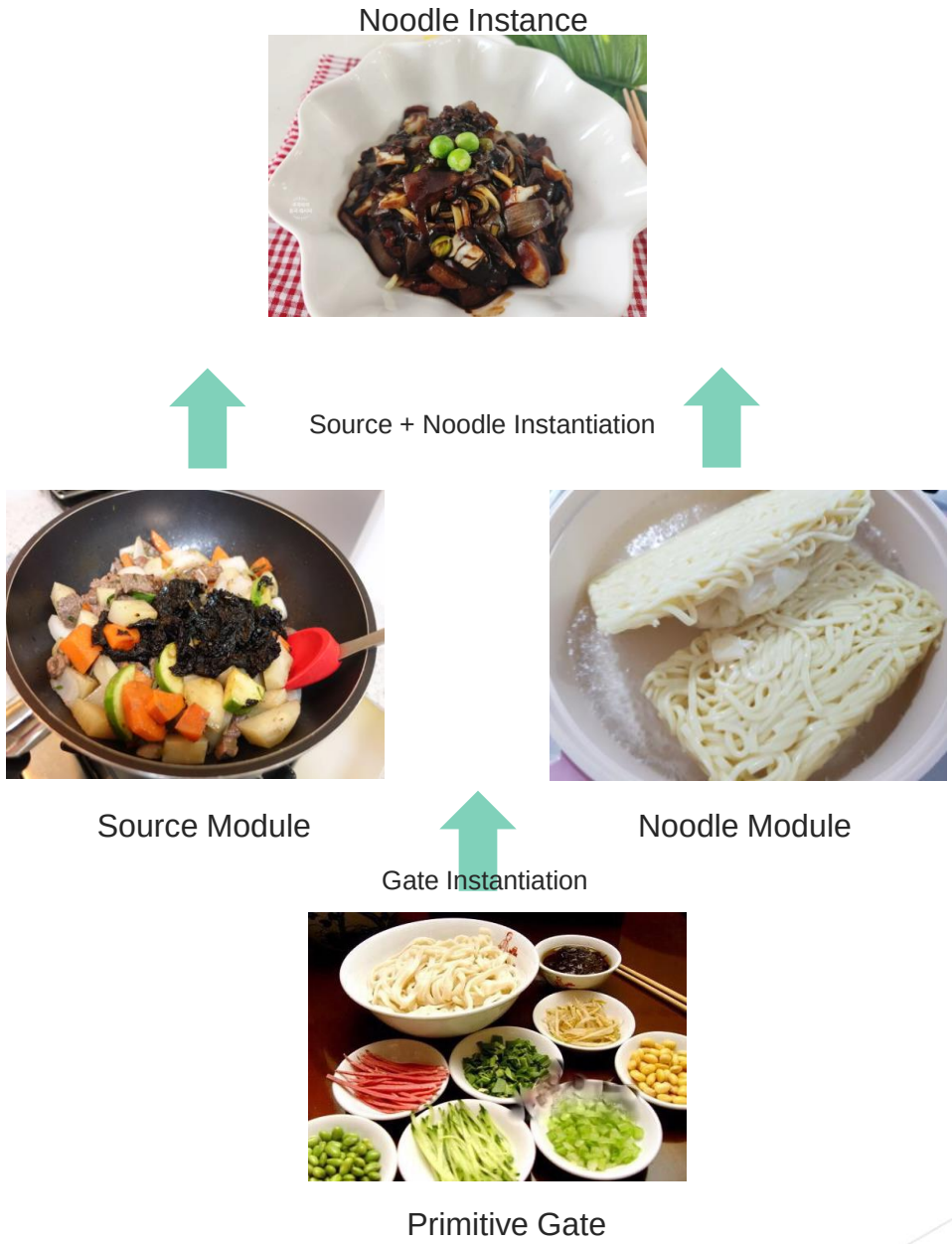
Module: component & skeleton



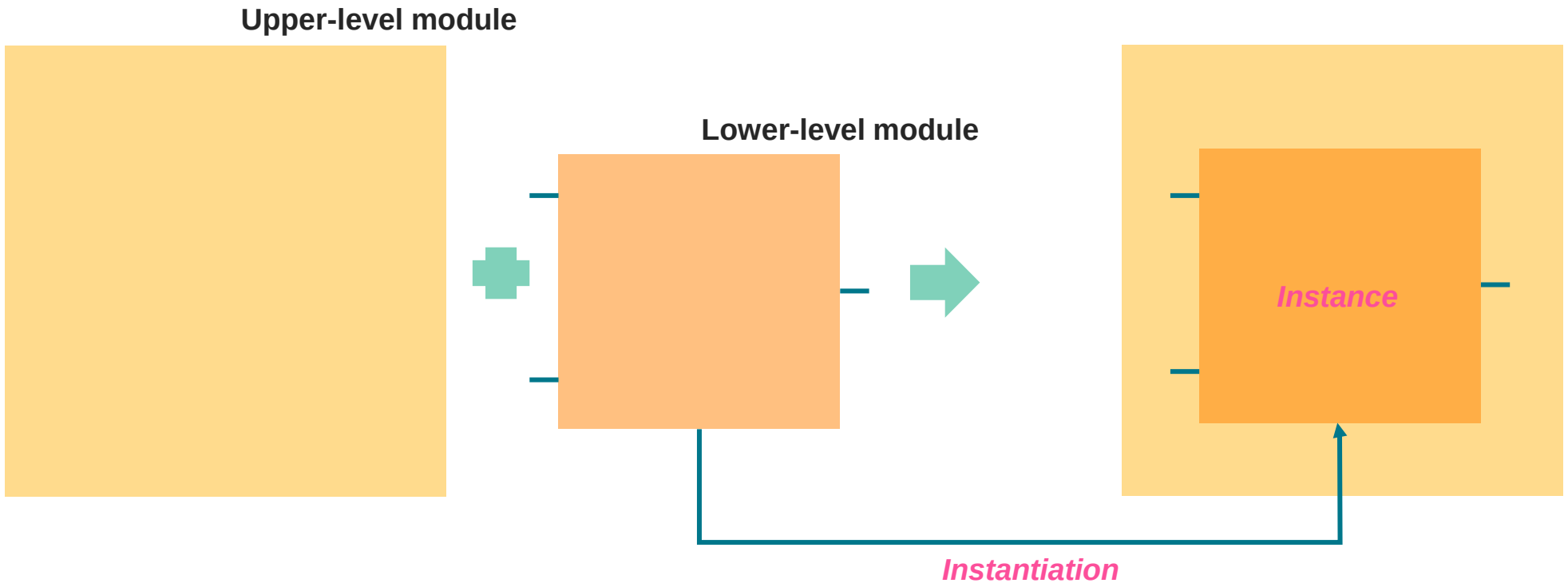

```
1   module module_name (
2
3       // emumerate port list
4       in_a,
5       in_b,
6       out_s
7   );
8
9   // port declaration
10  input wire in_a;
11  input wire in_b;
12  output [3:0] out_s;
13
14  // Data flow statement
15  assign =
16
17  // Behavioral statement
18  // 1. always block
19  always @() begin
20
21  end
22
23  // 2. initial block
24  initial begin
25
26  end
27
28  endmodule
```

Instance: Concept

- Instance
  - Module을 이용하여 실제로 구현된 객체(object)
  - Module → 조리법(template) | Instance → 실제 음식(조립)
  - Instantiation: Module이 Instance로 만들어지는 것. 상위 module에서 하위 module을 계층적으로 불러오는 것(invocation)
  - 실제적인 이름, 변수, 파라미터 및 I/O Interface를 가짐

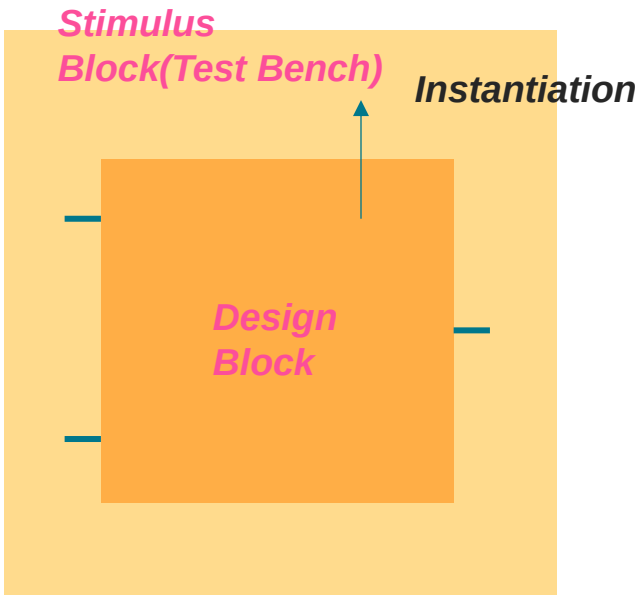


Instantiation



3. Testbench

Testbench



- Design Block
  - 설계 모듈 블록
  - DUT = Design Under Test
- Stimulus Block
  - DUT를 테스트하고 로직의 결과를 확인하는 블록
  - Testbench 라고도 불림

Testbench by example

```
1   module stimulus;
2   reg clk; // Input
3   reg reset; // Input
4   wire [3:0] q; // Output
5
6   ripple_carry_counter r1 (.q(q), .clk(clk), .reset(reset));
7   initial
8   clk = 1'b0; // Set clk to 0
9   always
10  #5 clk = ~clk; // Toggle clk every 5 time units
11  initial
12  begin
13  reset = 1'b1;
14  #25 reset = 1'b0;
15  #180 reset = 1'b1;
16  #10 reset = 1'b0;
17  #20 $finish;
18  end
19
20  initial
21  $monitor($time, "Output q = %d", q);
22  endmodule
```

DUT instantiation

Clock Generation

Apply stimulus

Check Result

LAB

1. Download Datasheet
2. DUT design(Gate Level)
3. Stimulus design(no clock)
4. Run simple simulation(single step)