Hierarchical Design and Components

Announcements

- Homework #5 due today
- Homework #6 posted
- Please complete homework group evaluations by Sunday.

Modular Design

- Methodology for designing complex systems
  - System partitioned into simpler modules
  - Modules are interconnected to accomplish system's overall function
  - Each module task is defined in a way that allows it to be independently designed and verified
  - Design effort is simplified by focusing on one module at a time

Hierarchical Design

- Top-Level Design is an integration of modules
  - Each module may be the integration of smaller modules
  - The lowest level consists of behaviorally defined entities
- Also known as Top-Down Design

Entity A

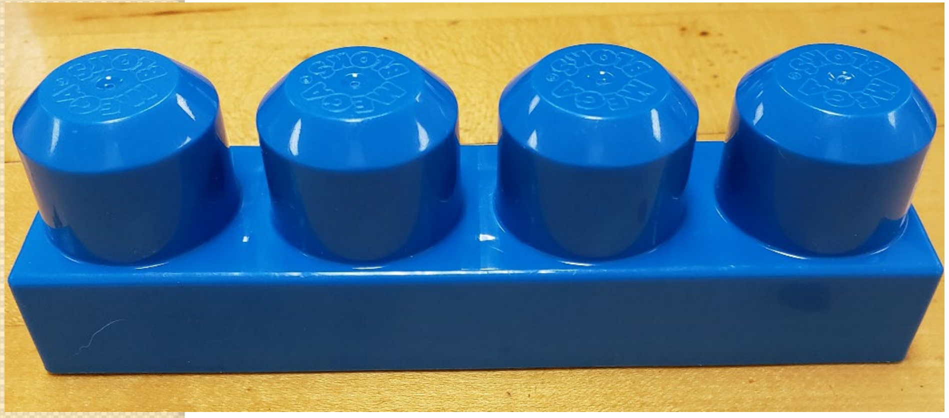
Entity B

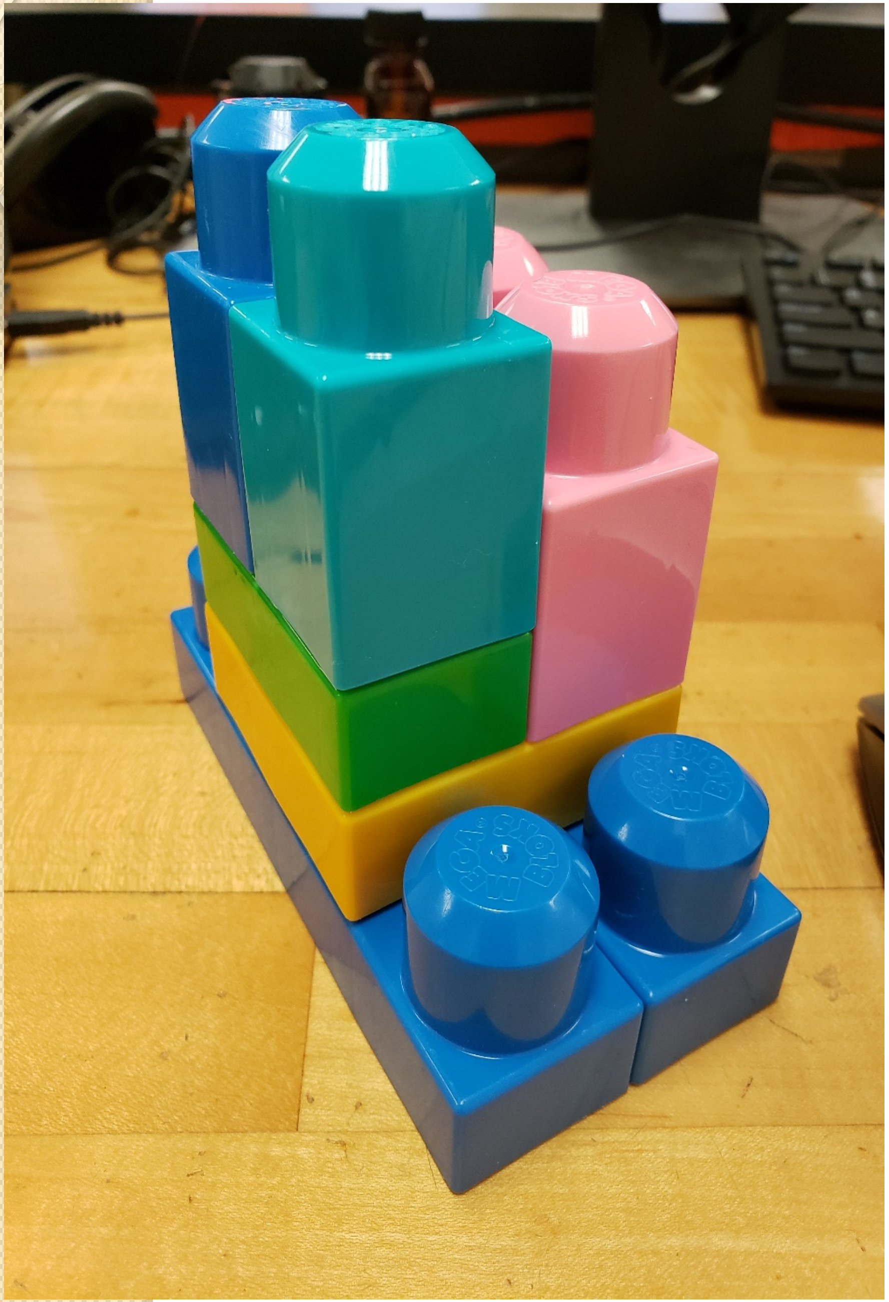
Entity D

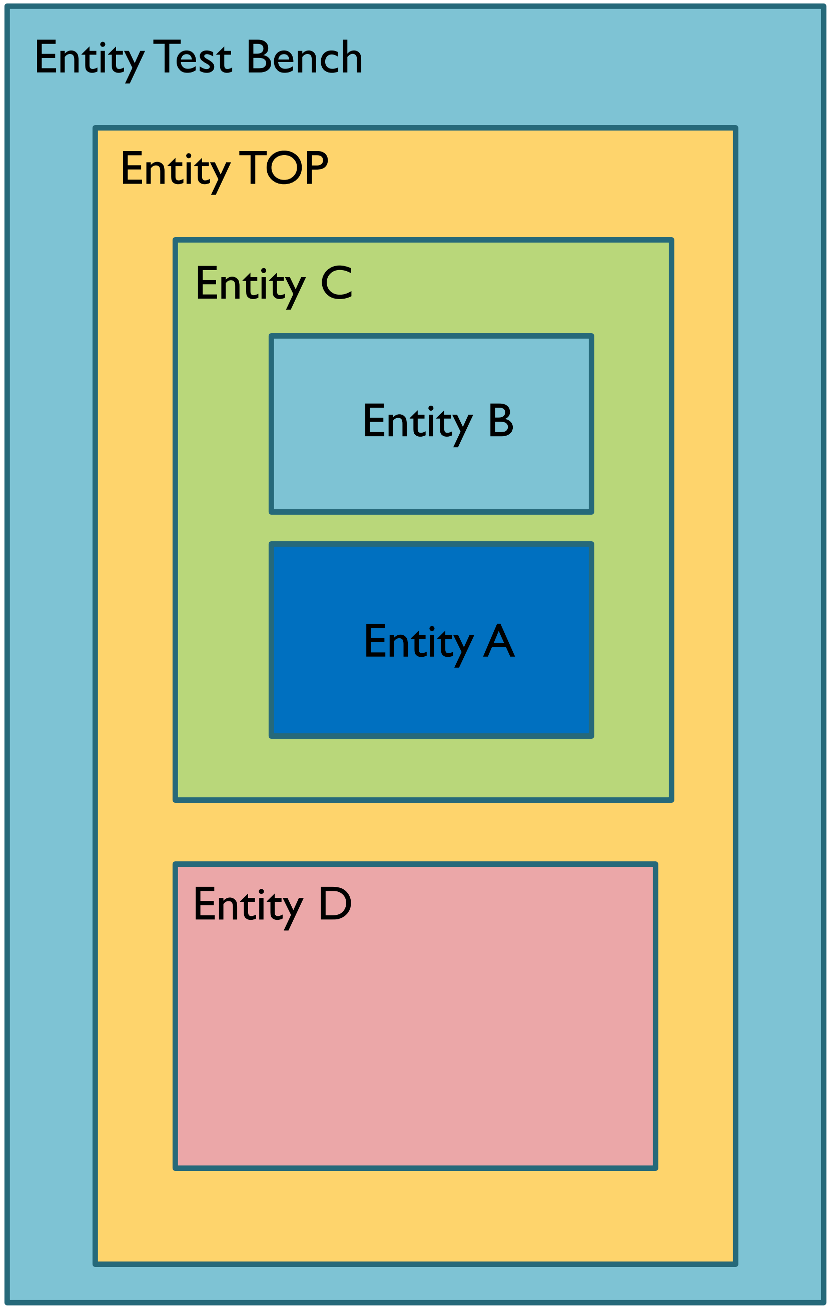
Entity C

Entity TOP

Entity TB






Entity Test Bench
Entity TOP
Entity C
Entity B
Entity A
Entity D

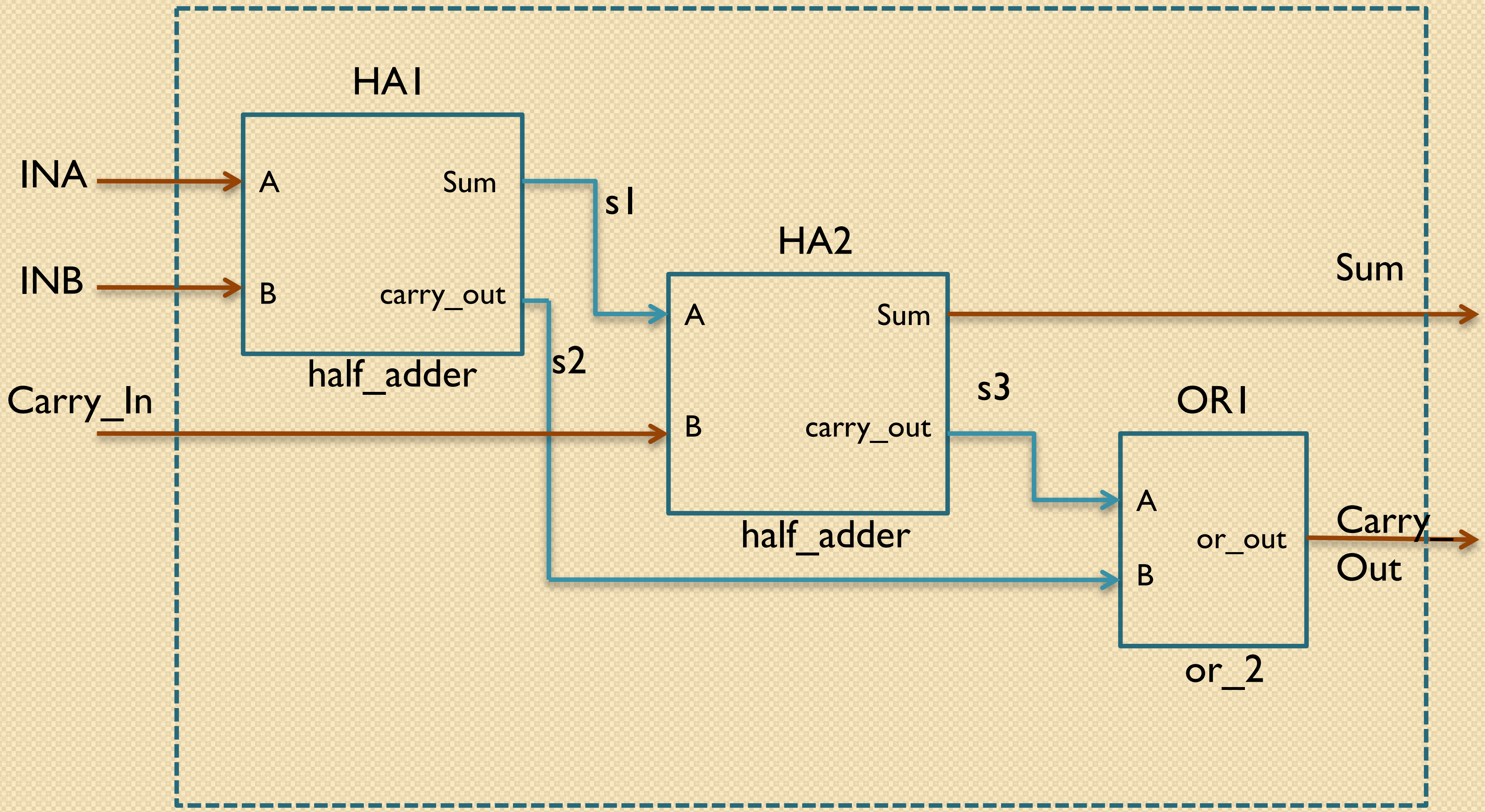
Hierarchical Design in VHDL

- Advantages of a well partitioned hierarchical design
  - Design management is easier
  - Modules can be designed and verified by different engineers
  - Maximum re-use of modules is possible
  - Design easier to read and comprehend
  - Verification is simpler
  - Enhanced portability

Hierarchical Design in VHDL

- Modules at each level are known as COMPONENTS
- Components are interconnected with signals
- A component may be used more than once in a level
- Levels
  - Lowest level modules are behavioral
  - Top level is structural
  - Other levels may be mixed

Full Adder Example



Full Adder Example

- The Full Adder entity has
  - Three inputs; INA, INB, Carry_in
  - Two outputs; Sum, Carry_out
- The Full Adder architecture is comprised of
  - Two half adders
  - A 2-input OR module
  - Three interconnect signals; s1, s2, s3

Half-Adder component

```
ENTITY half_adder IS
    PORT( a, b              :  IN STD_LOGIC;
              sum, carry_out : OUT STD_LOGIC);
    END half_adder;

ARCHITECTURE dataflow OF half_adder IS
BEGIN
    sum <= a XOR b;
    carry_out <= a AND b;
END dataflow;
```

Two-Input OR component

```
ENTITY or_2 IS
   PORT( a, b          : IN STD_LOGIC;
            or_out     : OUT STD_LOGIC);
   END or_2;

ARCHITECTURE dataflow OF or_2 IS
BEGIN
   or_out <= a OR b;
END dataflow;
```

Top Level – Full Adder

```
ENTITY full_adder IS
    PORT (INA, INB, Carry_in            : IN STD_LOGIC:
            Sum, Carry_out              : OUT STD_LOGIC);
    END full_adder;
```

Top Level – Full Adder

```
ARCHITECTURE structure OF full_adder IS
    SIGNAL s1, s2, s3   : std_logic;
    COMPONENT half_adder IS
        PORT( a, b                    :  IN STD_LOGIC;
             sum, carry_out : OUT STD_LOGIC);
        END COMPONENT;
    COMPONENT or_2 IS
        PORT( a, b          : IN STD_LOGIC;
             or_out     : OUT STD_LOGIC);
        END COMPONENT;
    BEGIN
      HA1 : half_adder
        PORT MAP(a                =>INA,
                   b              =>INB,
                   sum            => s1,
                   carry_out      => s2);
```

Top Level – Full Adder

```
      HA2 : half_adder
            PORT MAP(a                    =>s1,
                     b                    =>Carry_in,
                     sum                  => sum,
                     carry_out            => s3);
      OR1 : or_2
            PORT MAP(a                    =>s3,
                     b                    =>s2,
                     or_out               => carry_out);
END structure;
```

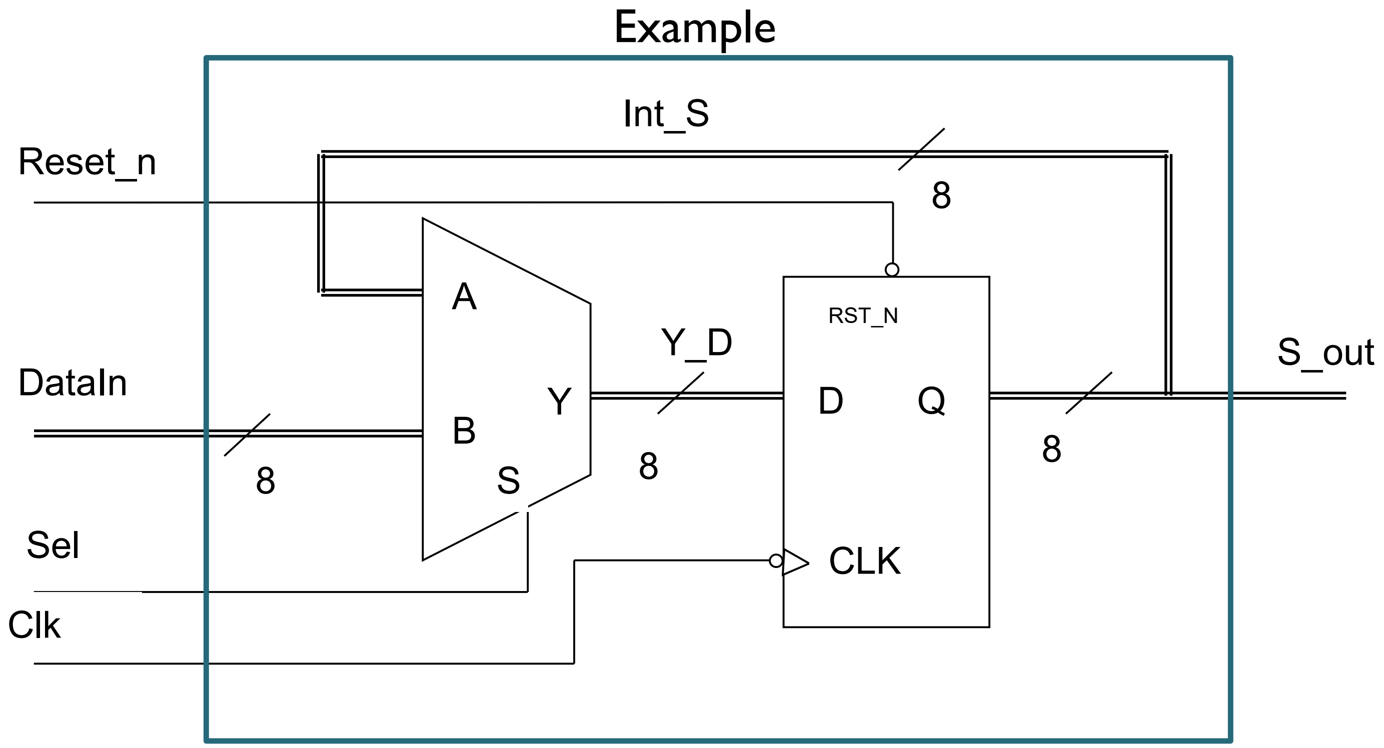
Terminology

- HA1, HA2 and OR1 are *instantiations*
- *PORT MAP* describes how a design entity is connected in a larger structure
  - *Formal* port is on the left. This is the port of the component's entity
  - *Actual* is on the right. This is the signal or port that is connected in the architecture

Port Map

- Components instantiations are treated as concurrent statements
  - the inputs to component represent the sensitivity list
  - whenever there is an event on one of its inputs the component is evaluated
  - component instantiations cannot be included in a process

Another example



```
ENTITY mux2to1
   PORT(A,B : IN STD_LOGIC_VECTOR(7 downto 0);
            S      : IN STD_LOGIC;
            Y   : OUT STD_LOGIC_VECTOR(7 downto 0));
END  mux2to1;
```

```
ENTITY Reg8 IS
   PORT(D   : IN STD_LOGIC_VECTOR(7 downto 0);
          clk, rst_n  : IN STD_LOGIC;
          Q  : OUT STD_LOGIC_VECTOR(7 downto 0));
 END Reg8
```