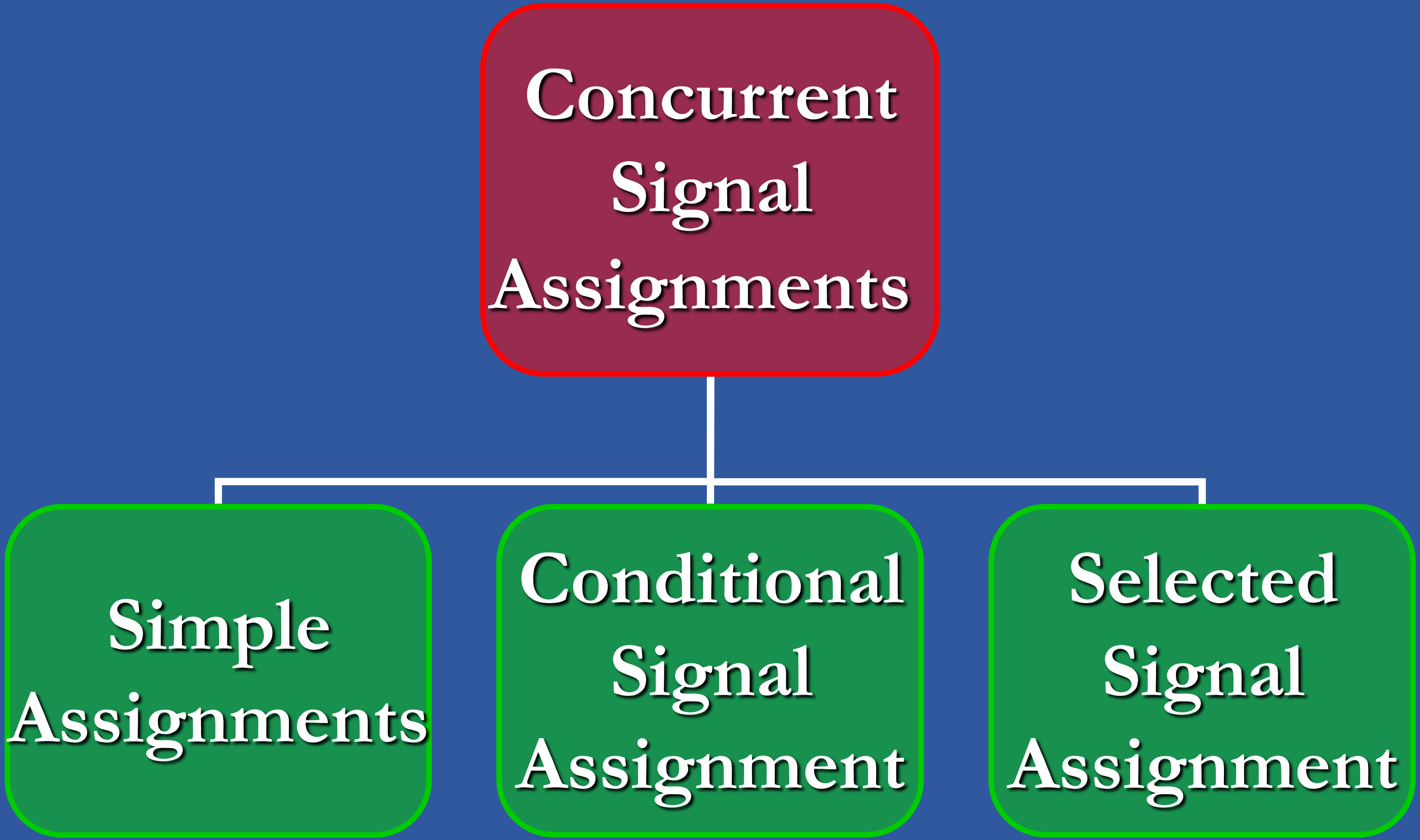
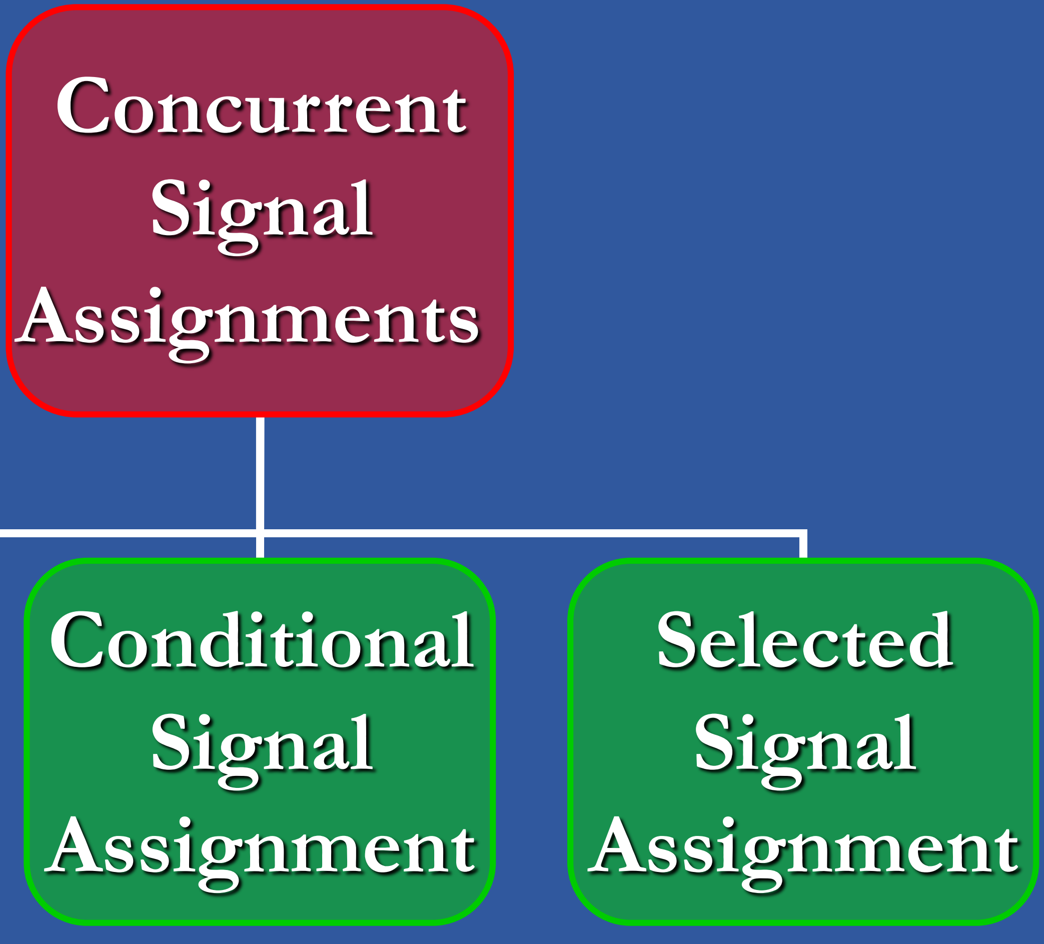
Chapter 4
Concurrent Constructs for RT Level Descriptions

Concurrent Constructs for RT Level Descriptions

Concurrent Signal Assignments


Concurrent Signal Assignments
Simple Assignments
Conditional Signal Assignment
Selected Signal Assignment

Simple Assignments

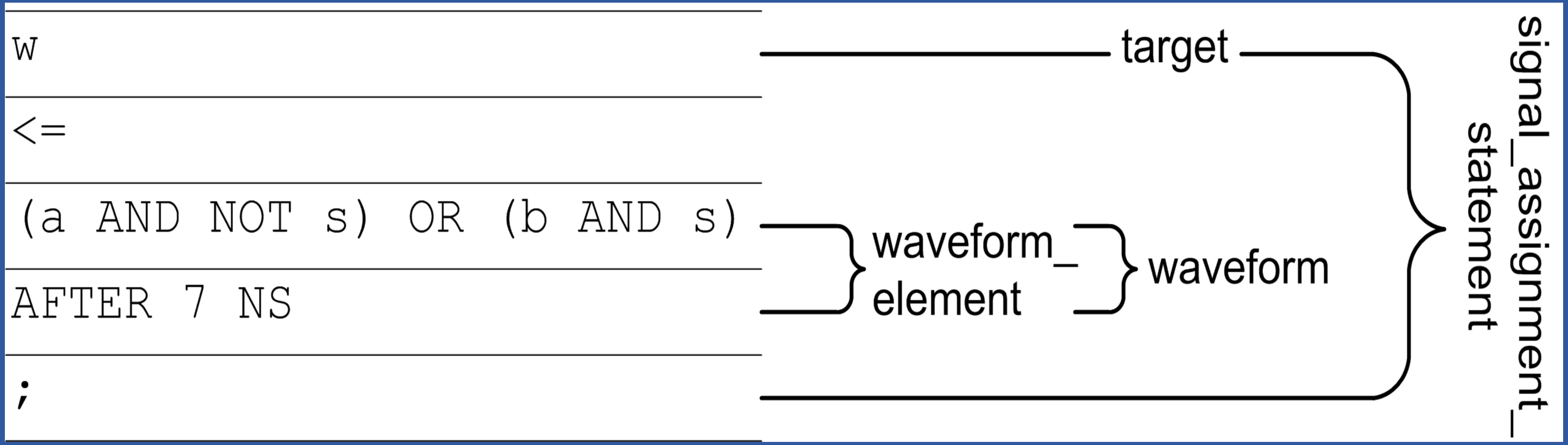

Concurrent Signal Assignments
Simple Assignments
Conditional Signal Assignment
Selected Signal Assignment

Simple Assignments

```
ENTITY multiplexer IS
   PORT (a, b, s : IN BIT; w : OUT BIT);
END ENTITY;
--
ARCHITECTURE expression OF multiplexer IS
BEGIN
   w <= (a AND NOT s) OR (b AND s) AFTER 7 NS;
END ARCHITECTURE expression;
```

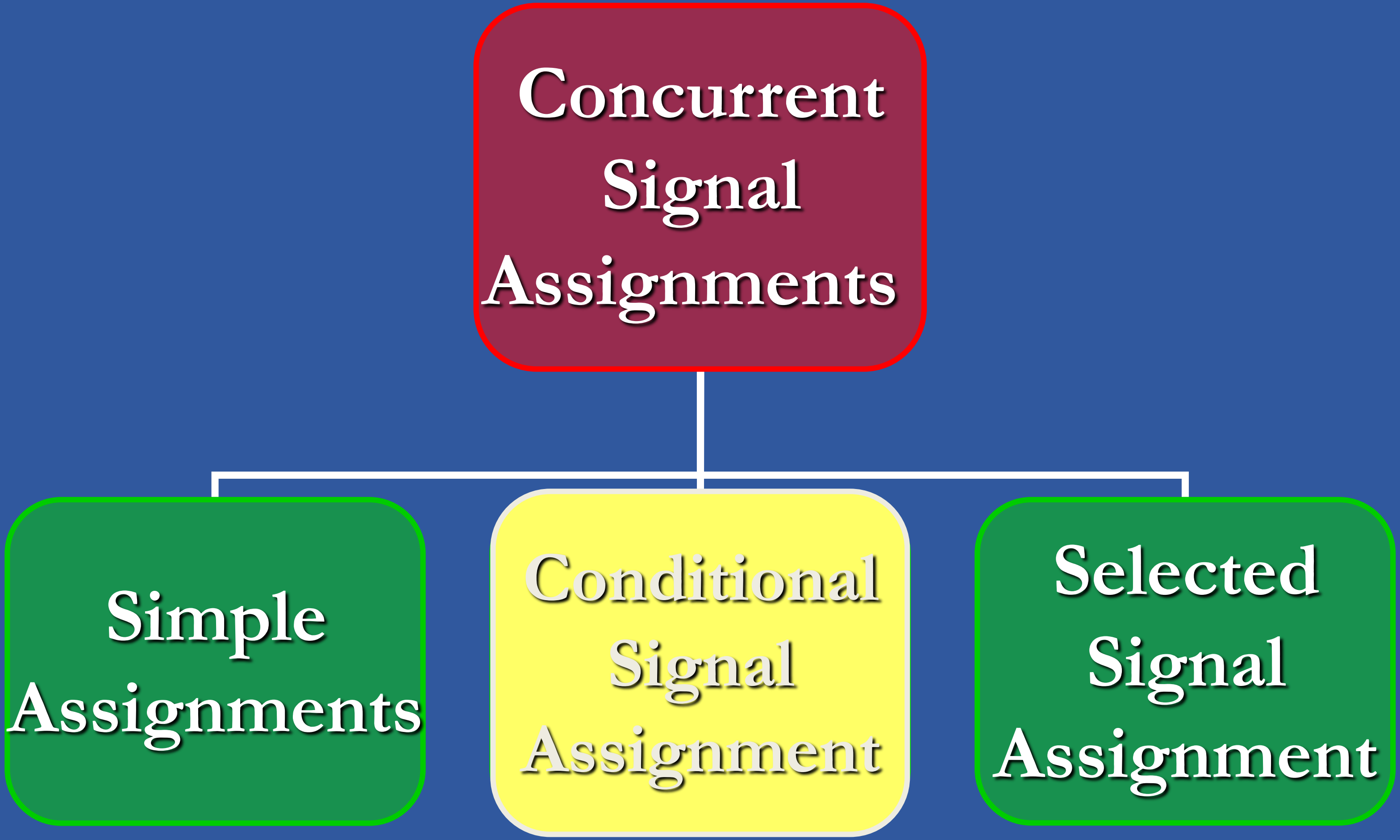
- Concurrent Signal Assignment

Simple Assignments



- **Signal Assignment Syntax Structure**

Conditional Signal Assignment

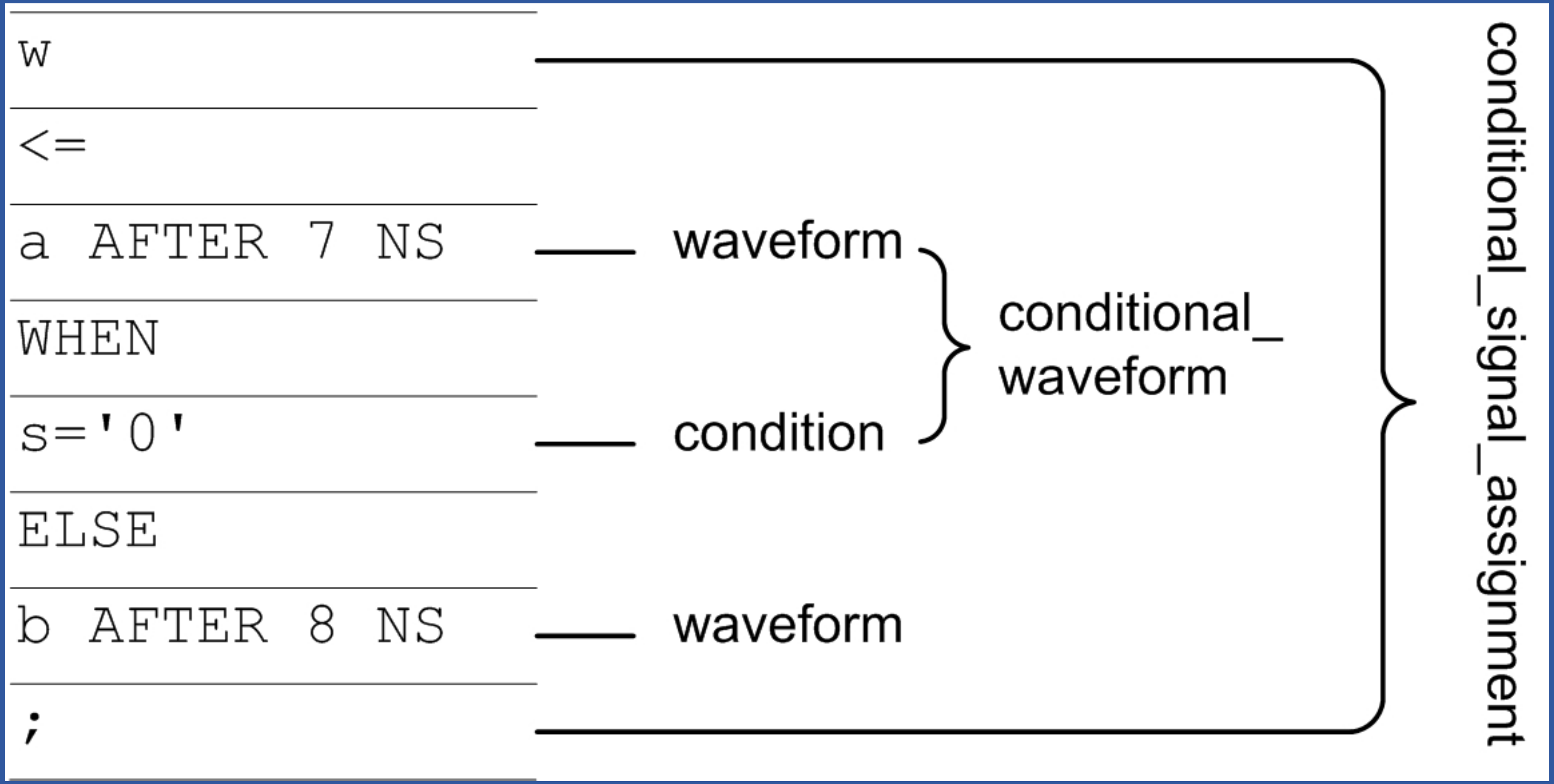

Concurrent Signal Assignments
Simple Assignments
Conditional Signal Assignment
Selected Signal Assignment

Conditional Signal Assignment

```
ARCHITECTURE conditional OF multiplexer IS
BEGIN
   w <= a AFTER 7 NS WHEN s='0' ELSE
        b AFTER 8 NS;
END ARCHITECTURE conditional;
```

- Using Conditional Signal Assignment

Conditional Signal Assignment



- **Syntax Structure of Conditional Signal Assignments**

Conditional Signal Assignment

```
ENTITY flipflop IS
   PORT (reset, din, clk : IN BIT; qout : OUT BIT);
END ENTITY;
--
ARCHITECTURE async_reset OF flipflop IS
BEGIN
   qout <= '0' WHEN reset = '1' ELSE
           din WHEN clk'EVENT AND clk = '1';
END ARCHITECTURE async_reset;
```

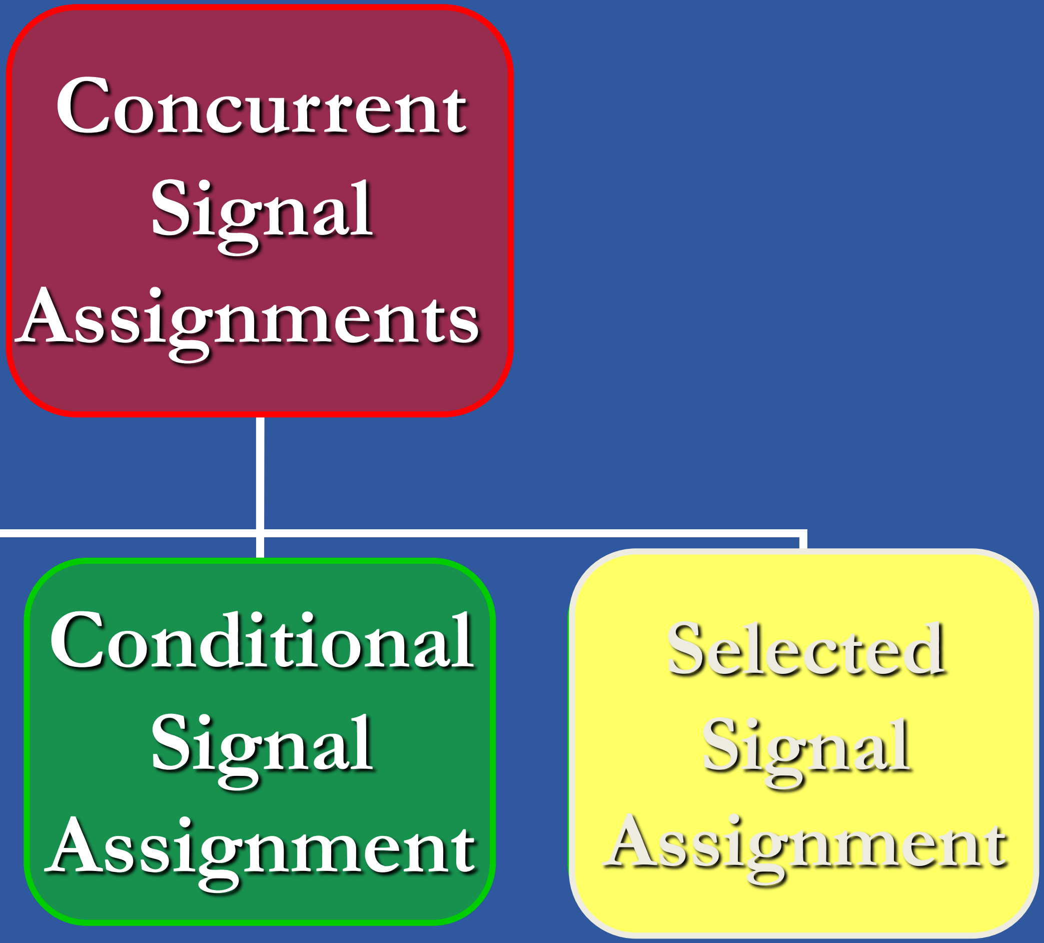
- Flip-Flop Using Conditional Signal Assignment

Conditional Signal Assignment

```
ENTITY latch IS
   PORT (din, clk : IN BIT; qout : OUT BIT);
END ENTITY;
--
ARCHITECTURE assign OF latch IS
BEGIN
   qout <= din WHEN clk = '1';
END ARCHITECTURE assign;
--
ARCHITECTURE unaffect OF latch IS
BEGIN
   qout <= din WHEN clk='1' ELSE UNAFFECTED;
END ARCHITECTURE unaffect;
```

- Conditional Signal Assignment Used in Latch Description

Selected Signal Assignment


Concurrent Signal Assignments
Simple Assignments
Conditional Signal Assignment
Selected Signal Assignment

Selected Signal Assignment

```
ENTITY binary3to8decoder IS
   PORT (bin_in : IN BIT_VECTOR (2 DOWNTO 0);
         en : IN BIT;
         dcd_ou : OUT BIT_VECTOR (0 TO 7));
END ENTITY binary3to8decoder;
--
ARCHITECTURE selected OF binary3to8decoder IS
   SIGNAL tmp : BIT_VECTOR(dcd_ou'RANGE);
   ..............
   ..............
   ..............
```

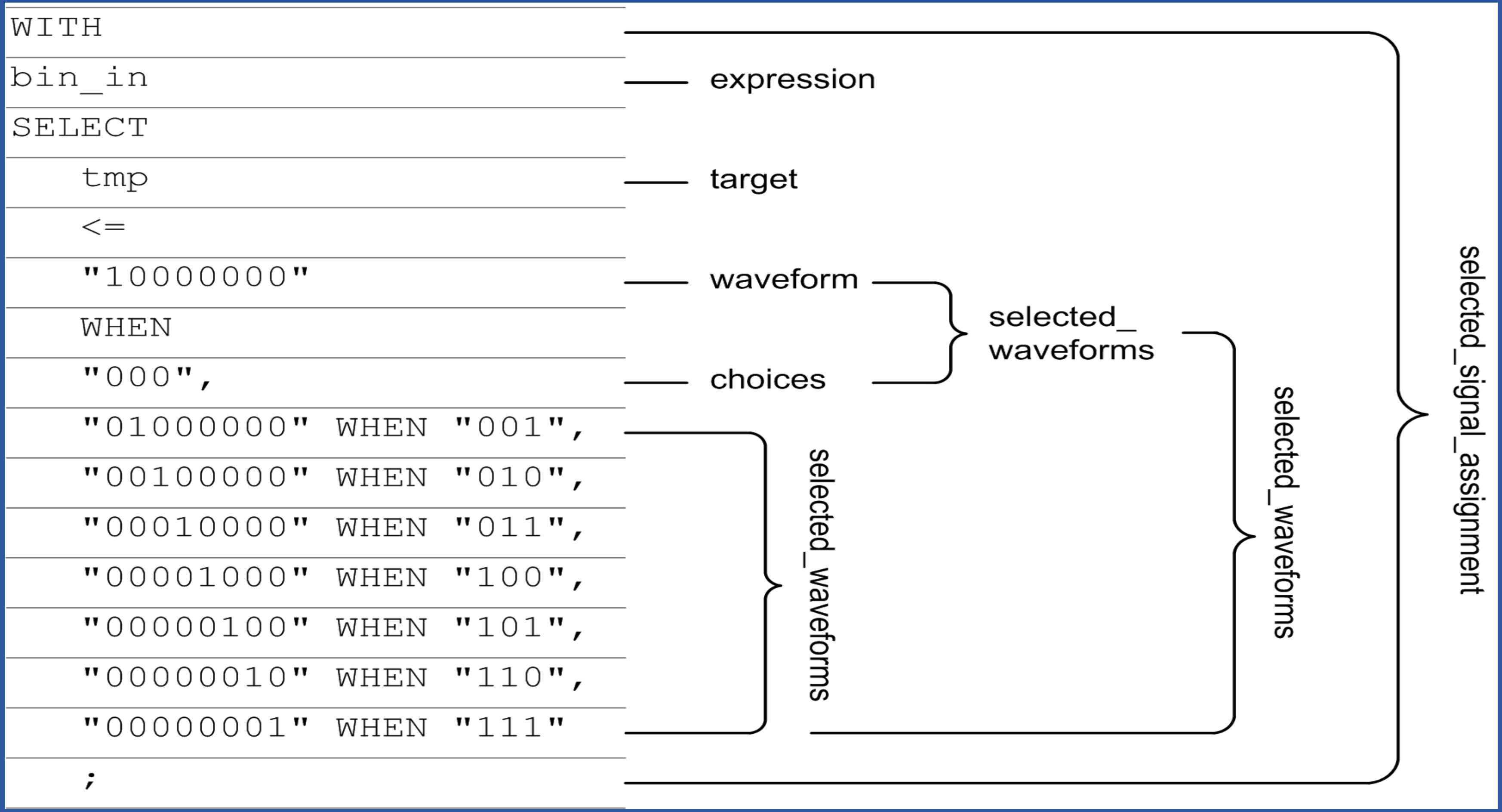
- Using a Selected Signal Assignment

Selected Signal Assignment

```
. . . . . . . . . . . . .
BEGIN
   WITH bin_in SELECT
      tmp <= "10000000" WHEN "000",
             "01000000" WHEN "001",
             "00100000" WHEN "010",
             "00010000" WHEN "011",
             "00001000" WHEN "100",
             "00000100" WHEN "101",
             "00000010" WHEN "110",
             "00000001" WHEN "111";
   dcd_ou <= tmp WHEN en = '1' ELSE (OTHERS =>
'0');
END ARCHITECTURE selected;
```

- Using a Selected Signal Assignment (Continued)

Selected Signal Assignment



```
WITH
bin_in
SELECT
    tmp
    <=
    "10000000"
    WHEN
    "000",
    "01000000" WHEN "001",
    "00100000" WHEN "010",
    "00010000" WHEN "011",
    "00001000" WHEN "100",
    "00000100" WHEN "101",
    "00000010" WHEN "110",
    "00000001" WHEN "111"
    ;
```

- Syntax of Selected Signal Assignment

Selected Signal Assignment

```
ENTITY ssd IS
   PORT (bcd : IN BIT_VECTOR (3 DOWNTO 0);
         display : OUT BIT_VECTOR (1 TO 7));
END ENTITY;
--
.............
.............
.............
```

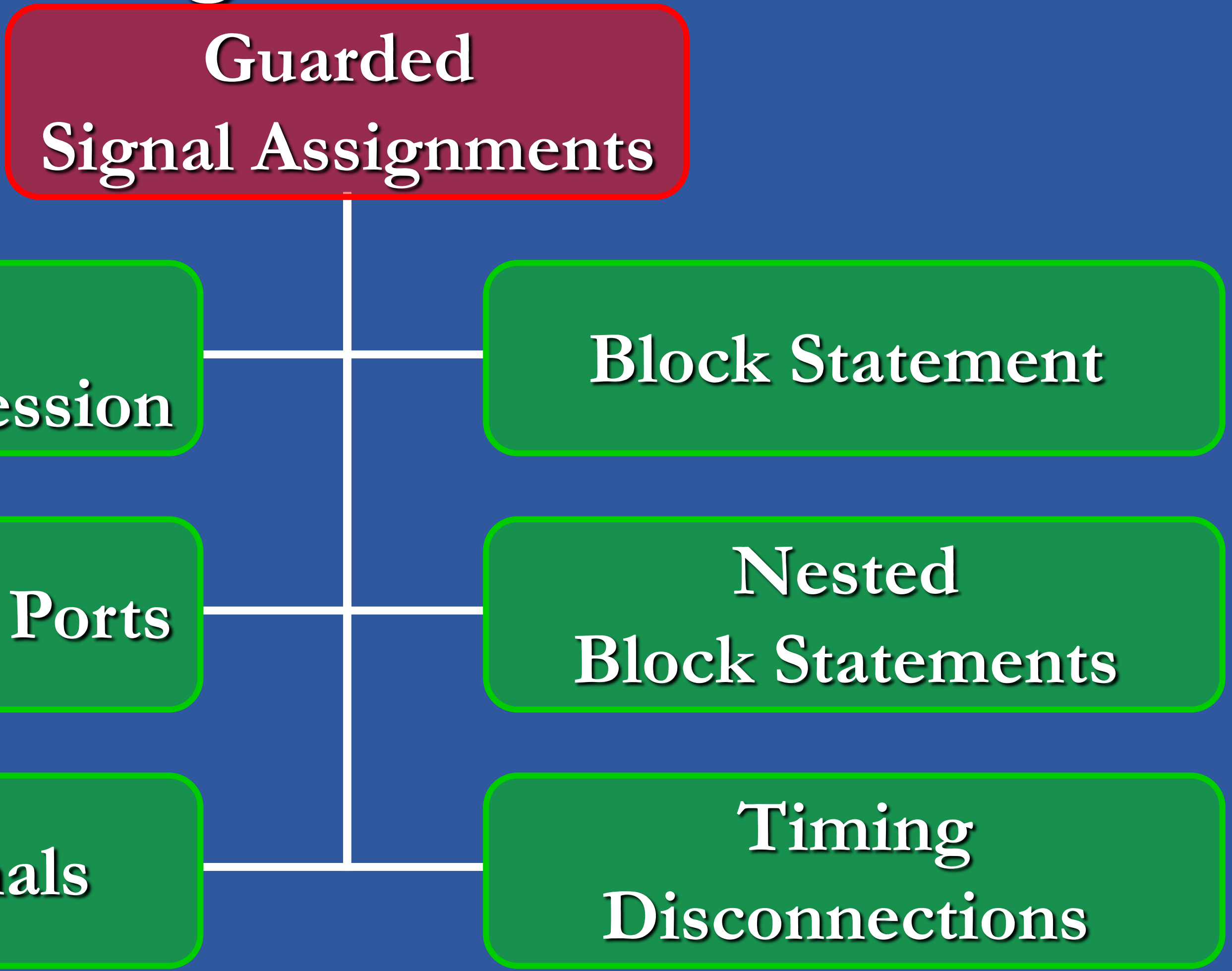
- Selected Signal Assignment with Others

Selected Signal Assignment

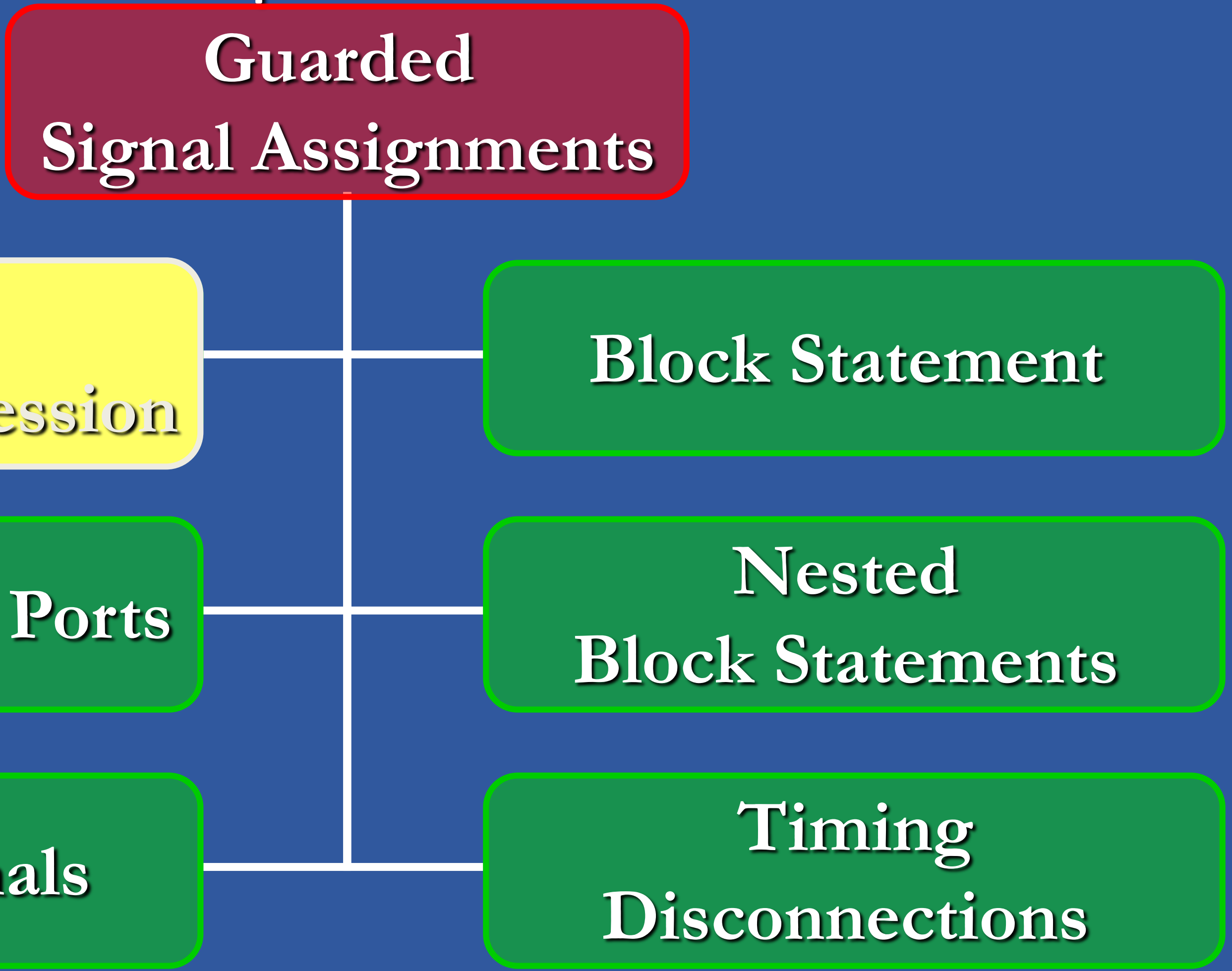
```
ARCHITECTURE selected OF ssd IS
BEGIN
   WITH bcd SELECT
      display <= "1111110" WHEN "0000",
                 "0110000" WHEN "0001",
                 "1101101" WHEN "0010",
                 "1111001" WHEN "0011",
                 "0110011" WHEN "0100",
                 "1101101" WHEN "0101",
                 "1011111" WHEN "0110",
                 "1110000" WHEN "0111",
                 "1111111" WHEN "1000",
                 "1111101" WHEN "1001",
                 "1001111" WHEN OTHERS;
END ARCHITECTURE selected;
```

- Selected Signal Assignment with Others (Continued)

Guarded Signal Assignments



GUARD Signal and Expression

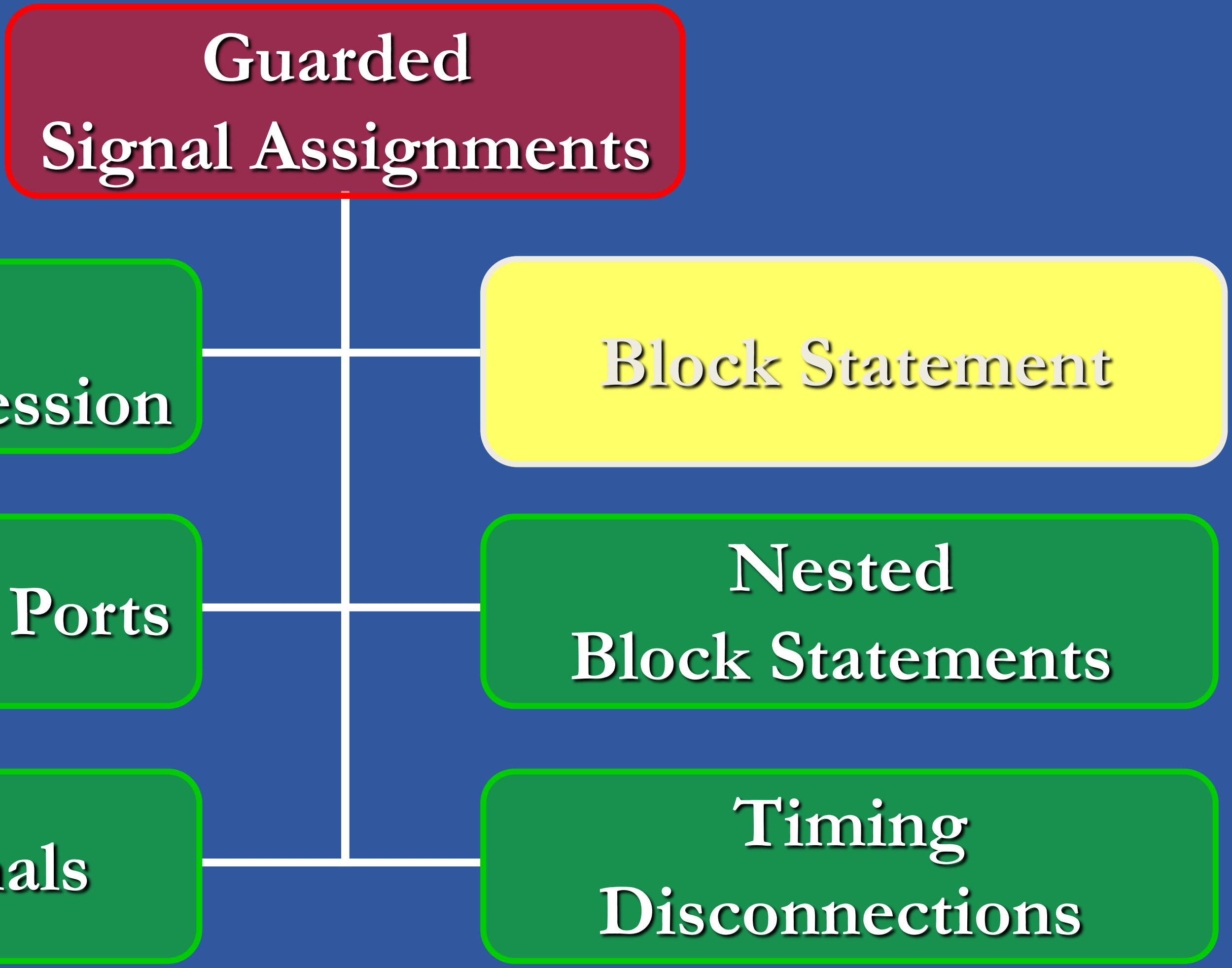


GUARD Signal and Expression

```
ARCHITECTURE explicit OF flipflop IS
   SIGNAL GUARD : BOOLEAN;
BEGIN
   GUARD <= clk = '1' AND NOT clk'STABLE;
   qout <= GUARDED '0' WHEN reset = '1' ELSE din;
END ARCHITECTURE explicit;
```

- Explicit GUARD Signal

Block Statement

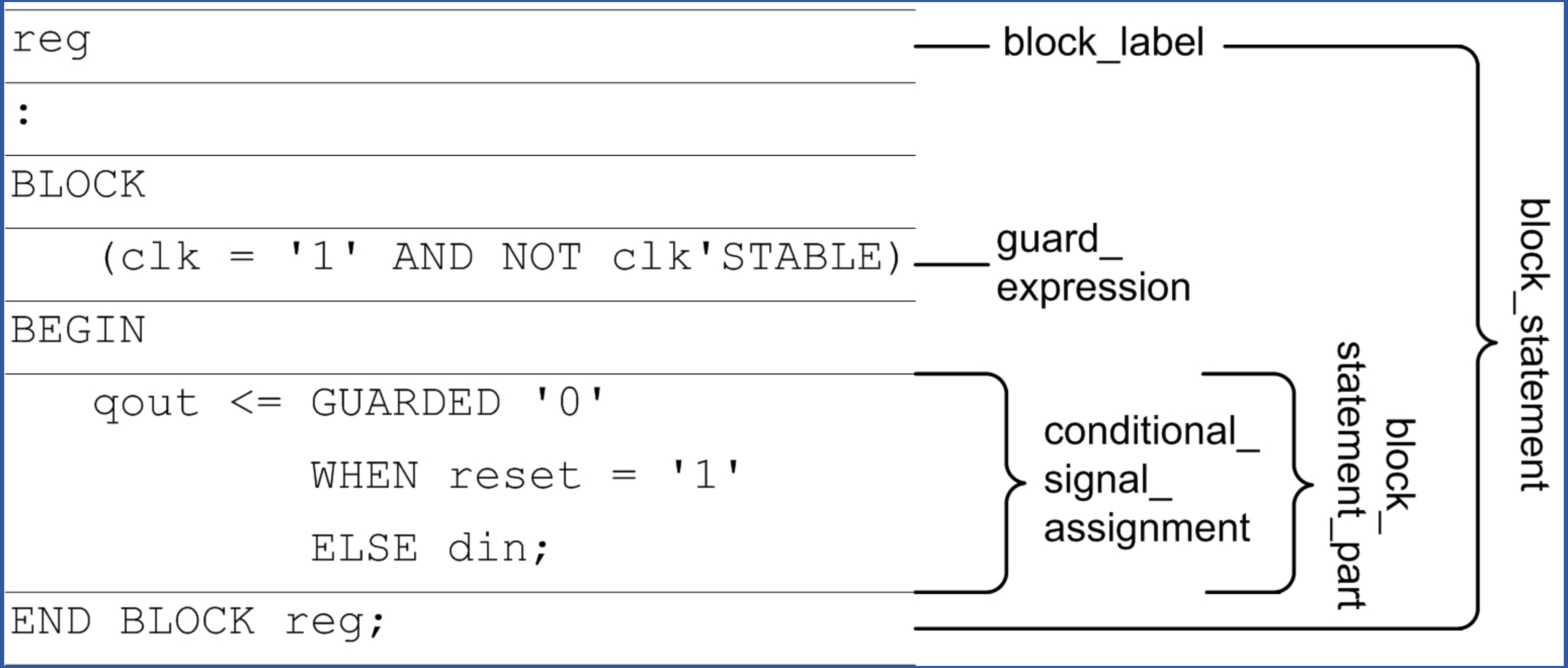


Block Statement

```
ARCHITECTURE blocking OF flipflop IS
BEGIN
   reg: BLOCK (clk = '1' AND NOT clk'STABLE) BEGIN
            qout <= GUARDED '0' WHEN reset = '1'
                    ELSE din;
        END BLOCK reg;
END ARCHITECTURE blocking;
```

- Block Statement with Guard Expression

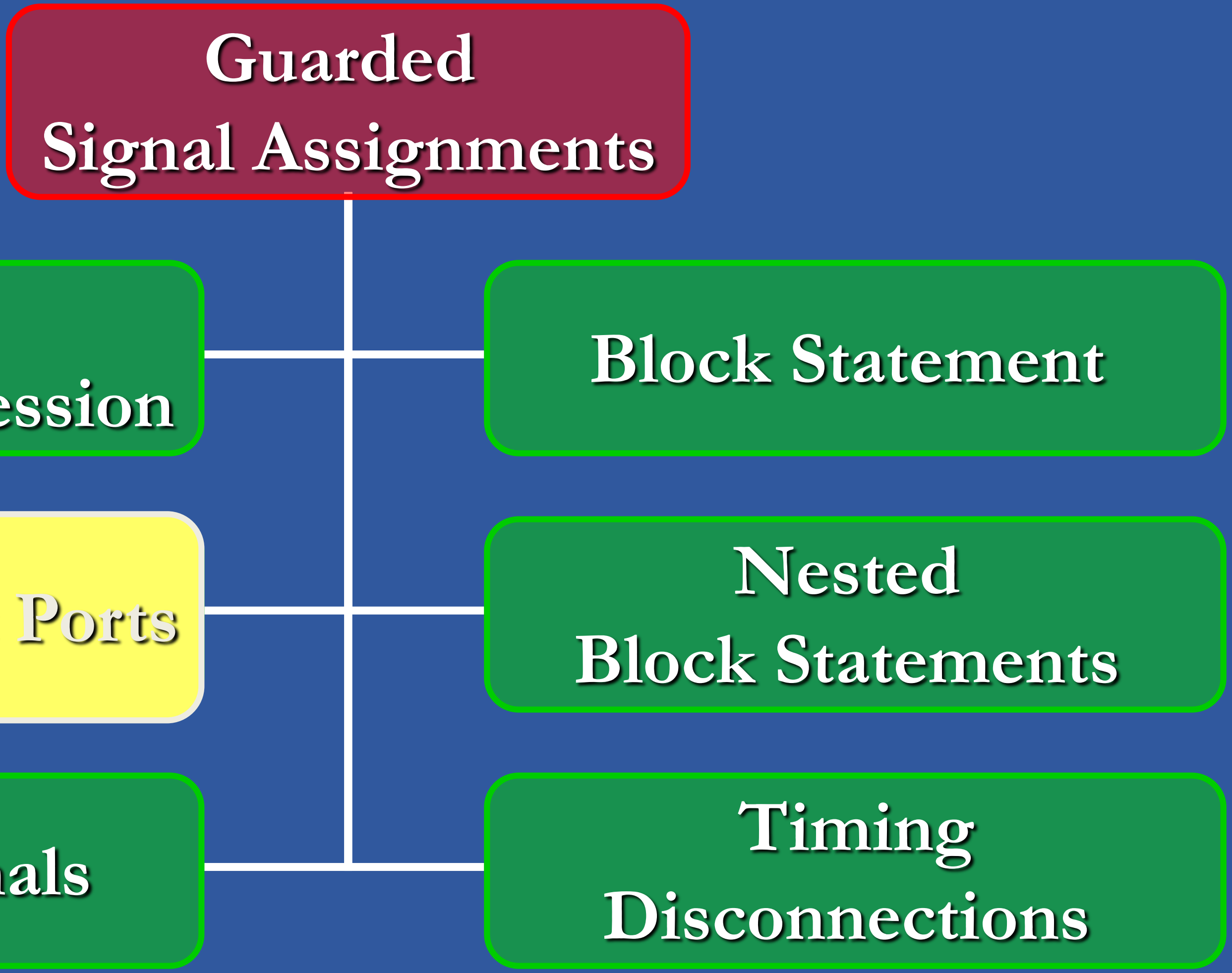
Block Statement



```
reg
:
BLOCK
    (clk = '1' AND NOT clk'STABLE)
BEGIN
    qout <= GUARDED '0'
            WHEN reset = '1'
            ELSE din;
END BLOCK reg;
```

- **Block Statement Syntax**

Block Statement Ports

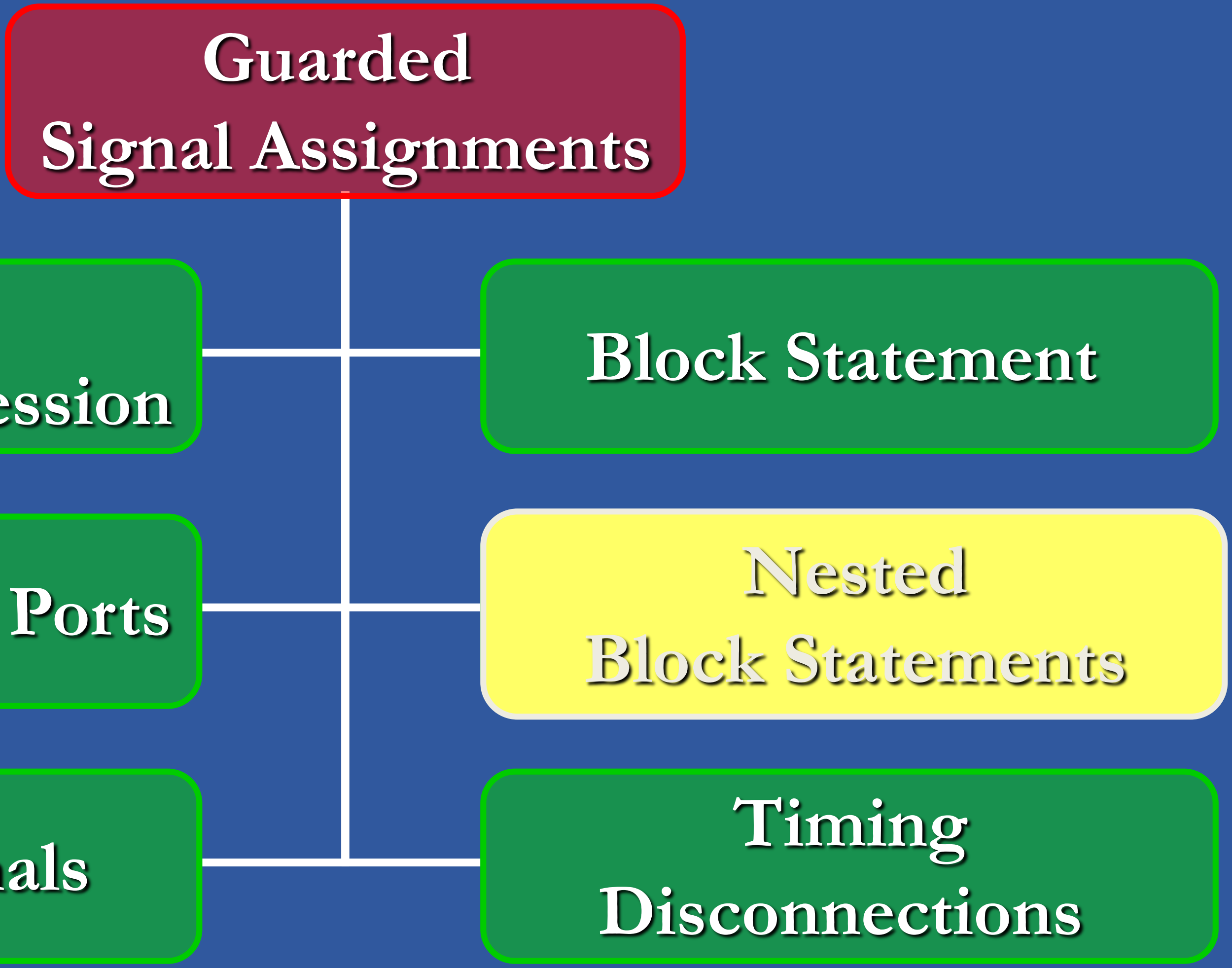


Block Statement Ports

```
ARCHITECTURE blockport OF flipflop IS
BEGIN
   reg: BLOCK (clk = '1' AND NOT clk'STABLE) IS
      PORT (r, d, c : IN BIT; q : OUT BIT);
      PORT MAP (reset, din, clk, qout);
   BEGIN
      q <= GUARDED '0' WHEN r = '1' ELSE d;
   END BLOCK reg;
END ARCHITECTURE blockport;
```

- Block Statement with Ports

Nested Block Statements

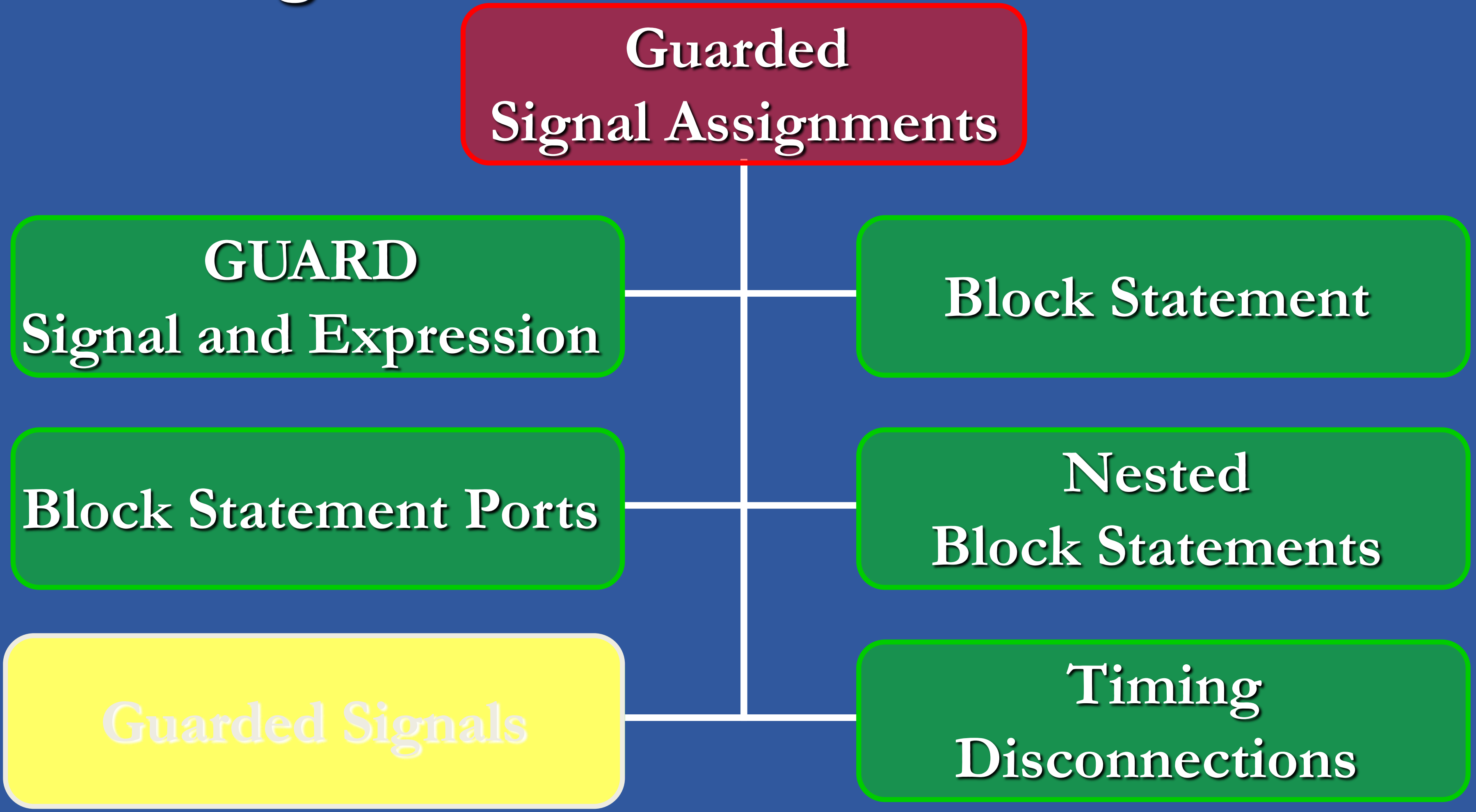


Nested Block Statements

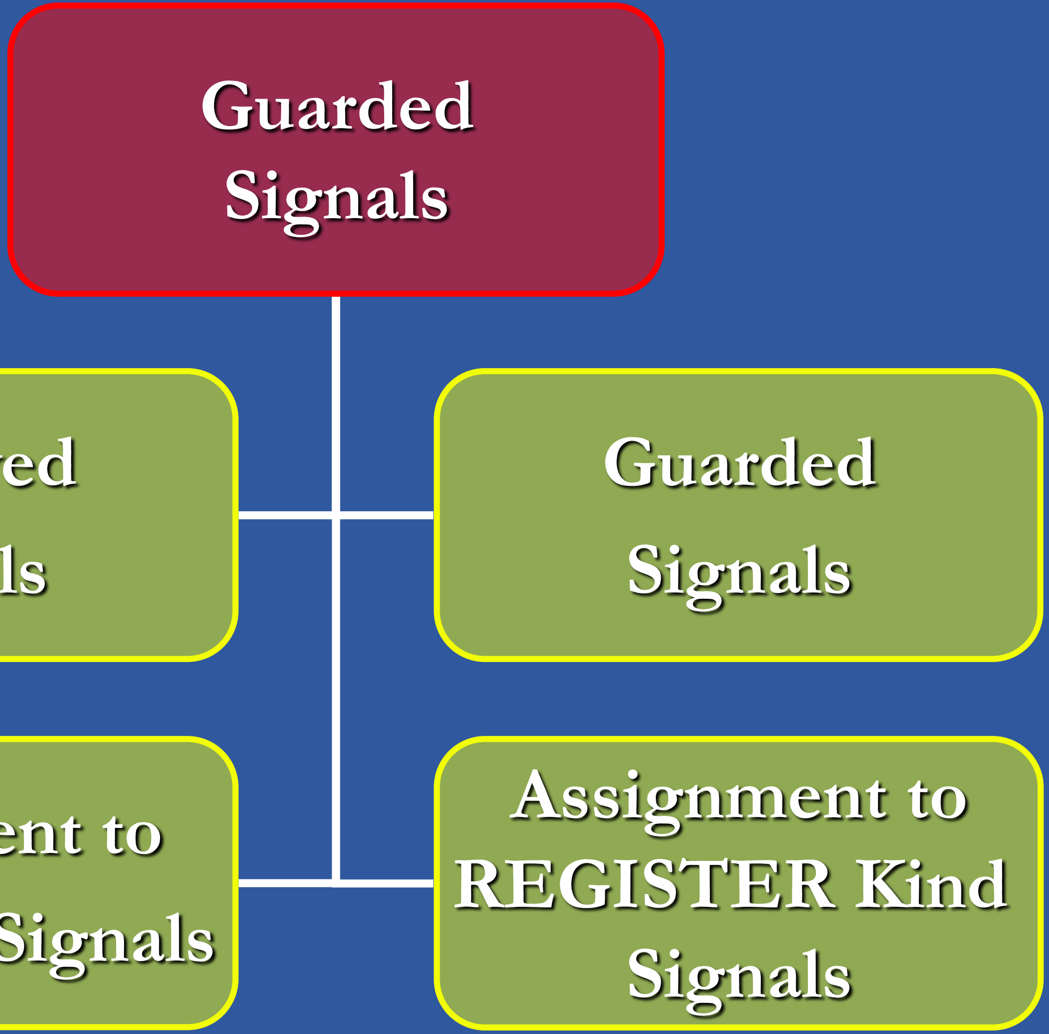
```
ENTITY latchflop IS
   PORT (din, clk : IN BIT; ql, qf : OUT BIT);
END ENTITY;
--
ARCHITECTURE nested OF latchflop IS
BEGIN
   lat: BLOCK (clk = '1') BEGIN
      ql <= GUARDED din;
      reg: BLOCK (GUARD AND NOT clk'STABLE) BEGIN
         qf <= GUARDED din;
      END BLOCK reg;
   END BLOCK lat;
END ARCHITECTURE nested;
```

- Nested Block Statements

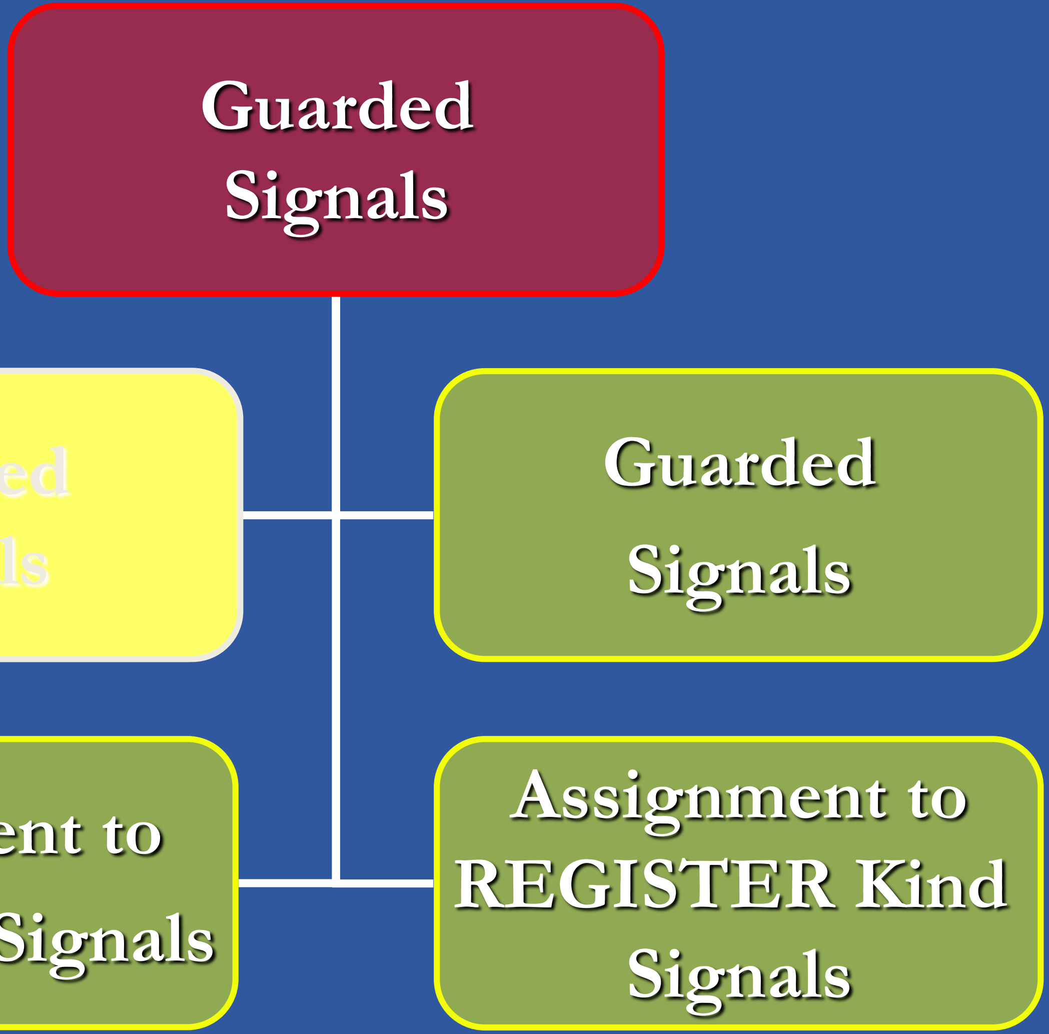
Guarded Signals



Guarded Signals


Guarded Signals
Resolved Signals
Guarded Signals
Assignment to BUS Kind Signals
Assignment to REGISTER Kind Signals

Resolved Signals


Guarded Signals
Resolved Signals
Guarded Signals
Assignment to BUS Kind Signals
Assignment to REGISTER Kind Signals

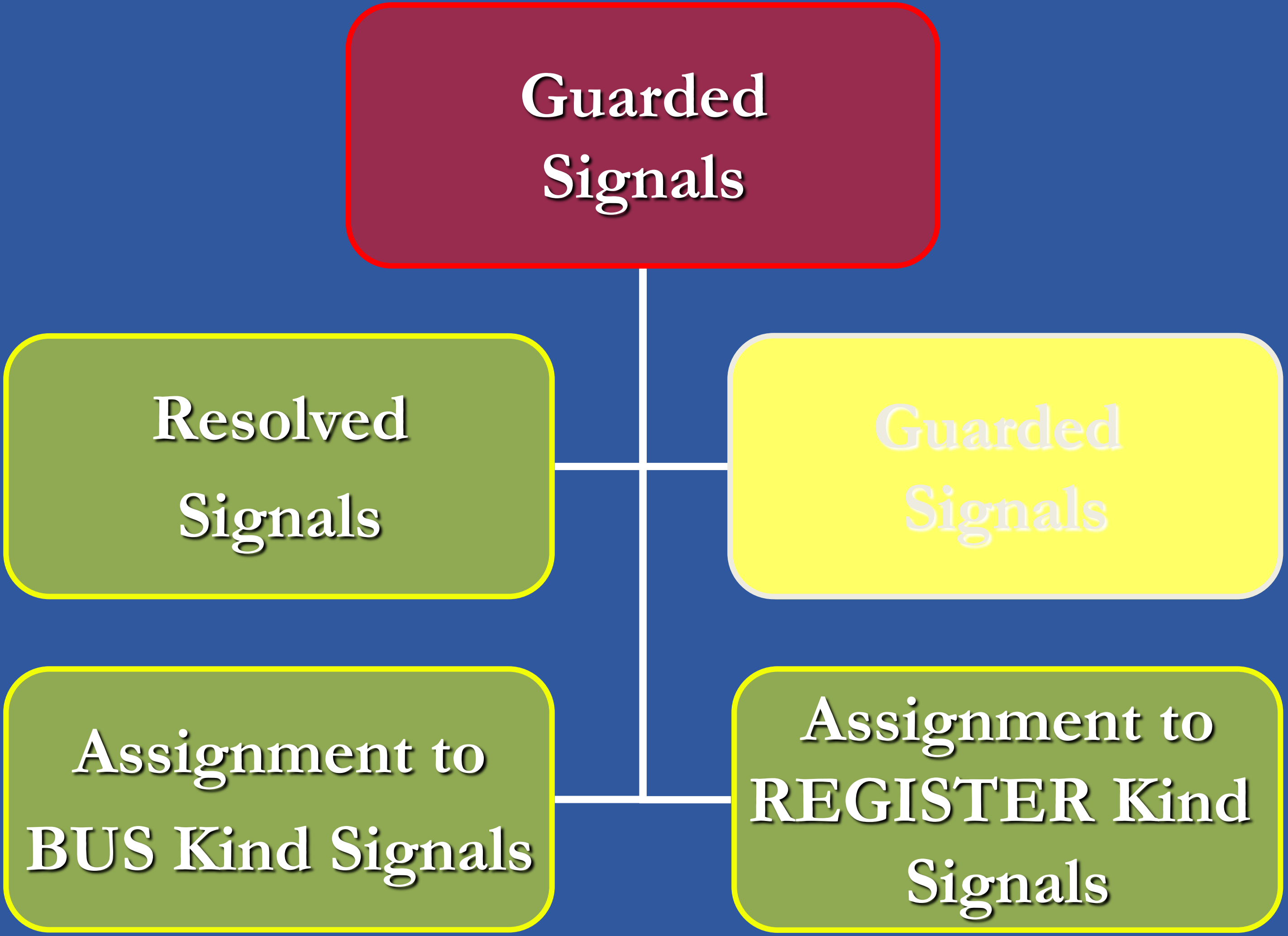
Resolved Signals

- **Resolved signals can have multiple concurrent drivers**
- **Associated with a resolved signal is a resolution function that resolves between multiple values assigned to a signal.**

- **VHDL predefined *std_logic* resolved type**
- **Limiting the use of resolved signals to having only one concurrent driver.**

- **With a resolution function, there is a default resolved value:**
  - **Becomes the value of the resolved signal if the signal has no driver.**
  - **Defined by the resolution function and may be different from the default value of signal**
  - **The default resolved value for *std_logic* type is ‘Z’.**

Guarded Signals


Guarded Signals
Resolved Signals
Guarded Signals
Assignment to BUS Kind Signals
Assignment to REGISTER Kind Signals

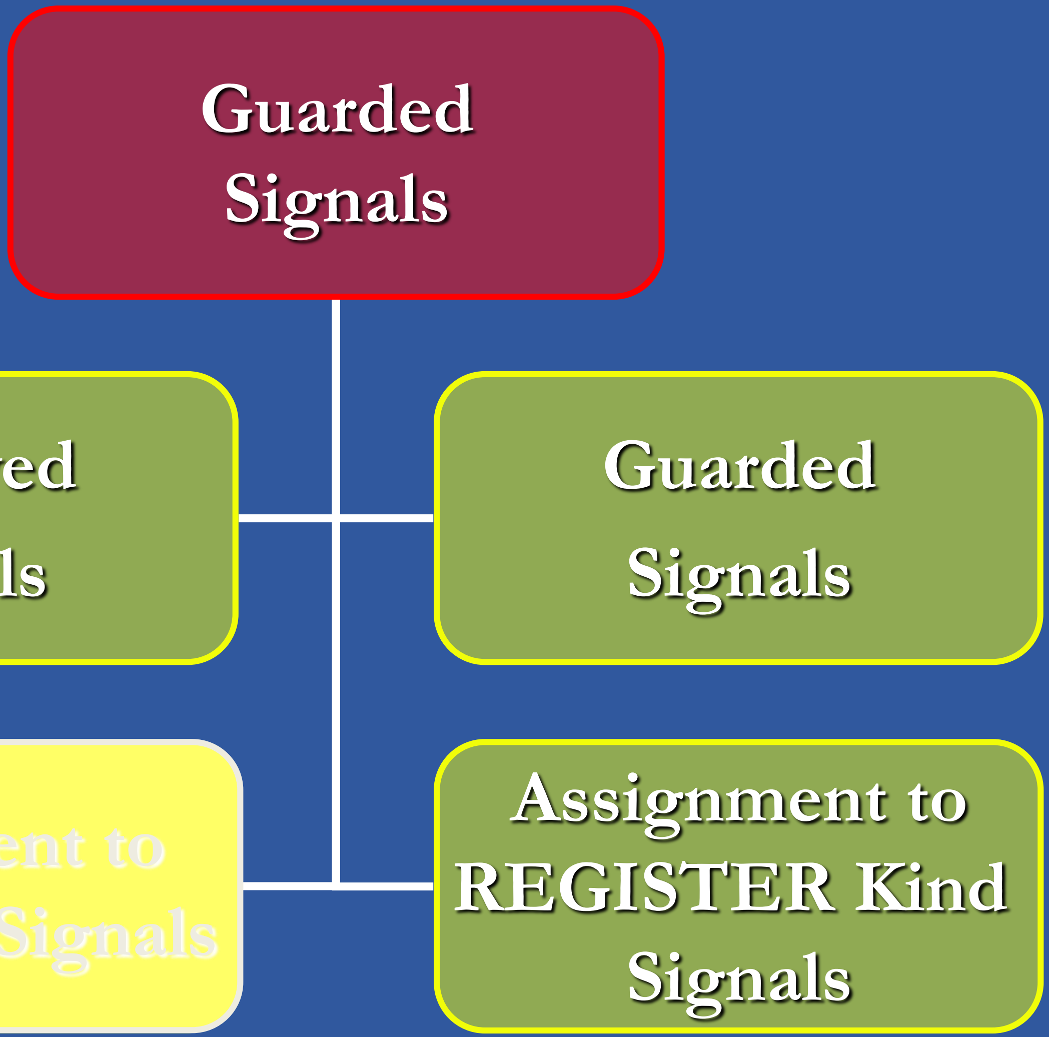
Resolved Signals

- A resolved signal can be declared to have a kind.
- Kind specification follows the type mark of the resolved signal in its declaration.

- Kind of a signal can be
  - BUS
  - REGISTER

- If undriven or if a disconnection occurs:
  - A guarded signal of BUS kind receives its default resolved value
  - An undriven REGISTER kind guarded signal retains its old value (the value before disconnection).

Assignment to BUS Kind Signals

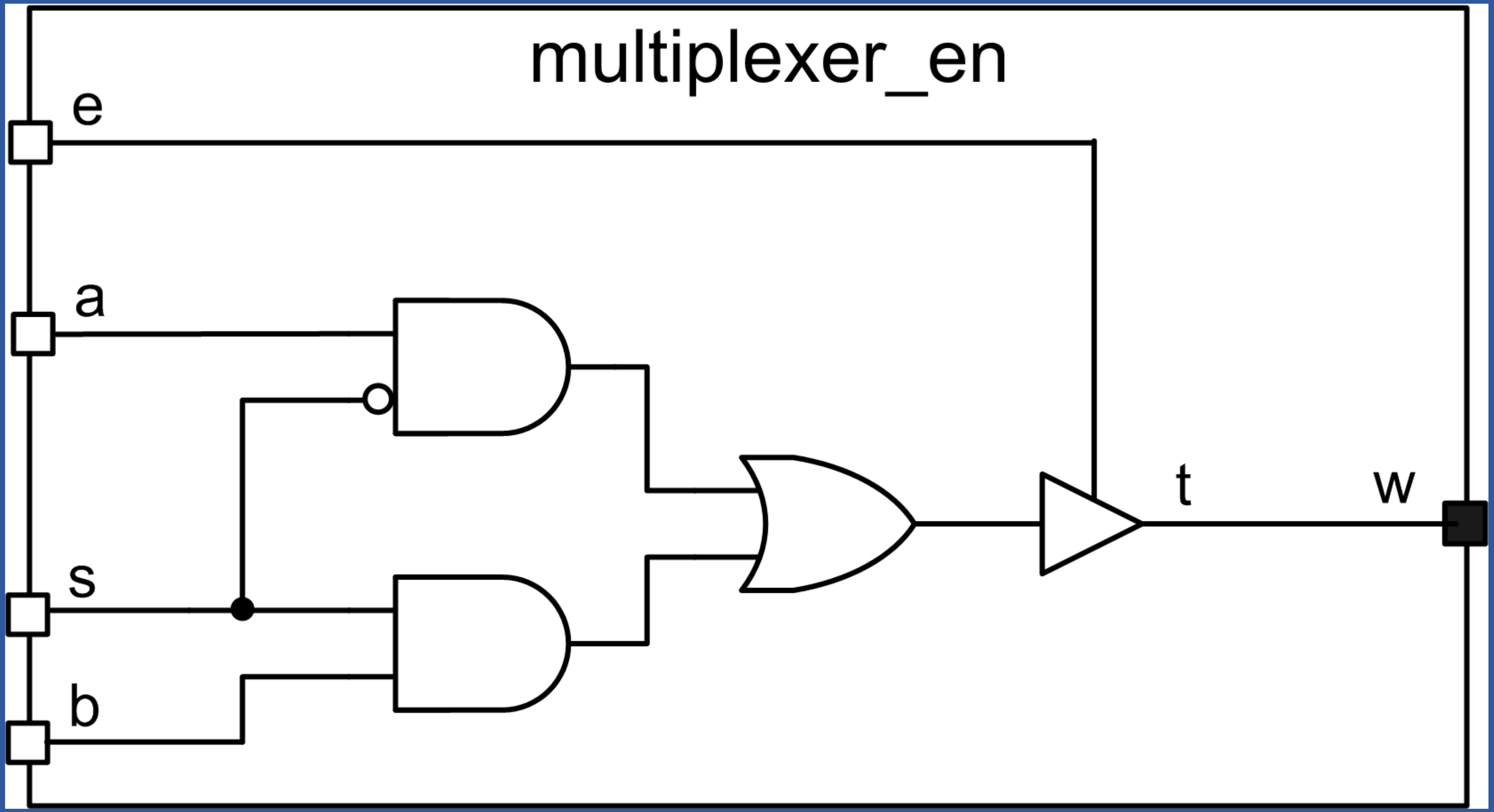

Guarded Signals
Resolved Signals
Guarded Signals
Assignment to BUS Kind Signals
Assignment to REGISTER Kind Signals

Assignment to BUS Kind Signals

```
ENTITY multiplexer_en IS
   PORT (a, b, s, e : IN std_logic; w : OUT std_logic);
END ENTITY;
--
ARCHITECTURE blocking OF multiplexer_en IS
   SIGNAL t : std_logic BUS;
BEGIN
   tri: BLOCK (e='1') BEGIN
      t <= GUARDED (a AND NOT s) OR (b AND s);
      w <= t;
   END BLOCK tri;
END ARCHITECTURE blocking;
```

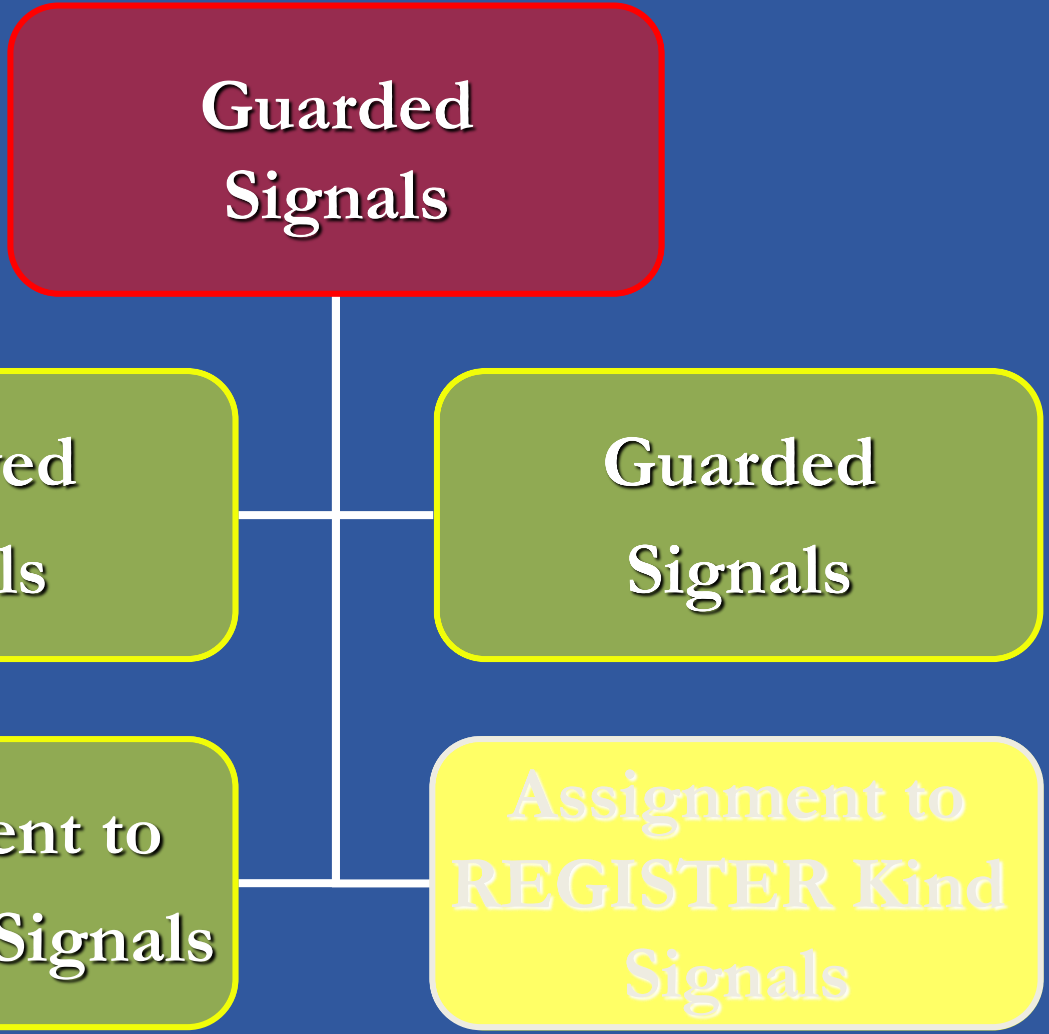
- BUS Kind Guarded Signal

Assignment to BUS Kind Signals



- **Hardware Corresponding to *multiplexer_en***

Assignment to REGISTER Kind Signals

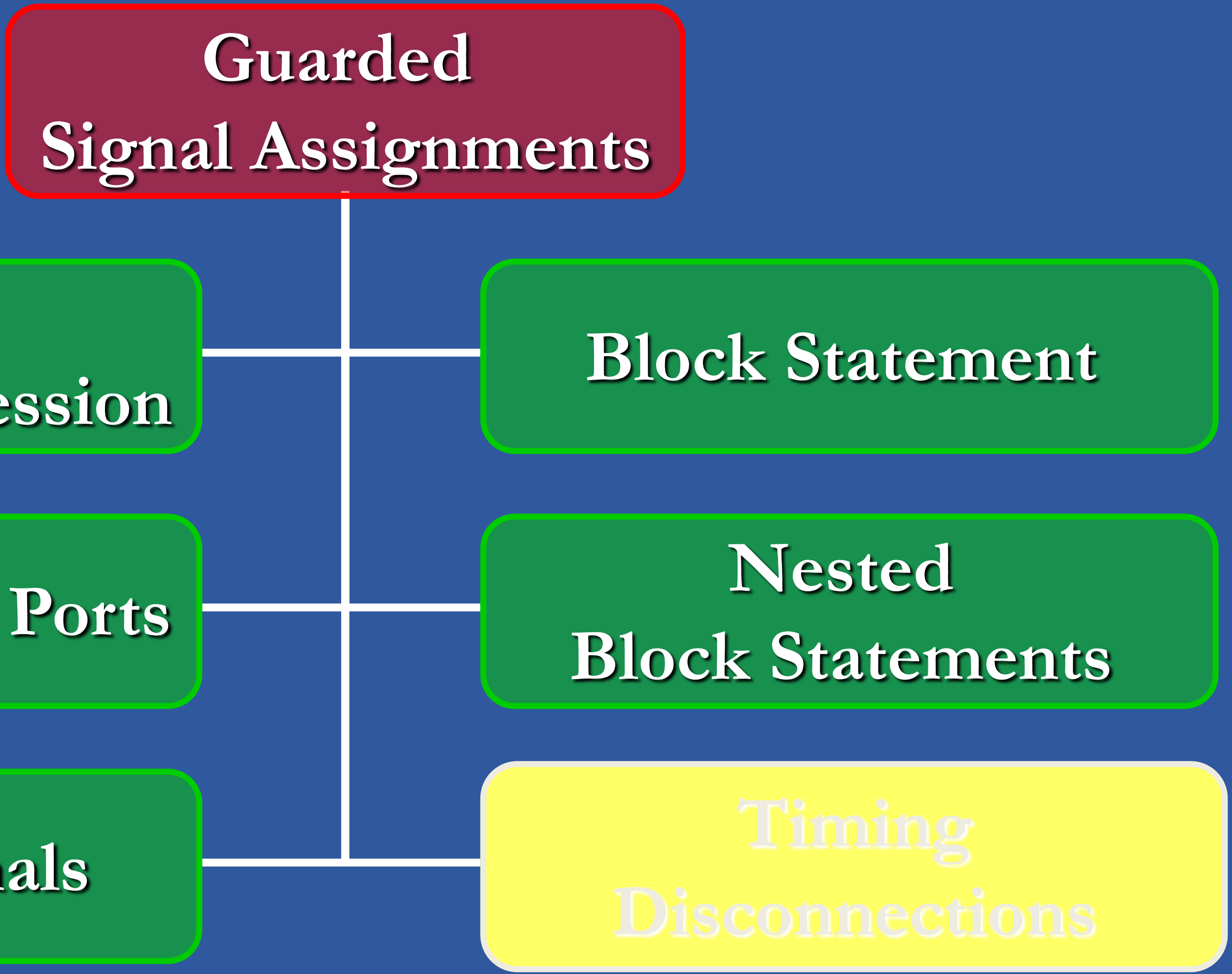

Guarded Signals
Resolved Signals
Guarded Signals
Assignment to BUS Kind Signals
Assignment to REGISTER Kind Signals

Assignment to REGISTER Kind Signals

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
ENTITY stdflop IS
   PORT (reset, din, clk, cen : IN std_logic:='0';
   qout : OUT std_logic);
END ENTITY;
--
ARCHITECTURE blockport OF stdflop IS BEGIN
   reg: BLOCK (clk = '1' AND NOT clk'STABLE)
      PORT (r, d, c : IN std_logic; q : OUT std_logic);
      PORT MAP (reset, din, clk, qout);
      SIGNAL ff : std_logic REGISTER;
   BEGIN
      ff <= GUARDED '0' WHEN r = '1' ELSE d;
      q <= ff;
   END BLOCK reg;
END ARCHITECTURE blockport;
```

- REGISTER Kind Guarded Signal

Timing Disconnections



Timing Disconnections

```
ARCHITECTURE timedisconnect OF multiplexer_en IS
   SIGNAL t : std_logic BUS;
   DISCONNECT t : std_logic AFTER 3.25 NS;
BEGIN
   tri: BLOCK (e='1') BEGIN
      t <= GUARDED (a AND NOT s) OR (b AND s);
      w <= t;
   END BLOCK tri;
END ARCHITECTURE timedisconnect;
```

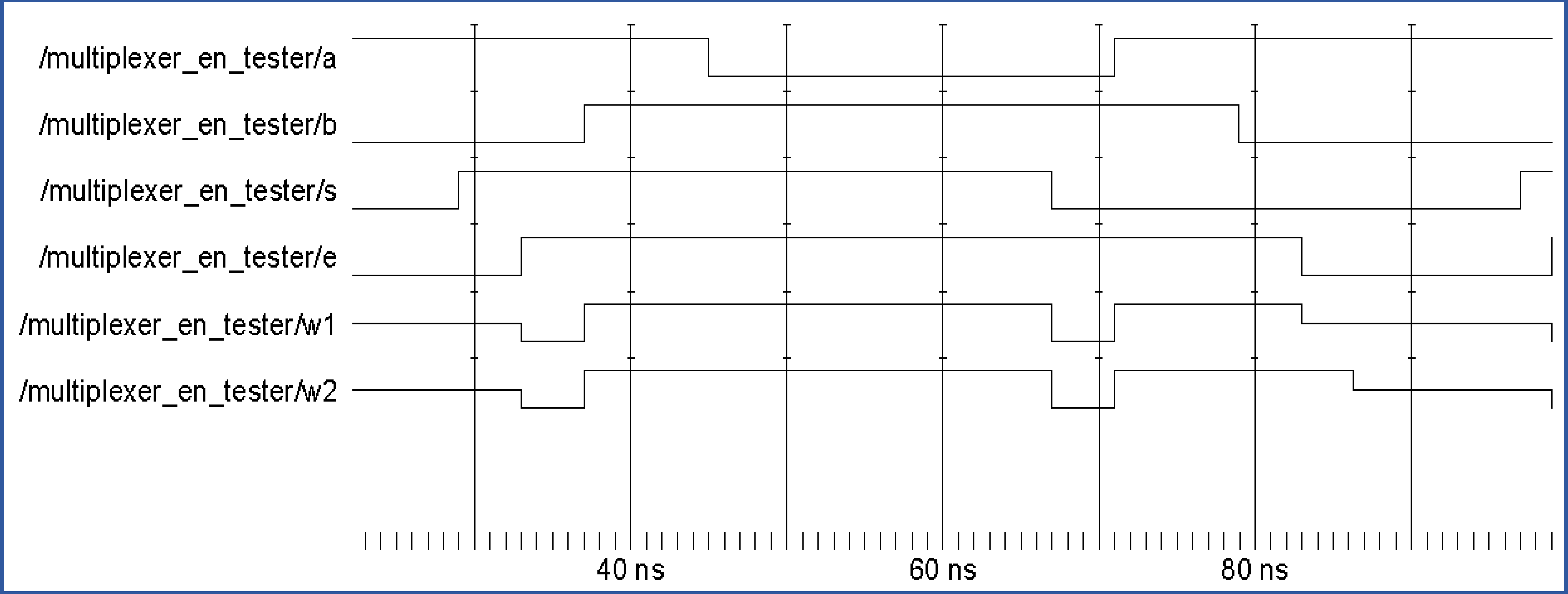
- Multiplexer with Disconnect Time

Timing Disconnections

```
ARCHITECTURE timed OF multiplexer_en_tester IS
   SIGNAL a, b, s, e, w1, w2 : std_logic;
BEGIN
   UUT1: ENTITY WORK.multiplexer_en (blocking)
         PORT MAP (a, b, s, e, w1);
   UUT2: ENTITY WORK.multiplexer_en (timedisconnect)
         PORT MAP (a, b, s, e, w2);
   a <= '0', '1' AFTER 20 NS, '0' AFTER 45 NS,
        '1' AFTER 71 NS;
   b <= '1', '0' AFTER 11 NS, '1' AFTER 37 NS,
        '0' AFTER 79 NS;
   s <= '0', '1' AFTER 29 NS, '0' AFTER 67 NS,
        '1' AFTER 97 NS;
   e <= '0', '1' AFTER 33 NS, '0' AFTER 83 NS,
        '1' AFTER 99 NS;
END ARCHITECTURE timed;
```

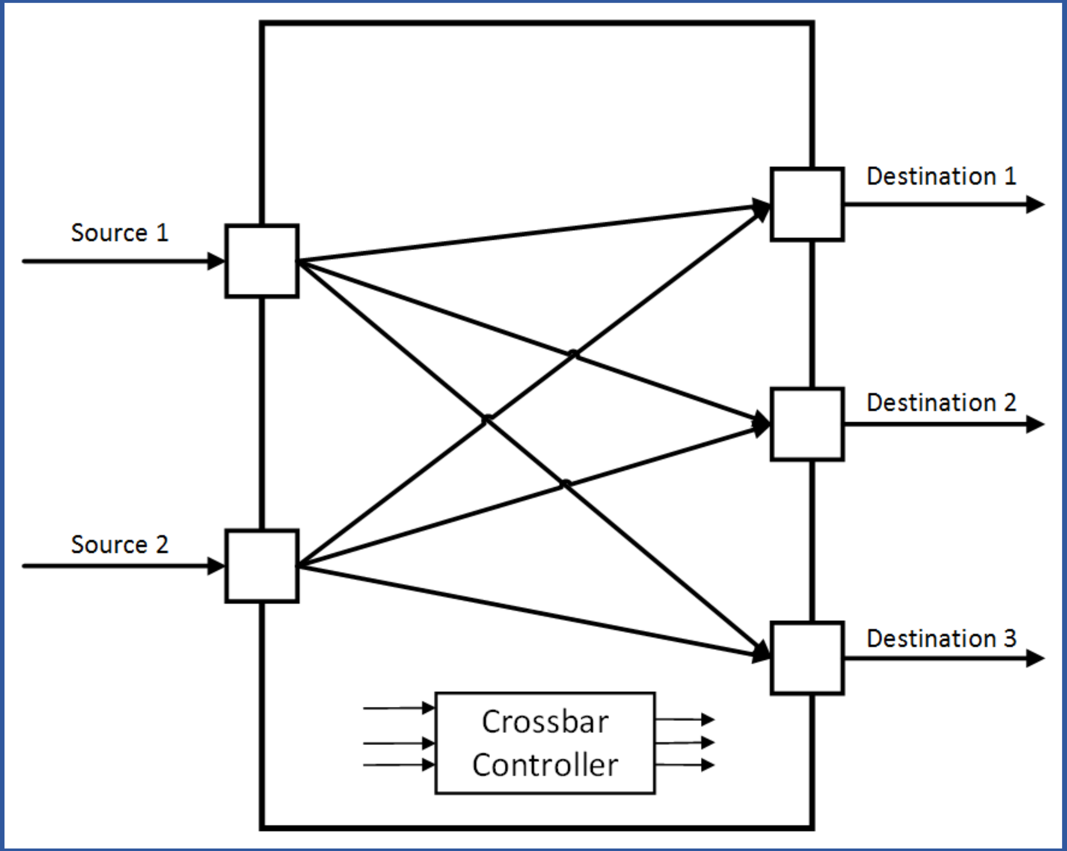
- Comparing Disconnection Timing

Timing Disconnections



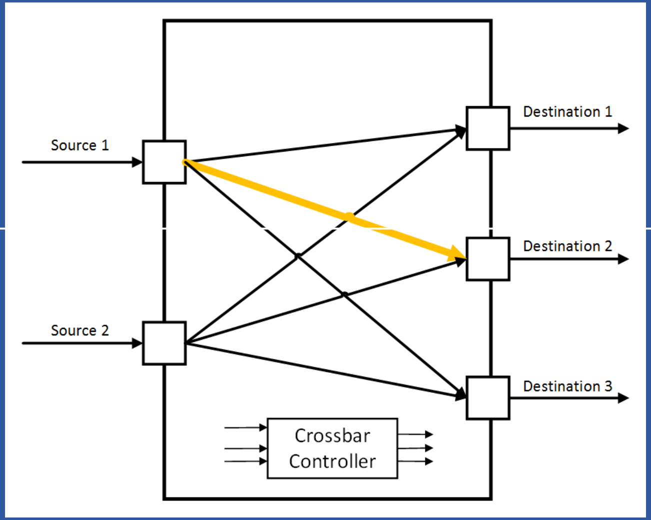
- **Comparing Disconnection Times**

Timing Disconnections



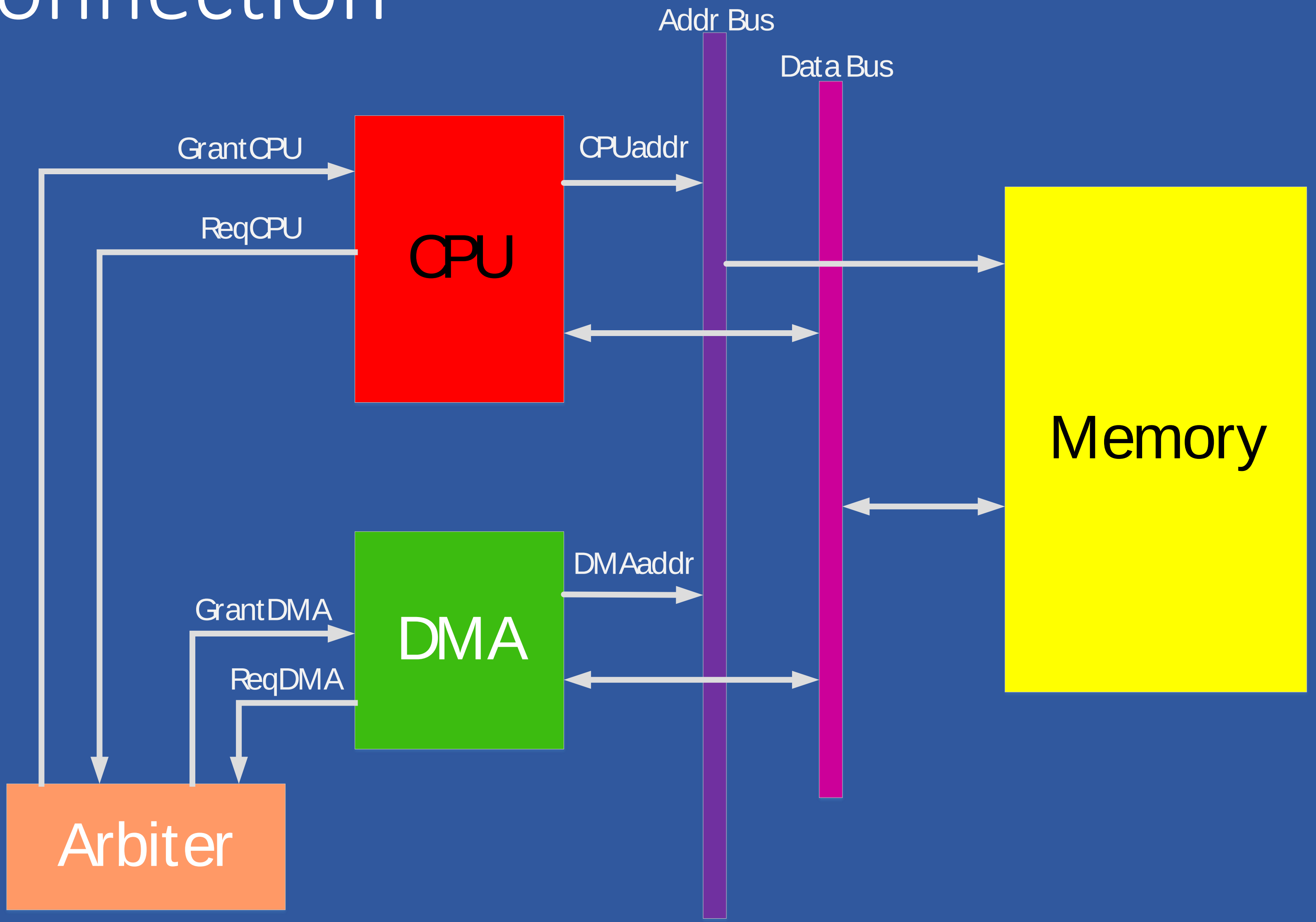
Timing Disconnections

```
ARCHITECTURE timedisconnect OF crossbar IS
   SIGNAL s1d2: std_logic_vector(7 DOWNTO 0) BUS;
   DISCONNECT s1d2: std_logic_vector(7 DOWNTO 0) AFTER 3.25 NS;
. . .
BEGIN
   tri: BLOCK (cont_en='1') BEGIN
      s1d2 <= GUARDED (s1turn AND s1tod2);
      d2 <= s1d2;
. . .
   END BLOCK tri;
END ARCHITECTURE timedisconnect;
```



- A crossbar connection with Disconnect Time

Bus Connection



Bus Connection

```
ARCHITECTURE Blocked OF DMA IS
BEGIN
   BlOCK (GrantDMA = '1') BEGIN
      addrBus <= GUARDED (DMAaddr);
   END BLOCK;

   PROCESS ( ... ) BEGIN
      ...
      WAIT FOR GrantDMA;
      ...
   END PROCESS;

END ARCHITECTURE Blocked;
```

Bus Connection

```
ARCHITECTURE Blocked OF CPU IS
BEGIN
   addr: BlOCK (GrantCPU = '1')
   BEGIN
      addrBus <= GUARDED (CPUaddr);
   END BLOCK addr;

   ...

   data: BlOCK (GrantCPU = '1‘ AND W_Mem = '1‘)
   BEGIN
      dataBus <= GUARDED (CPUdata);
   END BLOCK data;
   ...
END ARCHITECTURE Blocked;
```

Summary

- The focus of this chapter was on
  - Concurrent bodies of VHDL
  - Signal assignments
  - Block statements
  - Guarded signals and guarded assignments
- The constructs discussed here enable description of hardware at a level higher that what was discussed in the previous chapter.
- For more behavioral descriptions VHDL offers sequential statements within concurrent bodies that will be discussed in the next chapter.

Acknowledgment

Slides developed by:

Homa Alemzadeh

First revision 2017 by:

Bahar Behazin

Second revision 2019 by:

Saba Yousefzadeh