TM Associates Inc.
Training Specialists

VHDL

Sample Slides from the
2-day and 4-day VHDL
Training Courses

Rev. 4.7

VHDL © 2011 TM Associates, Inc.

1-1

These sample slides are taken from the 4-day basic VHDL training course. They are from a variety of topics covered in the training. The first two days of the 4-day course cover the same material as the 2-day course. Either the 2-day or 4-day course is a prerequisite for the 3-day *Advanced VHDL Coding Styles for Synthesis & Verification* course.

The training material was developed by a team of VHDL designers with a combined 50+ years of experience. The lead developer was Peter Ashenden, the author of

The Designer's Guide to VHDL, 3rd Edition

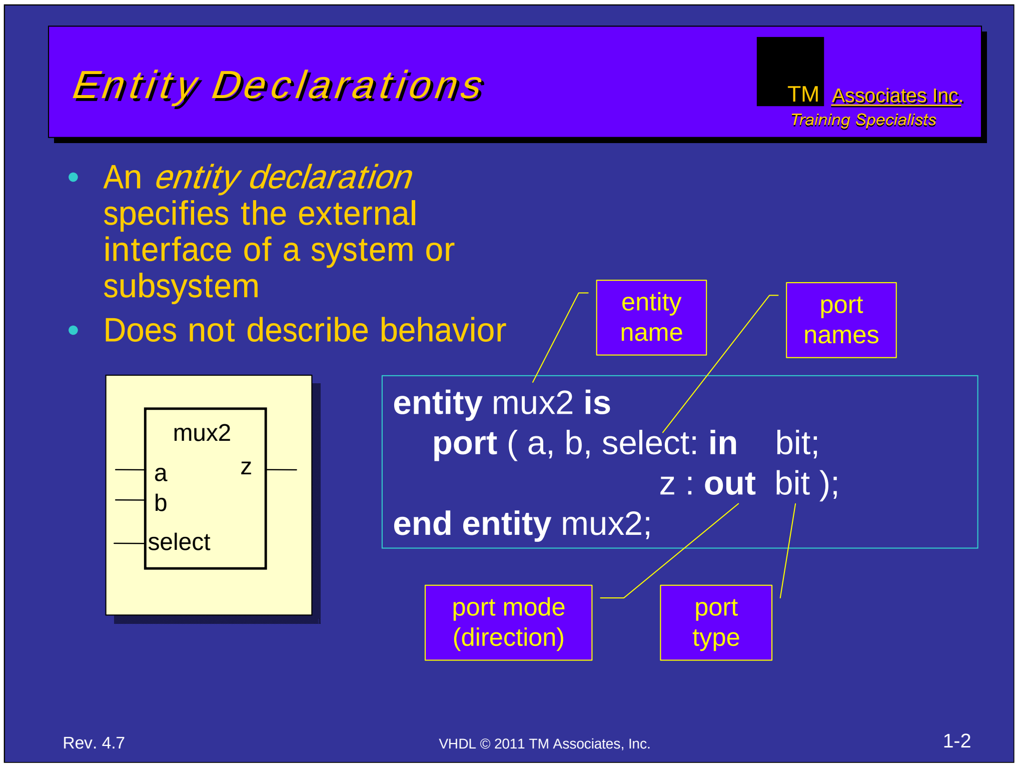
which is the most popular book on the VHDL language.

Comprehensive notes are included with each slide. This means the student will

- Be able to spend more time listening to the instructor instead of taking notes, and
- Be able to use the training manual as a valuable reference guide after the training.

For complete information on all of our courses visit our website

www.tm-associates.com



Entity Declarations

TM Associates Inc.
Training Specialists

- An *entity declaration* specifies the external interface of a system or subsystem
- Does not describe behavior

mux2
a
b
select
z

entity name

port names

```
entity mux2 is
   port ( a, b, select: in   bit;
                     z : out  bit );
end entity mux2;
```

port mode (direction)

port type

Rev. 4.7 VHDL © 2011 TM Associates, Inc. 1-2

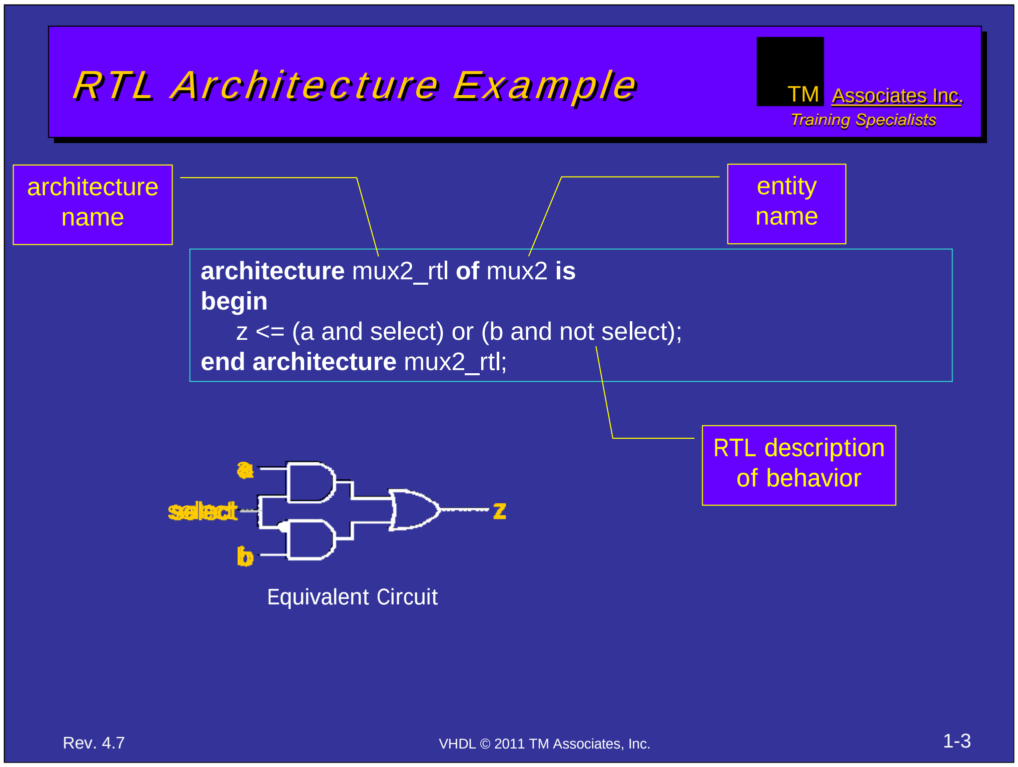
We introduce the basic concepts using a 2:1 multiplexer design as an example. Here is a schematic symbol representing the mux pins and a corresponding VHDL description of that symbol.

An entity declaration specifies:

- The name of a system or subsystem. We can repeat the name at the end of the entity declaration.
- Names for the ports. Ports are the connection points for the entity.
- Modes for the ports, specifying the direction of information flow.
- Types for the ports. In this example, the types are bit, so values on the ports are ‘0’ or ‘1’. We will see other types later.

We can list a number of ports of the same mode and type together, as we have done in this example. Or we can write them out separately.

Notice in the port declaration for the output ‘z’, there is a lot of white space in the line. White space is ignored in VHDL, allowing the author to put as much or little white space as needed to make the code most readable.



RTL Architecture Example

TM Associates Inc.
Training Specialists

architecture name

entity name

```
architecture mux2_rtl of mux2 is
begin
   z <= (a and select) or (b and not select);
end architecture mux2_rtl;
```

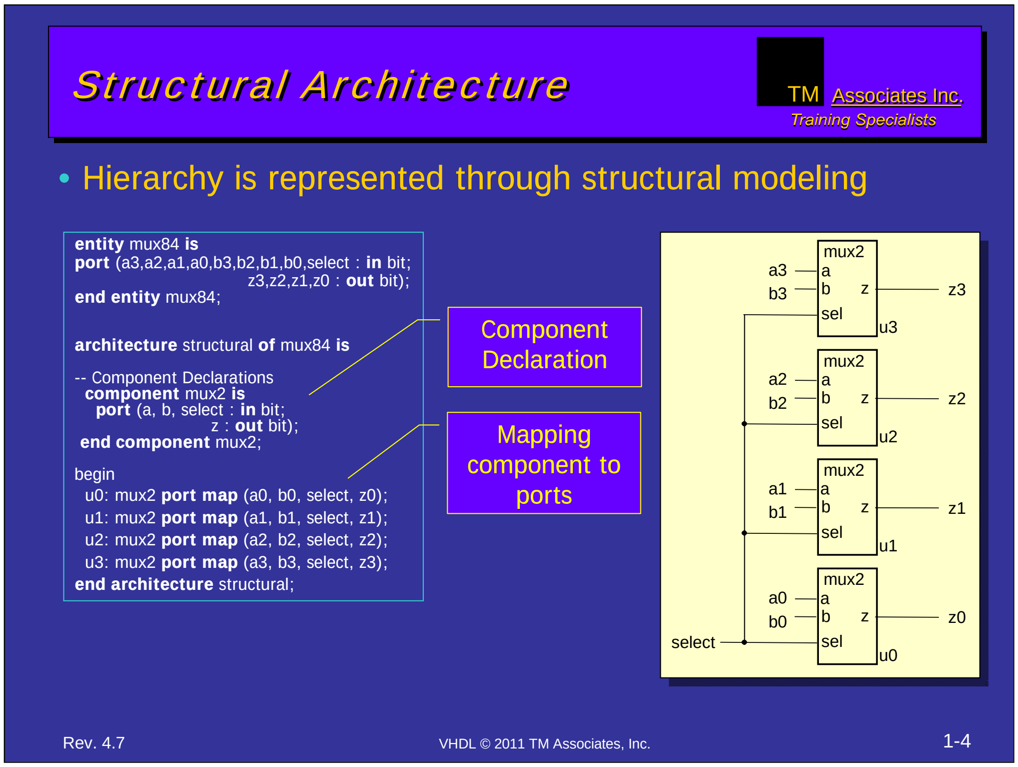
RTL description of behavior

Equivalent Circuit

Rev. 4.7 VHDL © 2011 TM Associates, Inc. 1-3

Here is an example of a simple RTL architecture body for the 2:1 mux example. This architecture body contains one process statement that describes the algorithm performed by the mux. The statement contains boolean operators (and, or, not) to describe a synthesizable description of a mux.

A single entity can have multiple architectures, but an architecture must be associated with a specific entity, as is described by the first line of the architecture statement.



Structural Architecture

TM Associates Inc.
Training Specialists

- Hierarchy is represented through structural modeling

```
entity mux84 is
port (a3,a2,a1,a0,b3,b2,b1,b0,select : in bit;
                         z3,z2,z1,z0 : out bit);
end entity mux84;

architecture structural of mux84 is

-- Component Declarations
  component mux2 is
    port (a, b, select : in bit;
                    z : out bit);
 end component mux2;

begin
  u0: mux2 port map (a0, b0, select, z0);
  u1: mux2 port map (a1, b1, select, z1);
  u2: mux2 port map (a2, b2, select, z2);
  u3: mux2 port map (a3, b3, select, z3);
end architecture structural;
```

Component Declaration

Mapping component to ports

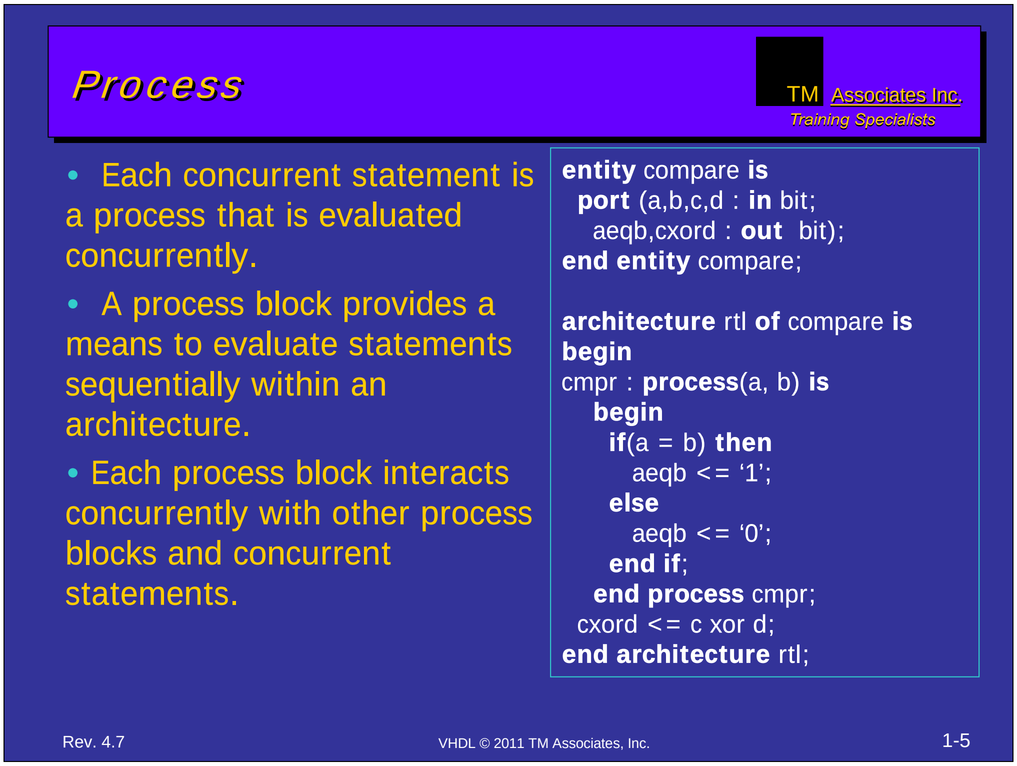
Rev. 4.7 VHDL © 2011 TM Associates, Inc. 1-4

Here we see an entity/architecture pair that creates a 2:1 mux of two 4-bit busses using structural VHDL code to wire four of the previously designed 2:1 mux components. Components must be declared in the declaration section of the architecture before being instantiated in the body of the architecture.

The components are wired as shown, using a mapping scheme to map the component ports to the intended targets. These statements start with a unique name for each individual instantiation, followed by the name of the previously declared component that provides a socket to wire to. Finally, the port map defines how the socket is wired.

Looking at the port map for the instantiation of ‘u0’, we see:
port ‘a0’ is wired to the u0 socket pin ‘a’,
port ‘b0’ is wired to u0 socket pin ‘b’,
port ‘select’ is wired to u0 socket pin ‘select’,
and
output port ‘z0’ is wired to u0 socket pin ‘z’.

This type of mapping is known as “positional association”, in that the signals in the list will be wired to the ports in the order defined in the component entity.
This style of mapping, while legal, can cause miswiring errors. A better way will be seen in later slides.



Process

TM Associates Inc.
Training Specialists

- Each concurrent statement is a process that is evaluated concurrently.
- A process block provides a means to evaluate statements sequentially within an architecture.
- Each process block interacts concurrently with other process blocks and concurrent statements.

```
entity compare is
  port (a,b,c,d : in bit;
    aeqb,cxord : out  bit);
end entity compare;

architecture rtl of compare is
begin
cmpr : process(a, b) is
    begin
      if(a = b) then
         aeqb <= ‘1’;
      else
         aeqb <= ‘0’;
      end if;
    end process cmpr;
  cxord <= c xor d;
end architecture rtl;
```

Rev. 4.7 VHDL © 2011 TM Associates, Inc. 1-5

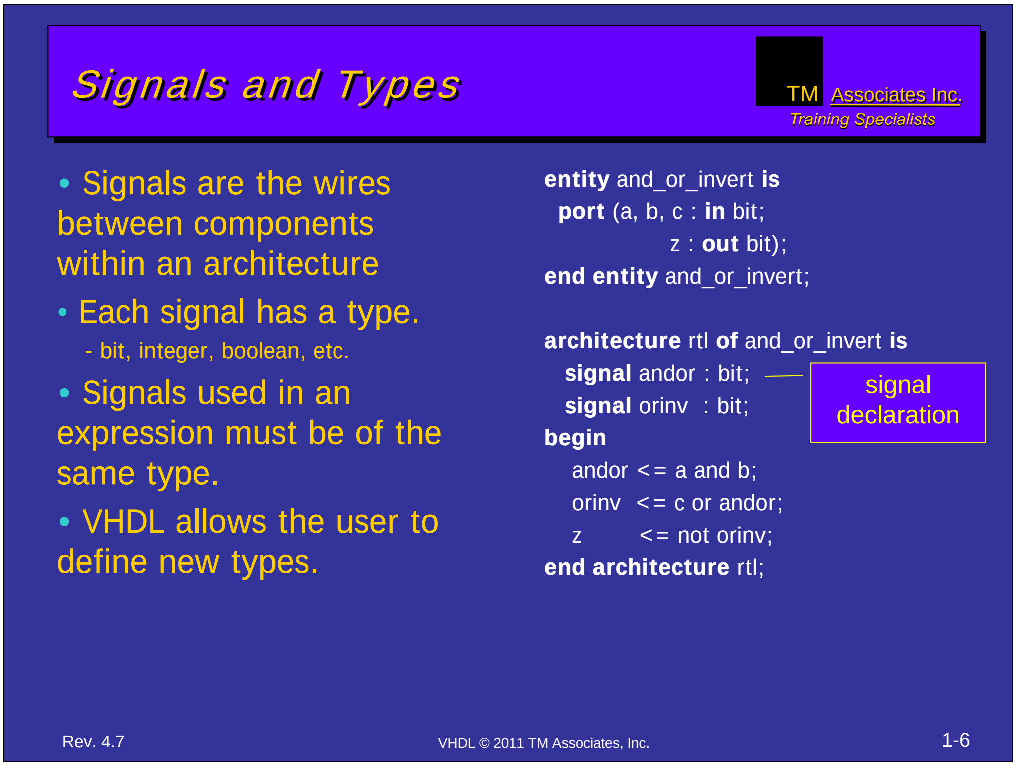
A process block allows the statements within to execute in a sequential fashion.

An architecture may contain a combination of process blocks, concurrent statements, and instantiated components.

A process will contain either a sensitivity list (as shown in the example) or a wait statement. A process is a continuous loop that never ends. The sensitivity list (or wait statement) defines the signals that start a process evaluation, after which the process becomes idle until one of the sensitive signals has another event.

Events resulting from the evaluation of statements within a process block are not scheduled until the entire block has been evaluated. The entire process block, therefore, is treated as a single concurrent statement, and functions concurrently with other process blocks and concurrent statements.

In the example given, the output aeqb is evaluated within a process block, while the output cxord is evaluated in a concurrent statement. Both the process block and the concurrent statement are evaluated at the same time, with results scheduled at the end of the current simulation cycle (not synonymous with a clock cycle but rather a zero time cycle called a delta)



Signals and Types

TM Associates Inc.
Training Specialists

- Signals are the wires between components within an architecture
- Each signal has a type.
  - bit, integer, boolean, etc.
- Signals used in an expression must be of the same type.
- VHDL allows the user to define new types.

```
entity and_or_invert is
  port (a, b, c : in bit;
                z : out bit);
end entity and_or_invert;

architecture rtl of and_or_invert is
   signal andor : bit;
   signal orinv  : bit;
begin
    andor <= a and b;
    orinv  <= c or andor;
    z        <= not orinv;
end architecture rtl;
```

signal declaration

Rev. 4.7 VHDL © 2011 TM Associates, Inc. 1-6

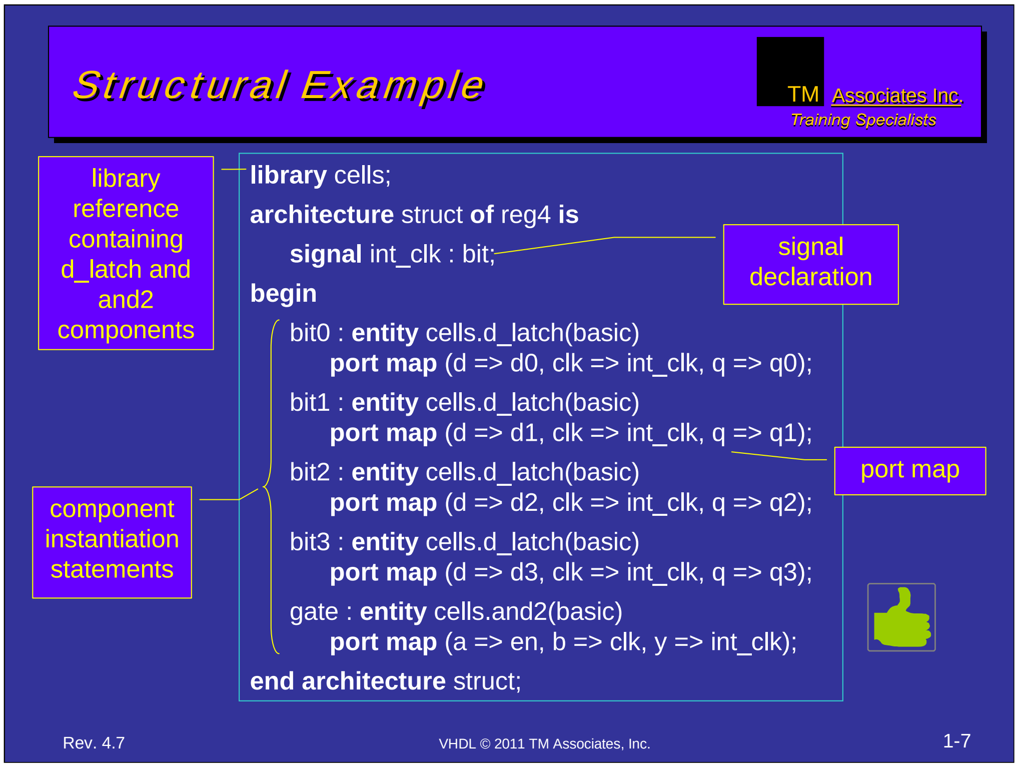
An object in VHDL is defined as an item that holds a value. The three classes of objects are Signals, Variables, and Constants.

Objects must have a type that describes the kind of value the object can hold. VHDL provides some predefined types, including bit, integer, boolean, and bit_vector. Our examples so far have all used type ‘bit’ for ports, which are a type of signal.

VHDL allows new types to be developed by the user to make code more readable and improve error checking.

VHDL is known as a strongly typed language, and requires all expressions to use objects with the same type attribute.   This provides one means for built in error checking.

An object is synthesizable if it is statically computable at compilation time. For example, a type will have a range of values, and the value assigned to an object must be contained within that range.



Structural Example

TM Associates Inc.
Training Specialists

library reference containing d_latch and and2 components

```
library cells;
architecture struct of reg4 is
    signal int_clk : bit;
begin
    bit0 : entity cells.d_latch(basic)
        port map (d => d0, clk => int_clk, q => q0);
    bit1 : entity cells.d_latch(basic)
        port map (d => d1, clk => int_clk, q => q1);
    bit2 : entity cells.d_latch(basic)
        port map (d => d2, clk => int_clk, q => q2);
    bit3 : entity cells.d_latch(basic)
        port map (d => d3, clk => int_clk, q => q3);
    gate : entity cells.and2(basic)
        port map (a => en, b => clk, y => int_clk);
end architecture struct;
```

signal declaration

port map

component instantiation statements

Rev. 4.7 VHDL © 2011 TM Associates, Inc. 1-7

This structural architecture body for the register describes the same connectivity as the schematic. The library clause at the top specifies that the architecture makes use of units that reside in the library named cells, which includes the latch and and gate used in the architecture.

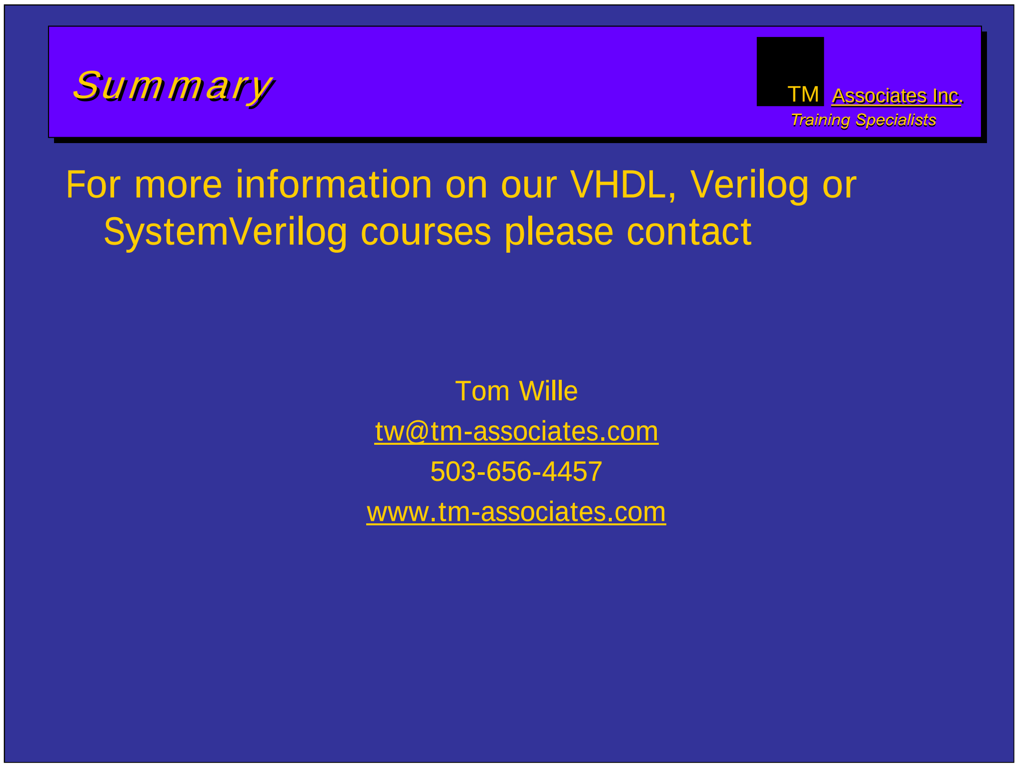
The architecture contains

- a signal declaration for the internal connection, int_clk
- four instances of the d_latch entity, with distinct labels bit0, bit1, bit2 and bit3
- an instance of the and2 entity labeled gate

The instances of the entities all are prefixed with “cells.”, indicating that the entities are to be found in the cells library. The part in parentheses (basic) selects the particular architecture to be used with that entity. This is optional, but is a good design practice, as it makes the code readable, and ensures the correct architecture is selected for synthesis.

In the absence of any guidance (by configuration or explicitly identifying an architecture in an instantiation) , the last analyzed architecture will be used as a default.

In the earlier example of a structural architecture, we mentioned the danger of implicit port mapping. The best design practice is to use explicit port mapping at all times, as shown in the example above. The name to the left of each “=>” symbol is a component port name, and the name to the right is an external port or architecture signal name. This avoids confusion, and makes the design more readable, as you can see exactly how the ports are connected without having to refer to the entity or component declaration.



Summary

TM Associates Inc.
Training Specialists

For more information on our VHDL, Verilog or SystemVerilog courses please contact

Tom Wille
tw@tm-associates.com
503-656-4457
www.tm-associates.com