The preceding lectures introduced a basic approach for synthesis of logic circuits.

A designer could use this approach manually for small circuits.

However, logic circuits found in complex systems, such as today’s computers, cannot be designed manually; they are designed using sophisticated CAD tools that automatically implement the synthesis techniques.

A number of CAD tools are packaged together into CAD system, which typically includes tools for the following tasks: design entry, synthesis and optimization, simulation, and physical design.
A typical CAD system

1. Design conception
2. Design Entry
3. Synthesis
   - Function simulation
4. Design correct?
   - Yes
   - No
5. Physical design
   - Timing simulation
6. Timing requirements met?
   - Yes
   - No
7. Chip configuration
Background on HDLs

- Hardware description languages (HDLs) are languages used to describe and model the operation of digital circuits.
  - You can use an HDL to describe a circuit.
  - You can also use an HDL to describe how to stimulate the circuit and check its response.
- Simulation of the above requires a logic simulator.
Background on HDLs

- An HDL circuit description may be used as an input to a synthesis tool. Such a tool transforms the HDL description into a representation that may be physically implemented (transistors, or logic gates...).
- When a human does this, it is called logic design.
- When a machine does this, it is called synthesis.
- Synthesis is algorithmic “design”.

Dr. Mohamed Abd el Ghany
Department of Electronics and Electrical Engineering

ELCT 501: Digital System Design
There are a fair number of HDLs, but two are by far most prevalent in use:

- **Verilog-HDL**, the Verilog Hardware Description Language, not to be confused with Verilog-XL, a logic simulator program sold by Cadence

- **VHDL**, or VHSIC Hardware Description Language and VHSIC is Very High Speed Integrated Circuit.

In this class, we will be using only VHDL.
Background on HDLs

Verilog history:

1983 Gateway Design Automation released the Verilog HDL and a simulator for it.

1989 Cadence acquired Gateway.

1990 Cadence separated the Verilog HDL from their simulator product, Verilog-XL, releasing the HDL into the public domain, guarded by OVI.

1995 IEEE adopted Verilog as standard 1364

Background on HDLs

VHDL history:

- 1983 VHDL was developed under the VHSIC program of Department of Defense (DOD).
- 1987 The IEEE adopted the VHDL language as standard 1076. The DOD mandated that all digital electronic circuits be described in VHDL.
- 1993 The VHDL language was slightly revised into what is often referred to as VHDL93 (versus the previous VHDL87).
Which is “better” Verilog or VHDL?

- Both are adequate for our purpose...
- What you use in industry may be dictated by company preference or government requirement.
- VHDL may be more powerful but very rigid.

- USA - IBM, TI, AT&T, INTEL – VHDL
- USA - Silicon Valley – Verilog
- Europe – VHDL
- Japan – Verilog
- Korea – 70-80% VHDL
Applications of HDL

- Model and document digital systems
  - Different levels of abstraction
    - Behavioral, structural, etc.
- Verify design
- Synthesize circuits
  - Convert from higher abstraction levels to lower abstraction levels
Problem: write VHDL code for detecting even parity of a 3-bit input signals.

\[ \text{even} = a(2)' \cdot a(1)' \cdot a(0)' + a(2)' \cdot a(1) \cdot a(0) + a(2) \cdot a(1)' \cdot a(0) + a(2) \cdot a(1) \cdot a(0)' \]

<table>
<thead>
<tr>
<th>a(2)</th>
<th>a(1)</th>
<th>a(0)</th>
<th>even</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

How VHDL describe this circuit?
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity even_detector is
    Port ( a : in STD_LOGIC_VECTOR (2 downto 0);
           even : out STD_LOGIC);
end even_detector;

architecture Behavioral of even_detector is
    signal p1, p2, p3, p4 : std_logic;
begin

    even <= (p1 or p2) or (p3 or p4) after 20 ns;
    p1 <= (not a(2)) and (not a(1)) and (not a(0)) after 15 ns;
    p2 <= (not a(2)) and a(1) and a(0) after 12 ns;
    p3 <= a(2) and (not a(1)) and a(0) after 12 ns;
    p4 <= a(2) and a(1) and (not a(0)) after 12 ns;

end Behavioral;
VHDL description of the even detection circuit

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity even_detector is
  Port ( a : in  STD_LOGIC_VECTOR (2 downto 0);
        even : out  STD_LOGIC);
end even_detector;

architecture Behavioral of even_detector is
  signal p1, p2, p3, p4 : std_logic;
begin
  even <= (p1 or p2) or (p3 or p4) after 20 ns;
p1 <= (not a(2)) and (not a(1)) and (not a(0)) after 15 ns;
p2 <= (not a(2)) and a(1) and a(0) after 12 ns;
p3 <= a(2) and (not a(1)) and a(0) after 12 ns;
p4 <= a(2) and a(1) and (not a(0)) after 12 ns;
end Behavioral;
VHDL description of the even detection circuit

Entity declaration

Specifying the circuit’s input and output ports
- Input ports: \( a(2), a(1), a(0) \)
- Output port: \( \text{even} \)

It provides a way for connecting the circuit to outside. Or the interface.
**VHDL description of the even detection circuit**

**Architecture body**

- Specifying the circuit's internal operation or organization
- The fundamental building block inside the architecture body is the concurrent statement.

```vhdl
architecture Behavioral of even_detector is
  signal p1, p2, p3, p4 : std_logic;
begin
  even <= (p1 or p2) or (p3 or p4) after 20 ns;
  p1 <= (not a(2)) and (not a(1)) and (not a(0)) after 15 ns;
  p2 <= (not a(2)) and a(1) and a(0) after 12 ns;
  p3 <= a(2) and (not a(1)) and a(0) after 12 ns;
  p4 <= a(2) and a(1) and (not a(0)) after 12 ns;
end Behavioral;
```

**Description of circuit’s operation or organization**

- **a(2)**
- **a(1)**
- **a(0)**

**Signal declaration**
VHDL description of the even detection circuit

```vhdl
even <= (p1 or p2) or (p3 or p4) after 20 ns;
```

The result of the operation is available at the output after a specific amount of a propagation delay.
VHDL description of the even detection circuit

```vhdl
even <= (p1 or p2) or (p3 or p4) after 20 ns;
p1 <= (not a(2)) and (not a(1)) and (not a(0)) after 15 ns;
p2 <= (not a(2)) and a(1) and a(0) after 12 ns;
p3 <= a(2) and (not a(1)) and a(0) after 12 ns;
p4 <= a(2) and a(1) and (not a(0)) after 12 ns;
```

Diagram showing the logic gates and connections for the even detection circuit.
VHDL description of the even detection circuit

```vhdl
even <= (p1 or p2) or (p3 or p4) after 20 ns;
p1 <= (not a(2)) and (not a(1)) and (not a(0)) after 15 ns;
p2 <= (not a(2)) and a(1) and a(0) after 12 ns;
p3 <= a(2) and (not a(1)) and a(0) after 12 ns;
p4 <= a(2) and a(1) and (not a(0)) after 12 ns;
```

Concurrent statement

- Order of the concurrent statements does not matter
- Concurrent statements are independent and can be activated in parallel
- The incorporation of propagation delay is the key ingredient in:
  - Modeling the operation of hardware, and,
  - Insuring the proper interpretation of VHDL code.
VHDL description of the even detection circuit

- **Structural description**

  ✓ for describing the structural view, a circuit is constructed from smaller parts.

  ✓ Structural description specifies
    - What types of parts are used, and
    - How these parts are connected.

  ✓ Treating a concurrent statement as a circuit part = out interpretation

  ✓ Formal VHDL structural description uses the concept of component
    - A component can be either an existing or a hypothetical part
    - It must first declared (make known), then
    - Can be actually used in the architecture body as needed
VHDL description of the even detection circuit

### Structural description

```vhdl
architecture str_arch of even_detector is
    component comp_p1
        Port ( i1 : in STD_LOGIC_VECTOR (2 downto 0);
            o1 : out STD_LOGIC);
    end component;
    component comp_p2
        Port ( i1 : in STD_LOGIC_VECTOR (2 downto 0);
            o1 : out STD_LOGIC);
    end component;
    component comp_p3
        Port ( i1 : in STD_LOGIC_VECTOR (2 downto 0);
            o1 : out STD_LOGIC);
    end component;
    component comp_p4
        Port ( i1 : in STD_LOGIC_VECTOR (2 downto 0);
            o1 : out STD_LOGIC);
    end component;
    component comp_even
        Port ( i1, i2, i3, i4 : in STD_LOGIC;
            o1 : out STD_LOGIC);
    end component;
signal p1, p2, p3, p4 : std_logic;
beg<20><10>in
    unit1: comp_p1 port map (i1=>a, o1=>p1);
    unit2: comp_p2 port map (i1=>a, o1=>p2);
    unit3: comp_p3 port map (i1=>a, o1=>p3);
    unit4: comp_p4 port map (i1=>a, o1=>p4);
    unit5: comp_even port map (i1=>p1, i2=>p2,i3=>p3, i4=>p4, o1=>even );
end str_arch;
```

Dr. Mohamed Abd el Ghany
Department of Electronics and Electrical Engineering
VHDL description of the even detection circuit

### Structural description

```vhdl
entity comp_p1 is
  Port ( i1 : in STD_LOGIC_VECTOR (2 downto 0);
        o1 : out STD_LOGIC);
end comp_p1;
architecture Behavioral of comp_p1 is
begin
  o1 <= (not i1(2)) and (not(i1(1)) and (not i1(0)));
end Behavioral;

description

entity comp_p2 is
  Port ( i1 : in STD_LOGIC_VECTOR (2 downto 0);
        o1 : out STD_LOGIC);
end comp_p2;
architecture Behavioral of comp_p2 is
begin
  o1 <= (not i1(2)) and i1(1) and i1(0);
end Behavioral;

description

entity comp_p3 is
  Port ( i1 : in STD_LOGIC_VECTOR (2 downto 0);
        o1 : out STD_LOGIC);
end comp_p3;
architecture Behavioral of comp_p3 is
begin
  o1 <= i1(2) and (not(i1(1)) and i1(0));
end Behavioral;

description

entity comp_p4 is
  Port ( i1 : in STD_LOGIC_VECTOR (2 downto 0);
        o1 : out STD_LOGIC);
end comp_p4;
architecture Behavioral of comp_p4 is
begin
  o1 <= i1(2) and i1(1) and (not i1(0));
end Behavioral;
```

Dr. Mohamed Abd el Ghany
Department of Electronics and Electrical Engineering

ELCT 501: Digital System Design
VHDL description of the even detection circuit

- **Structural description (structural view)**
  - Facilitate the hierarchical design
  - Provide a method to use predesigned circuits
    - IP cores
    - Library cells from device vendors (Xilinx, Altera...)

---

Dr. Mohamed Abd el Ghany  
Department of Electronics and Electrical Engineering

ELCT 501: Digital System Design
Behavioral Descriptions

- The Process Statement
- Can appear in the body of an architecture declaration
- Can include sequential statements like those found in software programming languages
- These statements are used to compute the outputs of the process from its inputs
- Sequential statements are often more powerful, but sometimes have no direct correspondence to a hardware implementation.
Behavioral Descriptions

- **The Process Statement**
  - Sensitivity list – a set of signals
    - When a signal in the list changes, the process is activated.
  - Inside the process the semantics is similar to that of a traditional programming
    - Variable can be used
    - Sequential execution

The basic skeleton of a process

```plaintext
Process(sensitivity_list)  
variable declaration;  
Begin  
  Sequential statements;  
End process;
```
Behavioral Descriptions

**Example:** describe the even detection circuit using behavioral description

```vhdl
architecture Behavioral of even_detection is 
begin 
    process(a)
        variable sum, r : integer;
    begin
        sum := 0;
        for i in 2 downto 0 loop
            if a(i)='1' then
                sum := sum +1;
            end if;
        end loop;
        r := sum mod 2;
        if (r=0) then
            even <= '1';
        else
            even <= '0';
        end if;
    end process;
end Behavioral;
```

While the code is very straightforward and easy to understand, it provides no clues about the underlying structure or how to realize the code in hardware. When an even occurs on a signal in the sensitivity list, the process is said to be resumed and the statements will be executed from top to bottom again.
Signals vs. Variables

**Signals**

- Signals follow the notion of ‘event scheduling’
- An event is characterized by a (time, value) pair
- Signal assignment example:
  
  $X <= xtmp;$, means schedule the assignment of the value of signal $xtmp$ to signal $x$ at (current time $+delta$)

  Where delta: infinitesimal time unit used by simulator for processing the signals

**Variables**

- Variables do not have notion of ‘events’
- Variables can be defined and used only inside the process block and some other special blocks
- Assignment takes effect immediately
- Used for loop counters, temp storage, etc
VHDL Simulation and Testbench

- One major use of VHDL code (program) is the simulation
- To study the operation of the circuits, or
- To verify the correctness of the design
- Performing simulation is similar to doing experiment with physical circuits

*Doiing experiment*

- A stimulus e.g. signal generators
- Output observer e.g. logic analyzer
- A physical Circuit

*Performing simulation*

- **testbech**
  - A stimulus generator or test vector generator
  - VHDL utility routines
  - Wave editors
- Output observer
  - VHDL utility routines
  - Human
- A model of the circuit (e.g. VHDL code)
Quiz 1