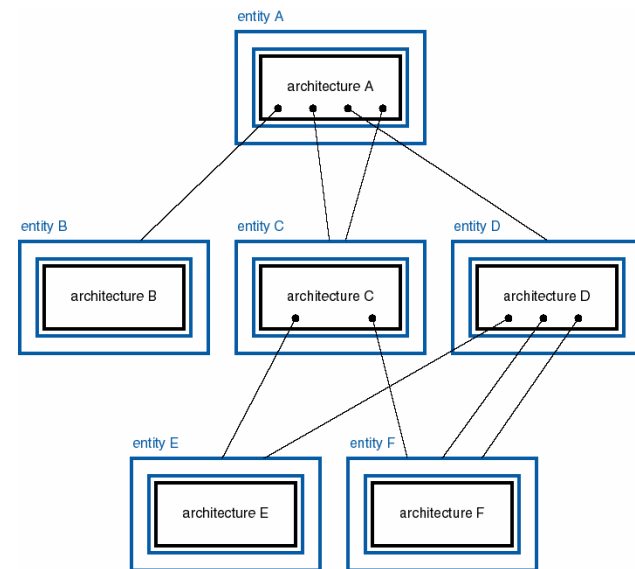


```
port (  
  I: in STD_LOGIC_VECTOR (1 to 9);  
  EVEN, ODD: out STD_LOGIC  
);
```

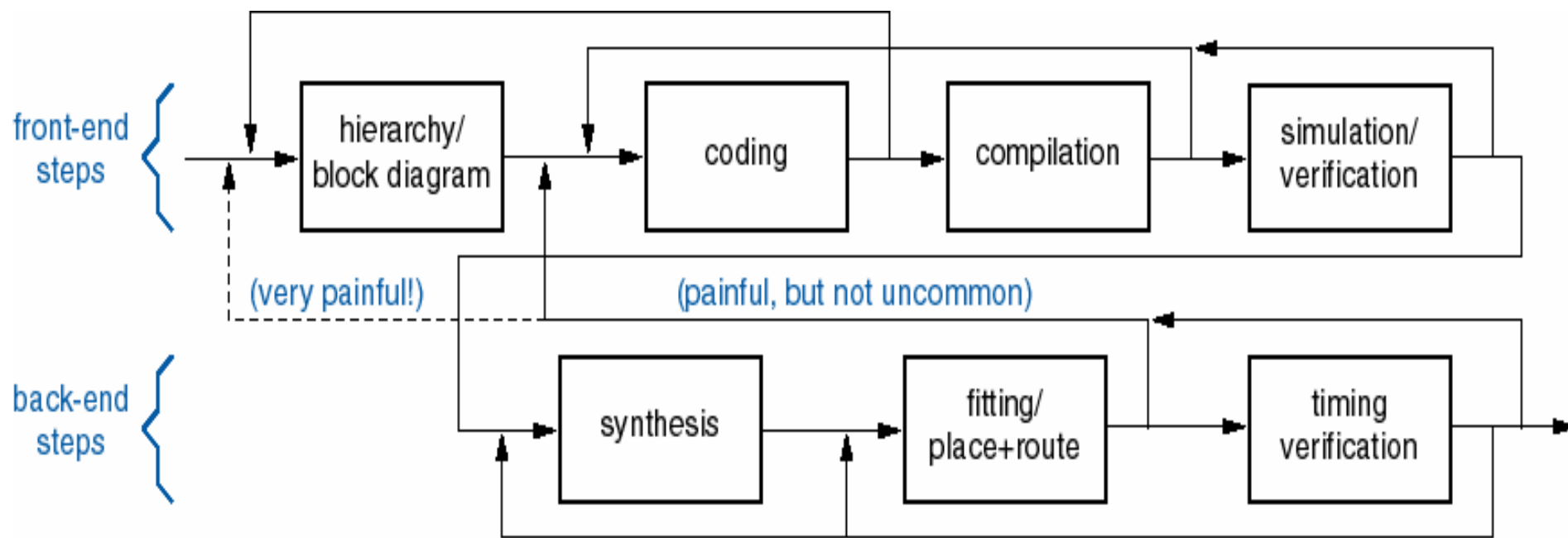
VHDL Design Principles

ECGR2181



Reading: Chapter 5.0, 5.1, 5.3

HDL-based design flow



- For ASICs, verification and fitting phases are usually much longer (as a fraction of overall project time) than what you've experienced in class.

VHDL

Developed in the mid-1980s under DoD sponsorship

- Mandated for federally-sponsored VLSI designs

Used for design description, simulation, and synthesis

- Synthesis became practical in the early 90s and use of VHDL (and Verilog) has taken off since then

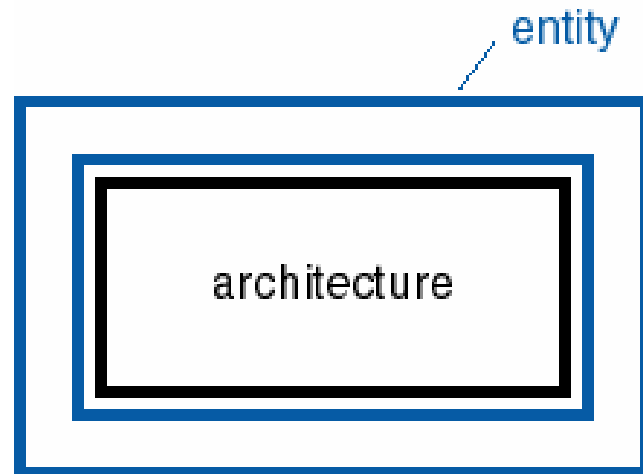
Only a subset of the language can be synthesized

VHDL entity and architecture concept

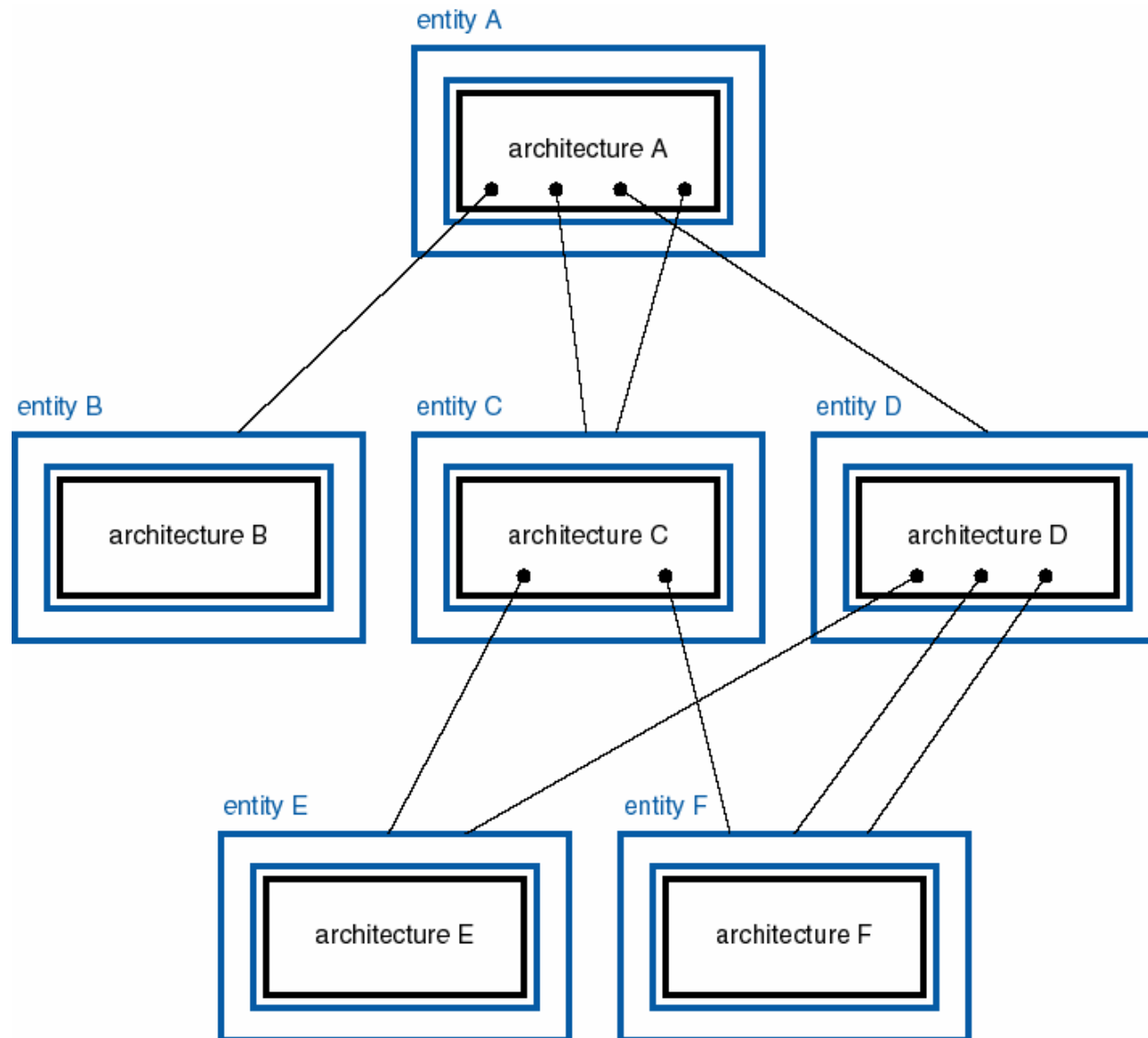
System is a collection of modules.

Architecture: detailed description of the internal structure or behavior of a module.

Entity: a “wrapper” for the architecture that exposes only its external interfaces, hiding the internal details.



VHDL Hierarchy



VHDL program file structure

text file (e.g., mydesign.vhd)

entity declaration

architecture definition

```
entity Inhibit is
  port (X,Y: in BIT;
        Z:   out BIT);
end Inhibit;

architecture Inhibit_arch of Inhibit is
begin
  Z <- '1' when X='1' and Y='0' else '0';
end Inhibit_arch;
```

Entity and architecture definitions for different modules can be in different files.

- Compiler maintains “work” library and keeps track of definitions using entity and architecture names.

VHDL -- designed by committee

Tries to be all things to all people.

- Result -- very general, but also very complex.

Standard logic values and elements are not built-in.

Standard logic defined by a “package”, IEEE 1164 STD_LOGIC.

- Must be explicitly “used” by program.

```
library IEEE;
use IEEE.std_logic_1164.all;
```

library name

package name

Use all definitions in package

Compiler knows where to find this (system-dependent)

Standard logic values -- not just 0,1

Need additional values for simulation, three-state logic, pull-ups, etc.

- Defined in IEEE 1164 STD_LOGIC package.

```
type STD_ULOGIC is ( 'U', -- Uninitialized
                    'X', -- Forcing Unknown
                    '0', -- Forcing 0
                    '1', -- Forcing 1
                    'Z', -- High Impedance
                    'W', -- Weak Unknown
                    'L', -- Weak 0
                    'H', -- Weak 1
                    '- ' -- Don't care
                    );
subtype STD_LOGIC is resolved STD_ULOGIC;
```


Logic functions defined by table lookup

```
-- truth table for "and" function
CONSTANT and_table : stdlogic_table := (
--
--   | U   X   0   1   Z   W   L   H   -   |
--   -----
--   ( 'U', 'U', '0', 'U', 'U', 'U', '0', 'U', 'U' ), -- | U |
--   ( 'U', 'X', '0', 'X', 'X', 'X', '0', 'X', 'X' ), -- | X |
--   ( '0', '0', '0', '0', '0', '0', '0', '0', '0' ), -- | 0 |
--   ( 'U', 'X', '0', '1', 'X', 'X', '0', '1', 'X' ), -- | 1 |
--   ( 'U', 'X', '0', 'X', 'X', 'X', '0', 'X', 'X' ), -- | Z |
--   ( 'U', 'X', '0', 'X', 'X', 'X', '0', 'X', 'X' ), -- | W |
--   ( '0', '0', '0', '0', '0', '0', '0', '0', '0' ), -- | L |
--   ( 'U', 'X', '0', '1', 'X', 'X', '0', '1', 'X' ), -- | H |
--   ( 'U', 'X', '0', 'X', 'X', 'X', '0', 'X', 'X' ) -- | - |
);

FUNCTION "and" ( L : std_ulogic; R : std_ulogic ) RETURN UXO1 IS
BEGIN
    RETURN (and_table(L, R));
END "and";
```

VHDL strong typing

Every signal, variable, function parameter, and function result has a “type”.

- A few built-in types, plus user defined types.

In assignment statements, comparisons, and function calls, types must match.

Commonly used IEEE-1164 types:

- STD_LOGIC (one bit)
- STD_LOGIC_VECTOR(*range*) (multibit vector)
- INTEGER (built-in integer type)

Pain in the neck: Must explicitly convert between INTEGER and STD_LOGIC_VECTOR.

```
library IEEE;
use IEEE.std_logic_1164.all;

entity parity9 is
  port (
    I: in STD_LOGIC_VECTOR
      (1 to 9);
    EVEN, ODD: out
      STD_LOGIC
  );
end parity9;
```

```
architecture parity9p of parity9 is
begin
  process (I)
    variable p : STD_LOGIC;
    variable j : INTEGER;
  begin
    p := I(1);
    for j in 2 to 9 loop
      if I(j) = '1' then p := not p; end if;
    end loop;
    ODD <= p;
    EVEN <= not p;
  end process;
end parity9p;
```

VHDL programming styles

Structural

- Define explicit components and the connections between them.
- Textual equivalent of drawing a schematic

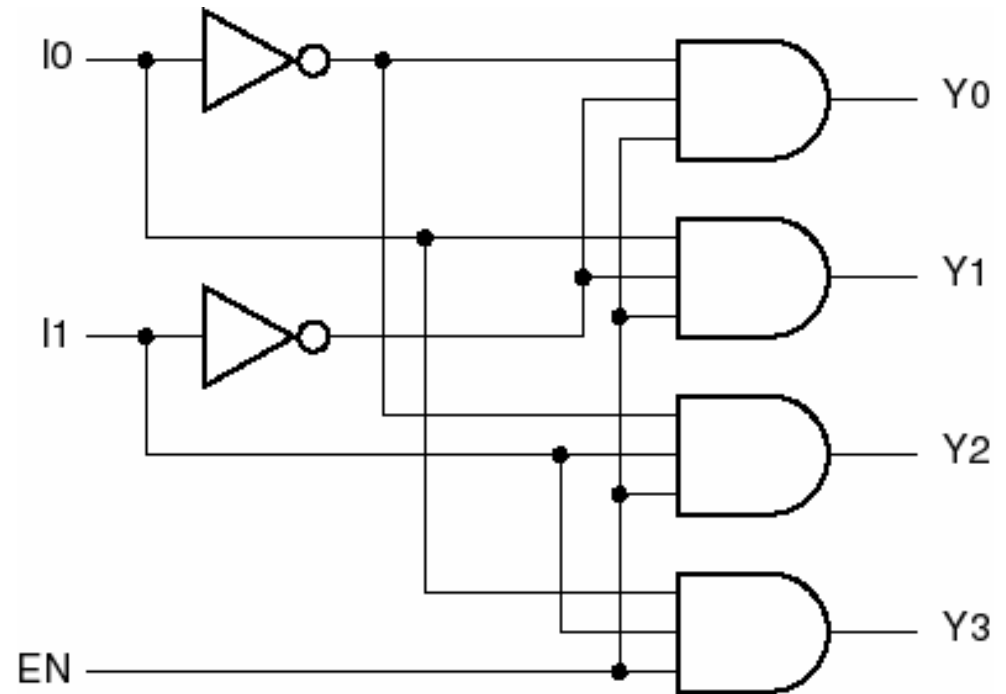
Dataflow

- Assign expressions to signals
- Includes “when” and “select” (case) statements

Behavioral

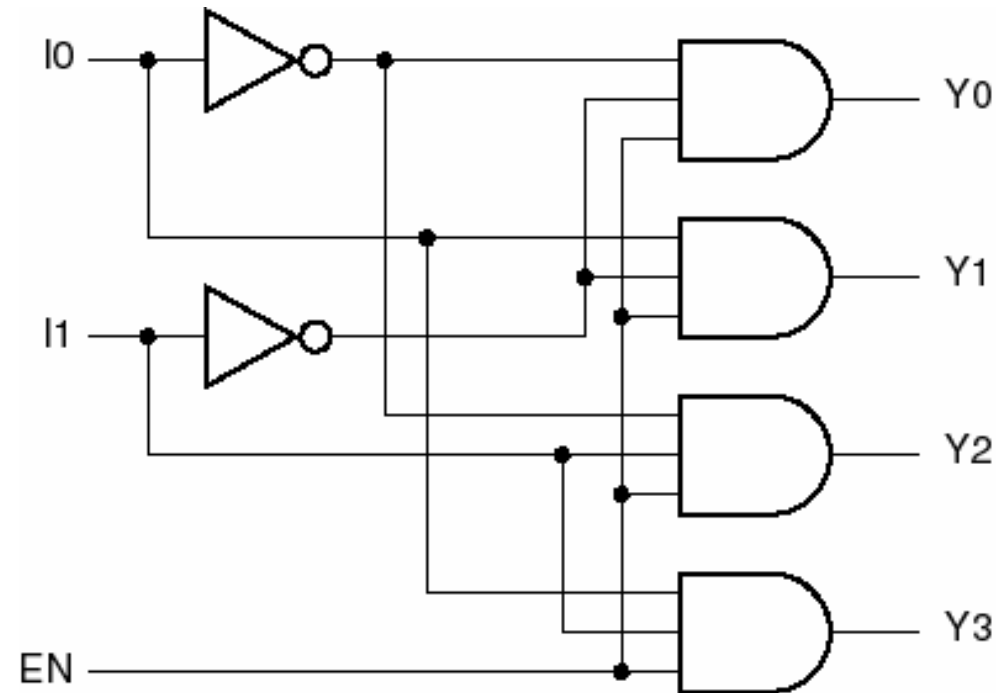
- Write an algorithm that describes the circuit’s output
- May not be synthesizable or may lead to a very large circuit
- Primarily used for simulation

Example: 2-to-4 decoder



EN	I1	I0	Y3	Y2	Y1	Y0

Example: 2-to-4 decoder



Entity

```
library IEEE;  
use IEEE.std_logic_1164.all;  
  
entity V2to4dec is  
    port (I0, I1, EN: in STD_LOGIC;  
          Y0, Y1, Y2, Y3: out STD_LOGIC );  
end V2to4dec;
```

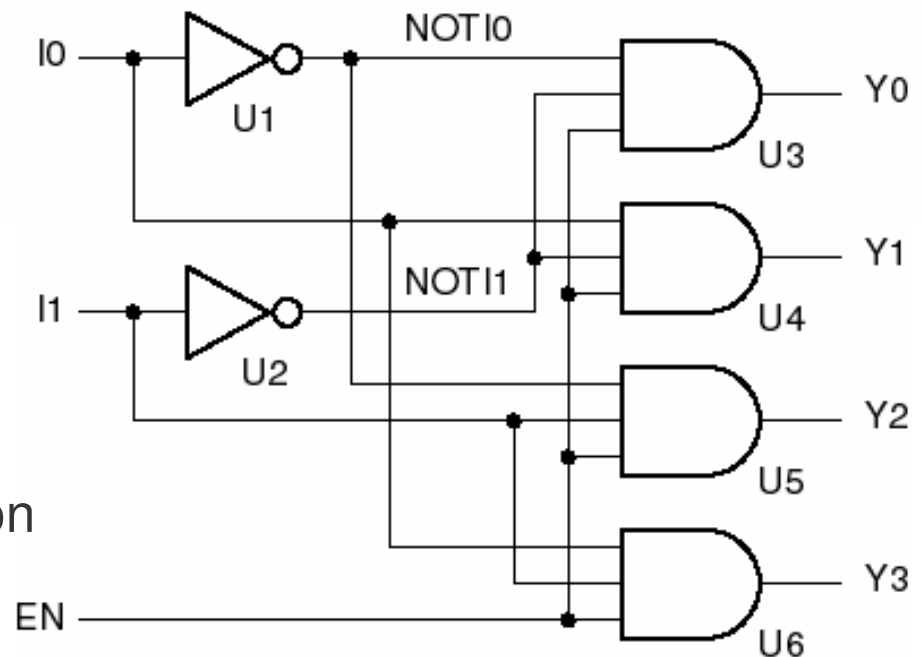
Example: 2-to-4 decoder

Architecture

```
architecture V2to4dec_s of V2to4dec is
  signal NOTI0, NOTI1: STD_LOGIC;
  component inv port (I: in STD_LOGIC; O: out STD_LOGIC ); end component;
  component and3 port (I0, I1, I2: in STD_LOGIC; O: out STD_LOGIC ); end component;
begin
  U1: inv port map (I0,NOTI0);
  U2: inv port map (I1,NOTI1);
  U3: and3 port map (NOTI0,NOTI1,EN,Y0);
  U4: and3 port map ( I0,NOTI1,EN,Y1);
  U5: and3 port map (NOTI0,  I1,EN,Y2);
  U6: and3 port map (  I0,  I1,EN,Y3);
end V2to4dec_s;
```

↑
built-in library
components

↑
positional
correspondence
with entity definition



Dataflow-style program for 3-to-8 decoder

```
library IEEE;
use IEEE.std_logic_1164.all;

entity V74x138 is
    port (G1, G2A_L, G2B_L: in STD_LOGIC;           -- enable inputs
          A: in STD_LOGIC_VECTOR (2 downto 0);     -- select inputs
          Y_L: out STD_LOGIC_VECTOR (0 to 7) );    -- decoded outputs
end V74x138;
```


Dataflow-style program for 3-to-8 decoder

```
architecture V74x138_b of V74x138 is
  signal G2A, G2B: STD_LOGIC;           -- active-high version of inputs
  signal Y: STD_LOGIC_VECTOR (0 to 7);  -- active-high version of outputs
  signal Y_s: STD_LOGIC_VECTOR (0 to 7); -- internal signal
begin
  G2A <- not G2A_L; -- convert inputs
  G2B <- not G2B_L; -- convert inputs
  Y_L <- Y;         -- convert outputs
  with A select Y_s <-
    "10000000" when "000",
    "01000000" when "001",
    "00100000" when "010",
    "00010000" when "011",
    "00001000" when "100",
    "00000100" when "101",
    "00000010" when "110",
    "00000001" when "111",
    "00000000" when others;
  Y <- not Y_s when (G1 and G2A and G2B)='1' else "00000000";
end V74x138_b;
```

Note: All assignment statements operate concurrently (combinational circuit).

Behavioral program style

Normally uses VHDL “processes”

Each VHDL process executes in parallel with other VHDL processes and concurrent statements

“Concurrent” statements include assignment and select statements in dataflow-style programs

Concurrency is needed to model the behavior of parallel, interconnected hardware elements

But “sequential” statements can be used within a process

VHDL process

```
process (signal-name, signal-name, ..., signal-name)  
  type declarations  
  variable declarations  
  constant declarations  
  function definitions  
  procedure definitions  
begin  
  sequential-statement  
  ...  
  sequential-statement  
end process;
```

A sequence of “sequential statements”.

Activated when any signal in the “sensitivity list” changes.

Primarily a simulation concept, but can be synthesized

Behavioral version of 74x138

```
architecture V3to8dec_b of V3to8dec is
    signal Y_s: STD_LOGIC_VECTOR (0 to 7);
begin
    process(A, G1, G2, G3, Y_s)
    begin
        case A is
            when "000" -> Y_s <- "10000000";
            when "001" -> Y_s <- "01000000";
            when "010" -> Y_s <- "00100000";
            when "011" -> Y_s <- "00010000";
            when "100" -> Y_s <- "00001000";
            when "101" -> Y_s <- "00000100";
            when "110" -> Y_s <- "00000010";
            when "111" -> Y_s <- "00000001";
            when others -> Y_s <- "00000000";
        end case;
        if (G1 and G2 and G3)='1' then Y <- Y_s;
        else Y <- "00000000";
        end if;
    end process;
end V3to8dec_b;
```

Except for different syntax, approach is not all that different from the dataflow version

Truly behavioral version

```
architecture V3to8dec_c of V3to8dec is
begin
process (G1, G2, G3, A)
variable i: INTEGER range 0 to 7;
begin
Y <- "00000000";
if (G1 and G2 and G3) = '1' then
for i in 0 to 7 loop
if i-CONV_INTEGER(A) then Y(i) <- '1'; end if;
end loop;
end if;
end process;
end V3to8dec_c;
```

← type conversion

May not be synthesizable, or may have a slow or inefficient realization. But just fine for simulation and verification.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

--Declaration of the module's inputs and outputs
entity fewgates is port (
    A: in std_logic;
    B: in std_logic;
    C: in std_logic;
    Y: out std_logic );
end fewgates;

--Defining the modules behavior
Architecture behavioral of fewgates is
begin
process (A, B, C) begin
    Y <= C OR ((NOT A) AND (NOT B));
end process;
end behavioral;
```

More VHDL

Powerful facilities for generating iterative circuit descriptions
(e.g., multiplier array)

Facilities for modeling timing behavior of known components

Program I/O facilities for use in simulation

Design-management facilities for selecting alternative
components and architectures

And more...