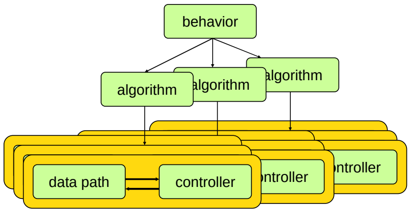
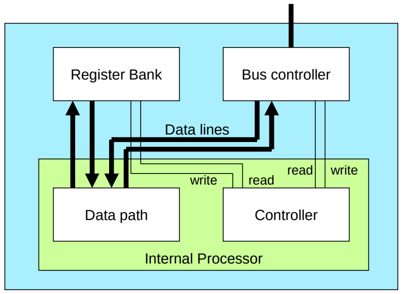
behavior ➔ algorithm ➔ datapath & controller

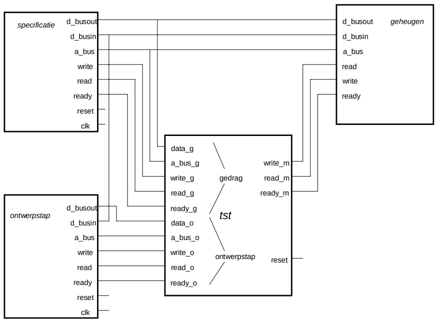
Global design flow



Processor: datapath & controller



Test environment: verification

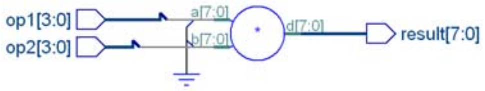


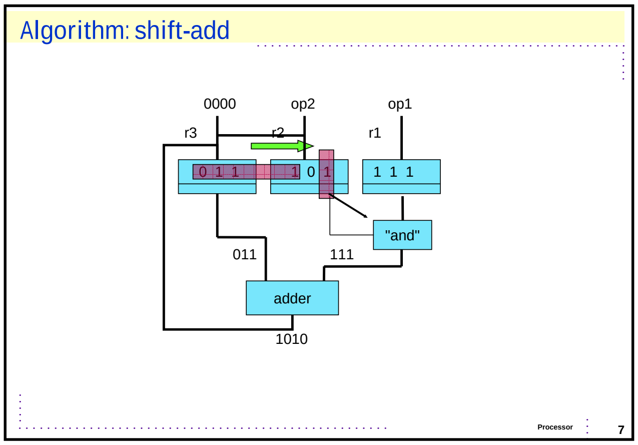
Multiplier: Behavior

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
ENTITY multiply IS
  GENERIC (width : integer := 4);
  PORT (op1, op2 : IN unsigned(width-1 DOWNTO 0);
        result   : OUT unsigned(2*width-1 DOWNTO 0));
END multiply;

ARCHITECTURE behaviour OF multiply IS
BEGIN
  result <= op1 * op2;
END behaviour;
```

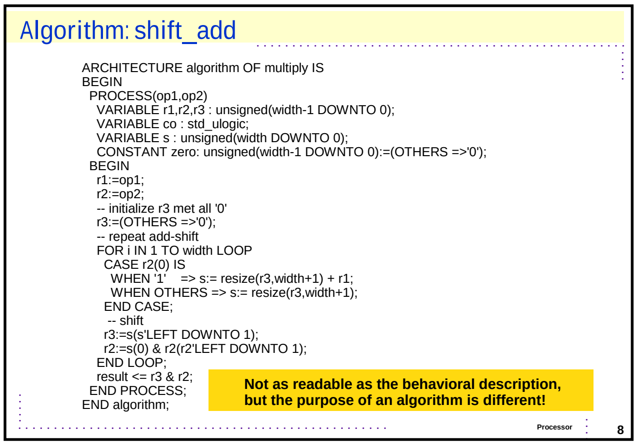
Notice that:

- **no implementation details are given**
- **description is readable**





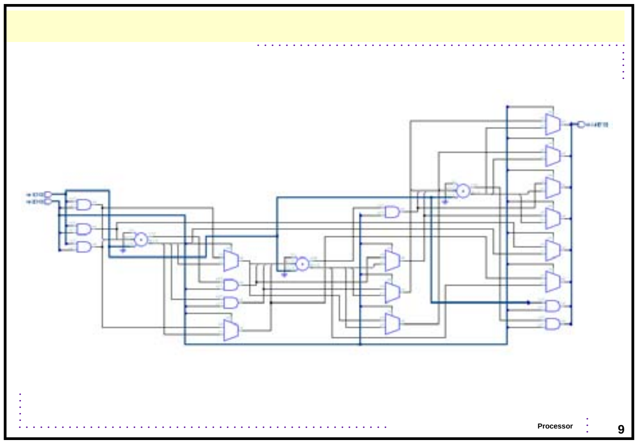
Algorithm: shift-add

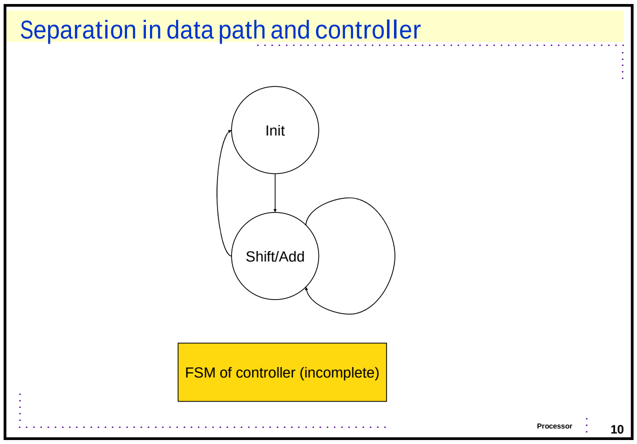


Algorithm: shift_add

```
ARCHITECTURE algorithm OF multiply IS
BEGIN
  PROCESS(op1,op2)
    VARIABLE r1,r2,r3 : unsigned(width-1 DOWNTO 0);
    VARIABLE co : std_ulogic;
    VARIABLE s : unsigned(width DOWNTO 0);
    CONSTANT zero: unsigned(width-1 DOWNTO 0):=(OTHERS =>'0');
  BEGIN
    r1:=op1;
    r2:=op2;
    -- initialize r3 met all '0'
    r3:=(OTHERS =>'0');
    -- repeat add-shift
    FOR i IN 1 TO width LOOP
      CASE r2(0) IS
        WHEN '1'    => s:= resize(r3,width+1) + r1;
        WHEN OTHERS => s:= resize(r3,width+1);
      END CASE;
       -- shift
      r3:=s(s'LEFT DOWNTO 1);
      r2:=s(0) & r2(r2'LEFT DOWNTO 1);
    END LOOP;
    result <= r3 & r2;
  END PROCESS;
END algorithm;
```

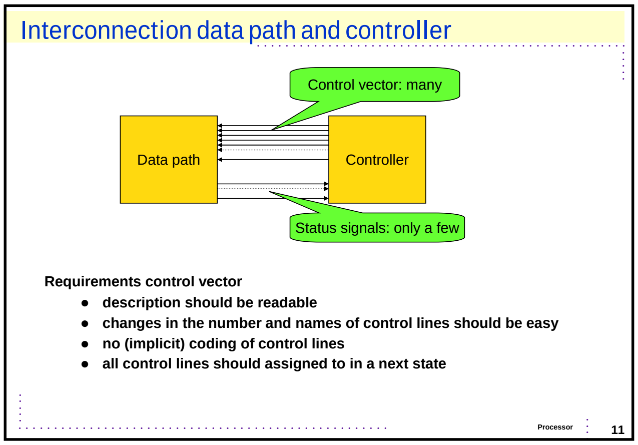
Not as readable as the behavioral description, but the purpose of an algorithm is different!





Separation in data path and controller

FSM of controller (incomplete)



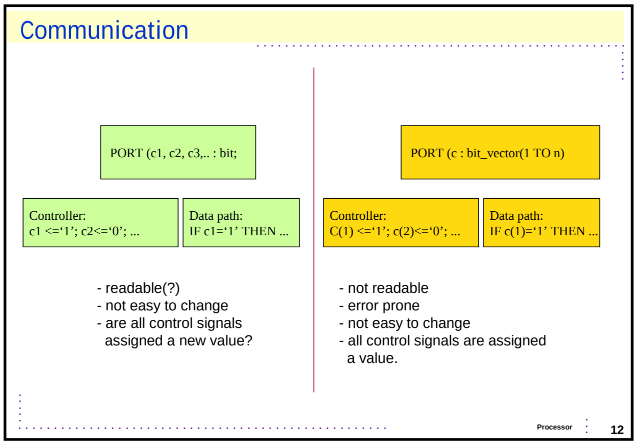
Interconnection data path and controller

Control vector: many

Status signals: only a few

Requirements control vector

- **description should be readable**
- **changes in the number and names of control lines should be easy**
- **no (implicit) coding of control lines**
- **all control lines should assigned to in a next state**



Communication

PORT (c1, c2, c3,.. : bit;

Controller:
c1 <=‘1’; c2<=‘0’; ...

Data path:
IF c1=‘1’ THEN ...

- readable(?)
- not easy to change
- are all control signals
  assigned a new value?

PORT (c : bit_vector(1 TO n)

Controller:
C(1) <=‘1’; c(2)<=‘0’; ...

Data path:
IF c(1)=‘1’ THEN ...

- not readable
- error prone
- not easy to change
- all control signals are assigned
  a value.

Communication: solution

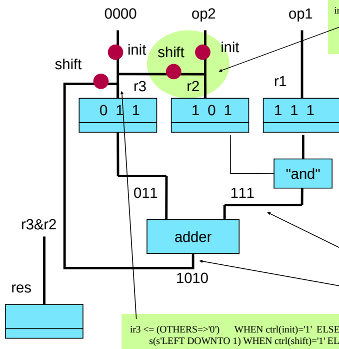
Package control_names, user defined!

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
PACKAGE control_names IS
  TYPE control_signals IS
     (init,shift,load,en_r1,en_r2,en_r3,en_res);

  -- do not change the following type declaration
  TYPE control_bus IS ARRAY (control_signals) OF std_logic;
END control_names;
```

```
ctrl : control_bus
Examples:
ctrl <= (OTHERS => '0');
ctrl <= (init => '1', OTHERS => '0'));
ctrl <= (init | shift => '1', OTHERS ='0');
```

Used in the entity declaration

Data path with gates & VHDL descriptions



```
ir2 <= op2 WHEN ctrl(init)='1'  ELSE
       s(0)&r2(r2'LEFT DOWNTO 1) WHEN ctrl(shift)='1' ELSE
       (OTHERS=>'-');
```

```
PROCESS
 BEGIN
  WAIT UNTIL rising_edge(clk);
  IF ctrl(en_r1)='1'  THEN r1<=op1; END IF;
  IF ctrl(en_r2)='1'  THEN r2<=ir2; END IF;
  IF ctrl(en_r3)='1'  THEN r3<=ir3; END IF;
  IF ctrl(en_res)='1' THEN result<=r3&r2; END IF;
 END PROCESS;
```

```
ar  <= r1 WHEN r2(0)='1' ELSE (OTHERS=>'0');
s <= resize(r3,width+1)+ ar;
```

```
ir3 <= (OTHERS=>'0')     WHEN ctrl(init)='1'  ELSE
       s(s'LEFT DOWNTO 1) WHEN ctrl(shift)='1' ELSE
       (OTHERS=>'-');
```

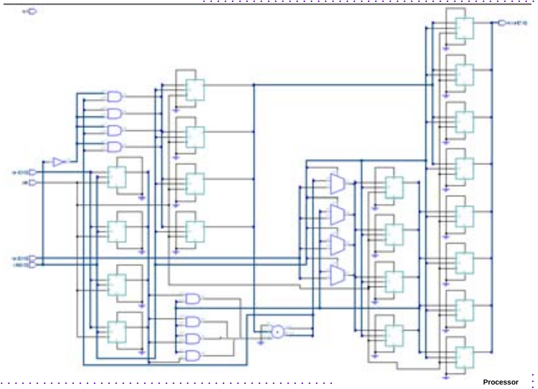
Datapath

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
USE work.control_names.ALL;
ENTITY datapath IS
  GENERIC (width : integer := 4);
  PORT (clk      : IN std_logic;
        rst      : IN std_logic;
        ctrl     : IN control_bus;
        op1, op2 : IN unsigned(width-1 DOWNTO 0);
        result   : OUT unsigned(2*width-1 DOWNTO 0));
END datapath;
```

Datapath

```
ARCHITECTURE RTL OF datapath IS
  SIGNAL ir2,ir3,ar,r1,r2,r3 : unsigned(width-1 DOWNTO 0);
  SIGNAL s : unsigned(width DOWNTO 0);
BEGIN

  ir2 <= op2                     WHEN ctrl(init)='1'  ELSE
         s(0)&r2(r2'LEFT DOWNTO 1) WHEN ctrl(shift)='1' ELSE
         (OTHERS=>'-');

  ir3 <= (OTHERS=>'0')     WHEN ctrl(init)='1'  ELSE
         s(s'LEFT DOWNTO 1) WHEN ctrl(shift)='1' ELSE
         (OTHERS=>'-');

  ar  <= r1 WHEN r2(0)='1' ELSE (OTHERS=>'0');
  s <= resize(r3,width+1)+ ar;

  PROCESS
  BEGIN
    WAIT UNTIL rising_edge(clk);
    IF ctrl(en_r1)='1'  THEN r1<=op1; END IF;
    IF ctrl(en_r2)='1'  THEN r2<=ir2; END IF;
    IF ctrl(en_r3)='1'  THEN r3<=ir3; END IF;
    IF ctrl(en_res)='1' THEN result<=r3&r2; END IF;
  END PROCESS;

END RTL;
```

Datapath



Controller: behavior

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
USE work.control_names.ALL;
ENTITY controller IS
  GENERIC (width : integer := 4);
  PORT (clk      : IN std_logic;
        rst      : IN std_logic;
        ctrl     : OUT control_bus);
END controller;

 ARCHITECTURE behaviour OF controller IS
 BEGIN
   PROCESS
   BEGIN
   reset_lp:LOOP
       -- reset
       ctrl<=(en_r1|en_r2|en_r3|init|en_res =>'1', OTHERS=>'0');
       WAIT UNTIL rising_edge(clk);
       EXIT WHEN rst='1';
       FOR i IN 1 TO width LOOP
         ctrl <= (en_r2|en_r3|shift=>'1', OTHERS=>'0');
         WAIT UNTIL clk='1';
         EXIT WHEN rst='1';
       END LOOP;
     END LOOP;
   END PROCESS;
 END behaviour;
```