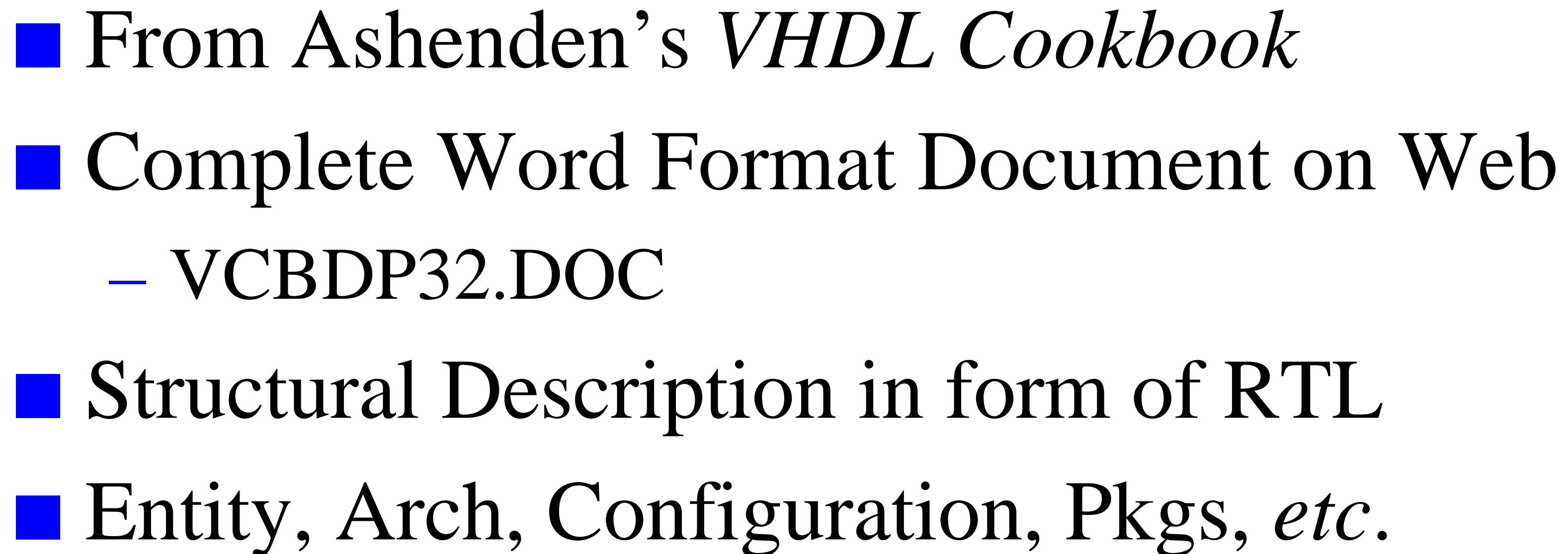
Case Study Components

Prof. K. J. Hintz

Department of Electrical
and
Computer Engineering
George Mason University

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

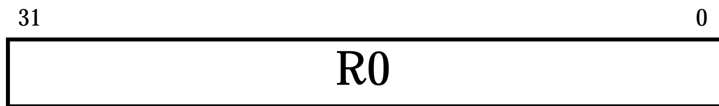
DP-32 Processor, RTL

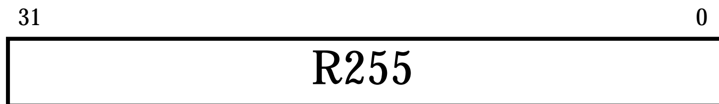


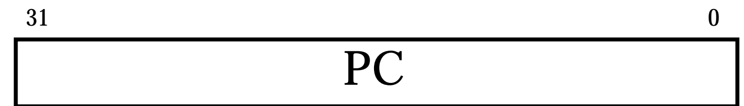
- From Ashenden's *VHDL Cookbook*
- Complete Word Format Document on Web
  – VCBDP32.DOC
- Structural Description in form of RTL
- Entity, Arch, Configuration, Pkgs, *etc*.

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

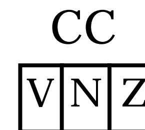
DP-32 Registers







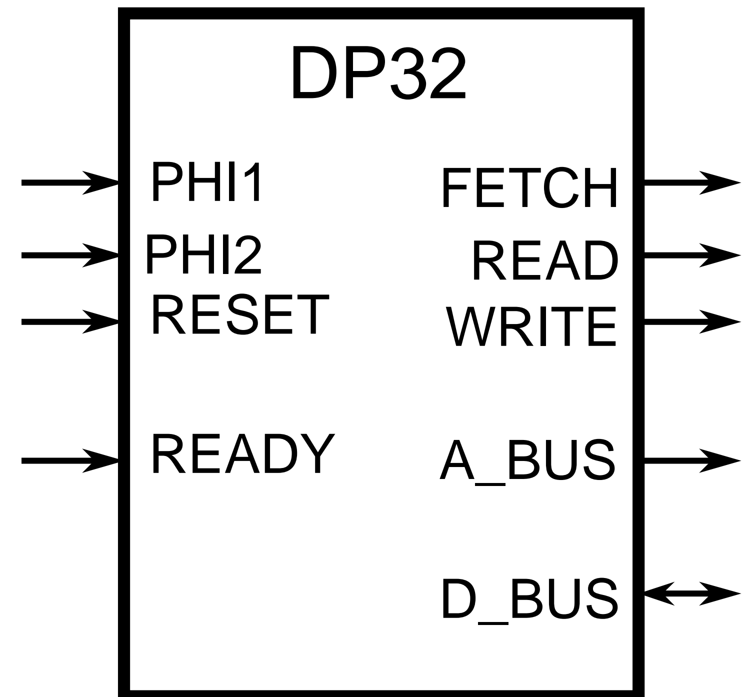
CC



Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

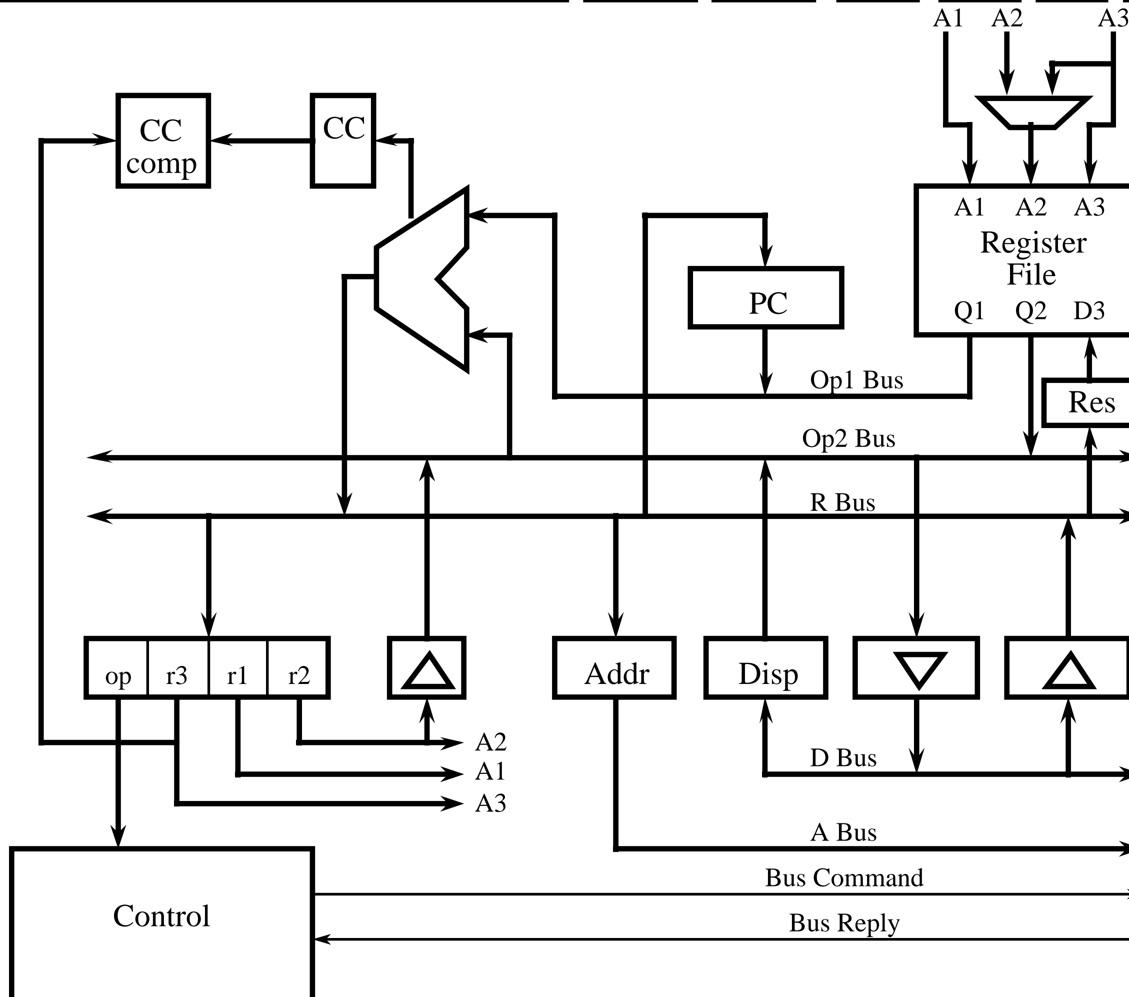
Port Diagram of DP32

- 32-bit Address and Data Busses
- D_Bus is Bidirectional
- Fetch Means Instruction Being Read



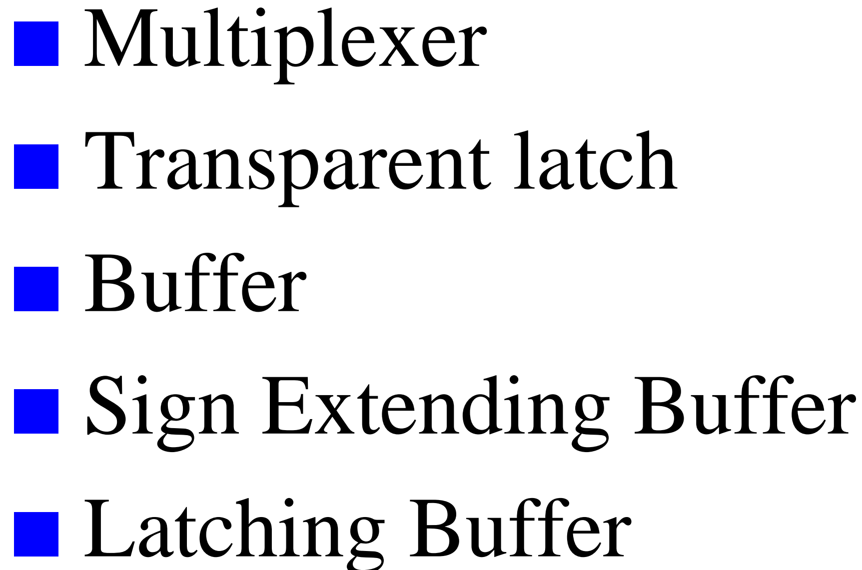
Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

RTL Implementation of DP32



Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

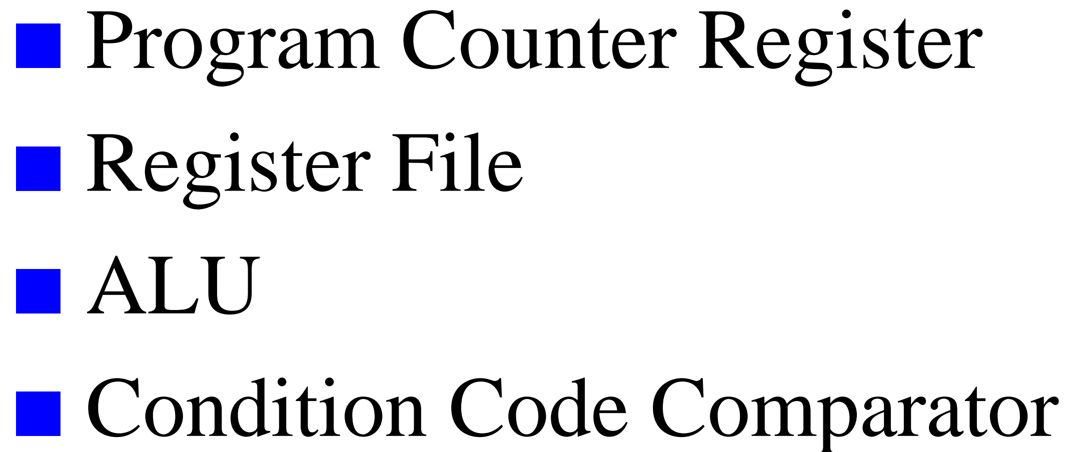
Components Needed



- Multiplexer
- Transparent latch
- Buffer
- Sign Extending Buffer
- Latching Buffer

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Components Needed



- Program Counter Register
- Register File
- ALU
- Condition Code Comparator

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Default

- Assumed to be included in all the following code:

```
use work.dp32_types.all ;
```

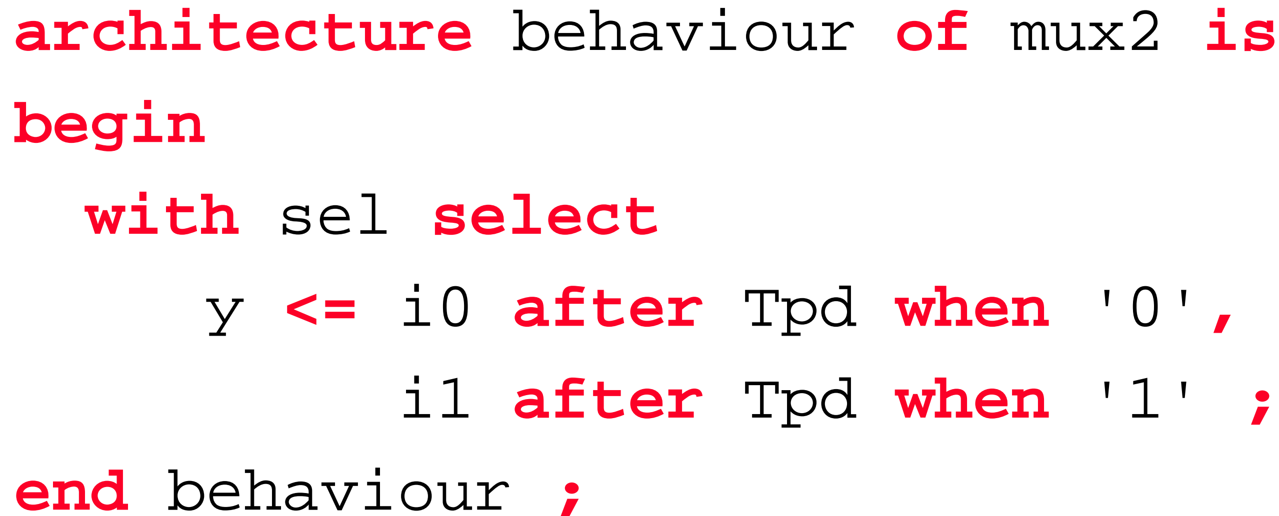
Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

2-input Multiplexer

```
entity mux2 is
 generic ( width : positive;
           Tpd : Time := unit_delay )
 port ( i0, i1 : in bit_vector
                 ( width-1 downto 0 );
        y      : out  bit_vector
                 ( width-1 downto 0 );
        sel    : in bit );
end mux2 ;
```

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

2-input Multiplexer



Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Transparent Latch

- Transparent
  - When ***en*** is '1', changes on ***d*** are transmitted through to ***q***
  - When ***en*** changes to '0'
    - » new value on ***d*** is ignored
    - » current value on q is maintained

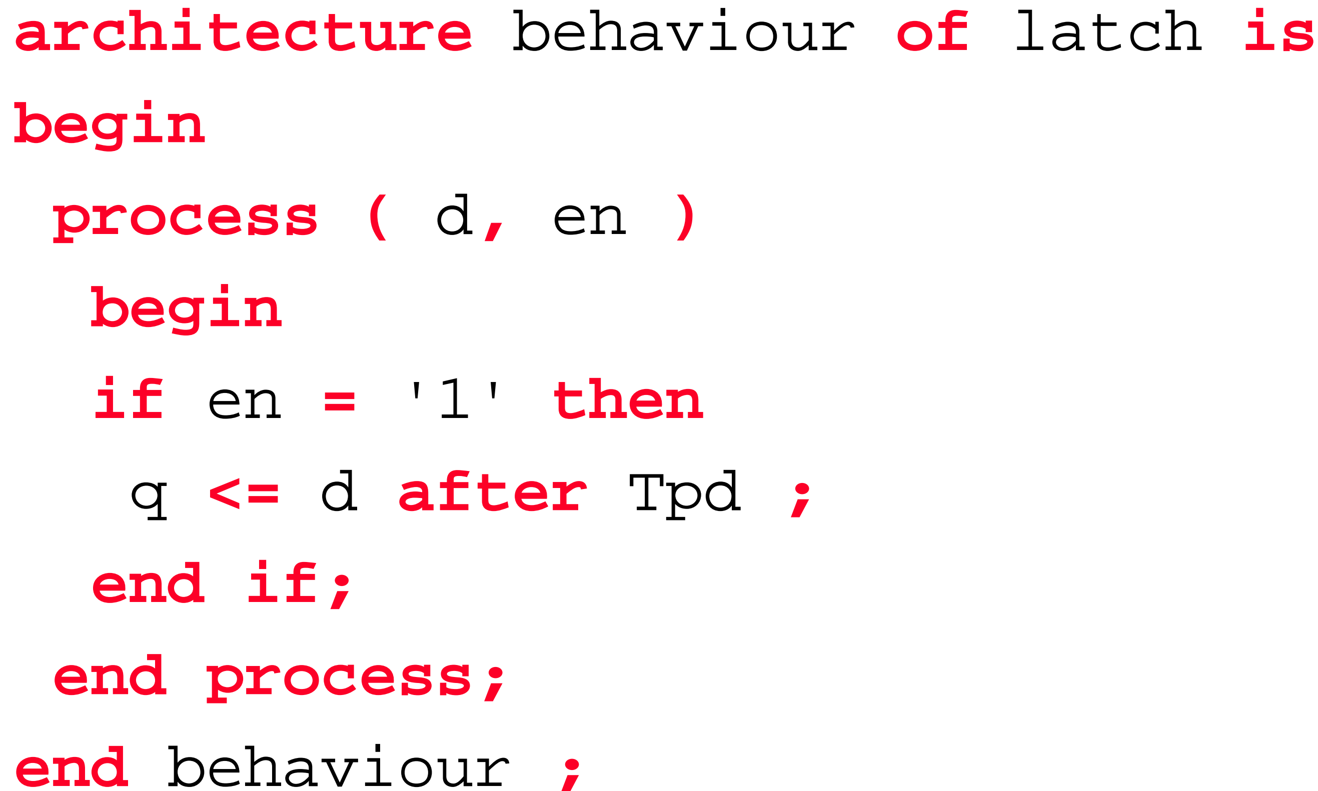
Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Transparent latch

```
entity latch is
 generic ( width : positive ;
           Tpd   : Time := unit_delay );
 port (    d     : in bit_vector
                  ( width-1 downto 0 );
           q     : out bit_vector
                  ( width-1 downto 0);
           en    : in bit );
end latch ;
```


Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Transparent latch



```
architecture behaviour of latch is
begin
 process ( d, en )
  begin
  if en = '1' then
   q <= d after Tpd ;
  end if;
 end process;
end behaviour ;
```

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Buffer

- Resolved Bit-Vector Bus Output B
  - A bus resolution function is specified in the definition of the port type.
- Buffer Is Implemented by a Process Sensitive to the ***en*** and ***a*** Inputs.
  - If ***en*** is '1', the ***a*** input is transmitted through to the ***b*** output.
  - If ***en*** is '0', the driver for ***b*** is disconnected, and the value on ***a*** is ignored

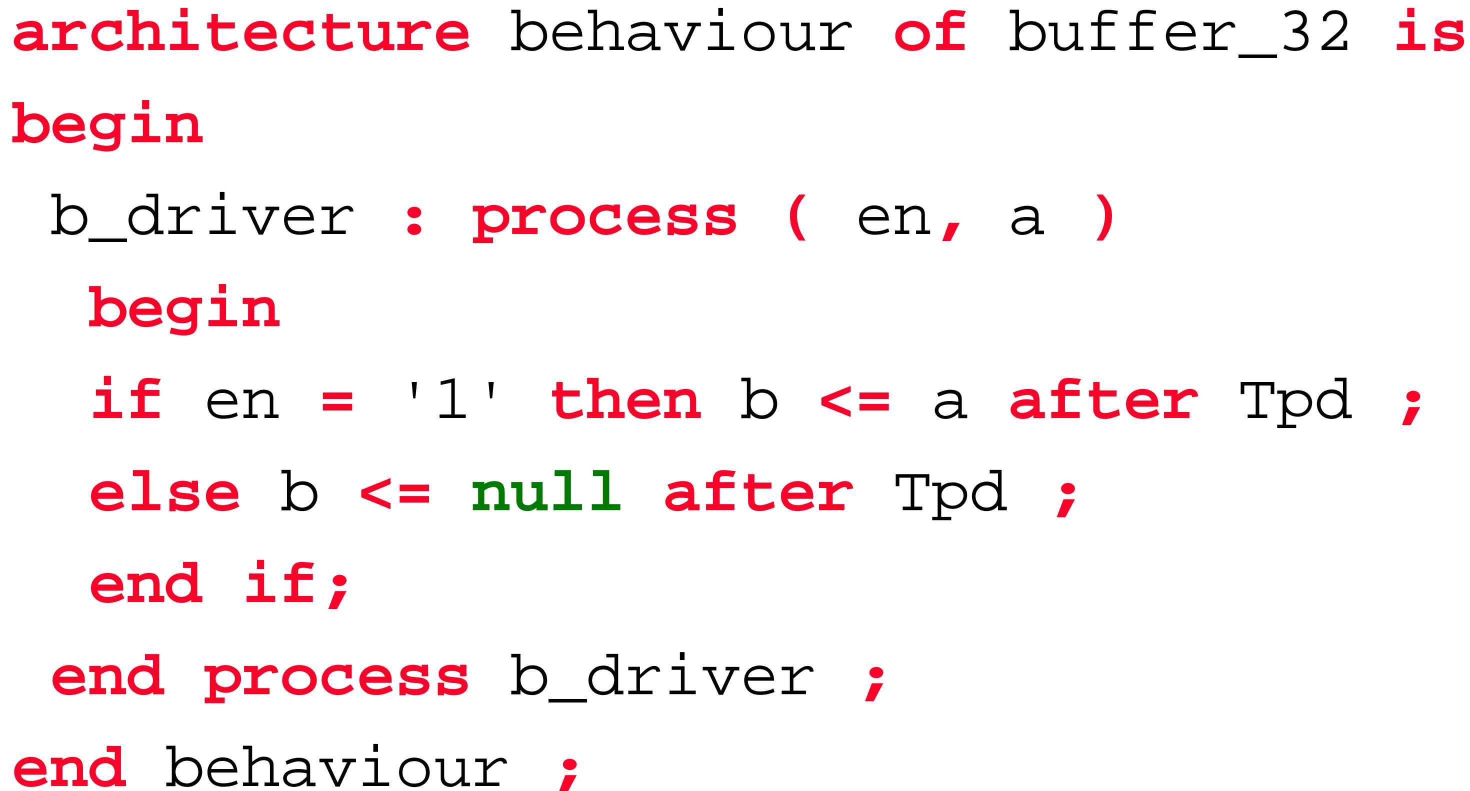
Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Buffer

```
entity buffer_32 is
 generic ( Tpd : Time := unit_delay ) ;
 port ( a : in bit_32 ;
        b : out bus_bit_32 bus ;
        en : in bit );
end buffer_32 ;
```

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Buffer



```
architecture behaviour of buffer_32 is
begin
 b_driver : process ( en, a )
  begin
  if en = '1' then b <= a after Tpd ;
  else b <= null after Tpd ;
  end if;
 end process b_driver ;
end behaviour ;
```

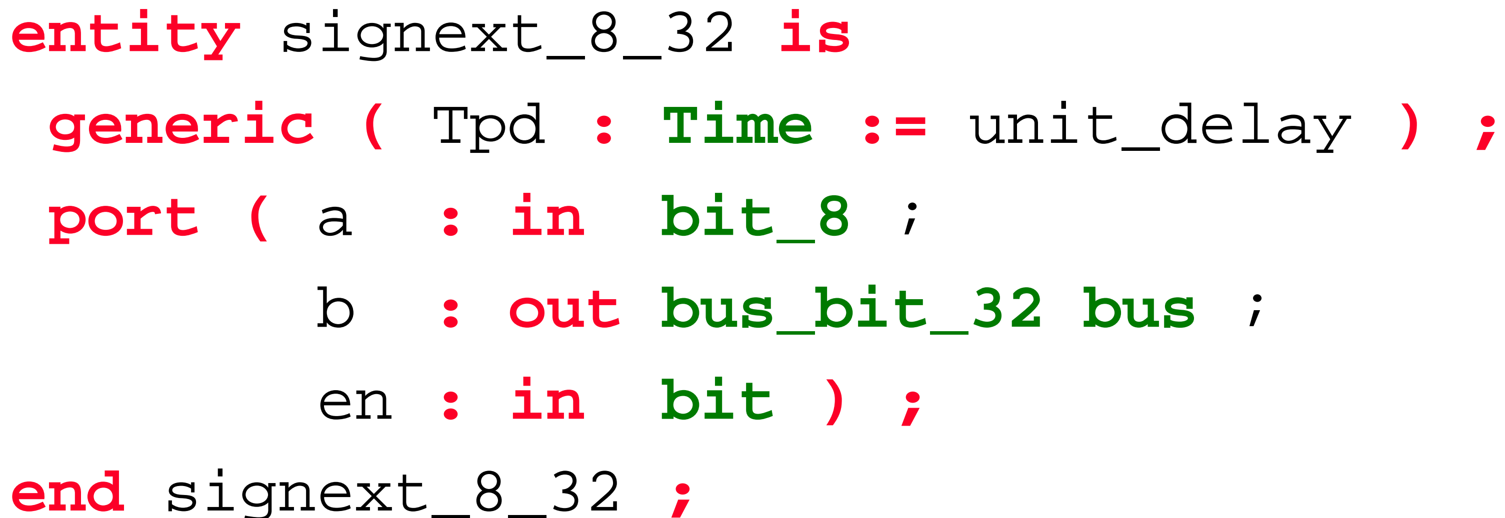
Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Sign-extending Buffer

- Input is 8-bit, 2's Complement
  - Leftmost bit is sign bit
    - 1 = negative number
    - Magnitude of negative number is found by taking 2's complement of the number
- Output is sign-extended, 32 bit
  - Sign bit is replicated from bit-7 through bit-31

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Sign Extending Buffer



```
entity signext_8_32 is
 generic ( Tpd : Time := unit_delay ) ;
 port ( a  : in  bit_8 ;
        b  : out bus_bit_32 bus ;
        en : in  bit ) ;
end signext_8_32 ;
```


Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Sign Extending Buffer

```
architecture behaviour of signext_8_32 is
begin
 b_driver : process ( en, a )
 begin
  if en = '1' then
   b( 7 downto 0 ) <= a after Tpd ;
  if a(7) = '1' then
   b( 31 downto 8 ) <= X"FFFF_FF" after Tpd ;
```

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Sign Extending Buffer

```
   else
    b(31 downto 8) <= X"0000_00" after Tpd;
   end if;
   else
    b <= null after Tpd ;
   end if ;
  end process b_driver ;
end behaviour ;
```

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Latching Buffer

```
entity latch_buffer_32 is
 generic ( Tpd : Time := unit_delay );
 port ( d        : in bit_32 ;
        q        : out bus_bit_32 bus ;
        latch_en : in bit ;
        out_en   : in bit) ;
end latch_buffer_32 ;
```

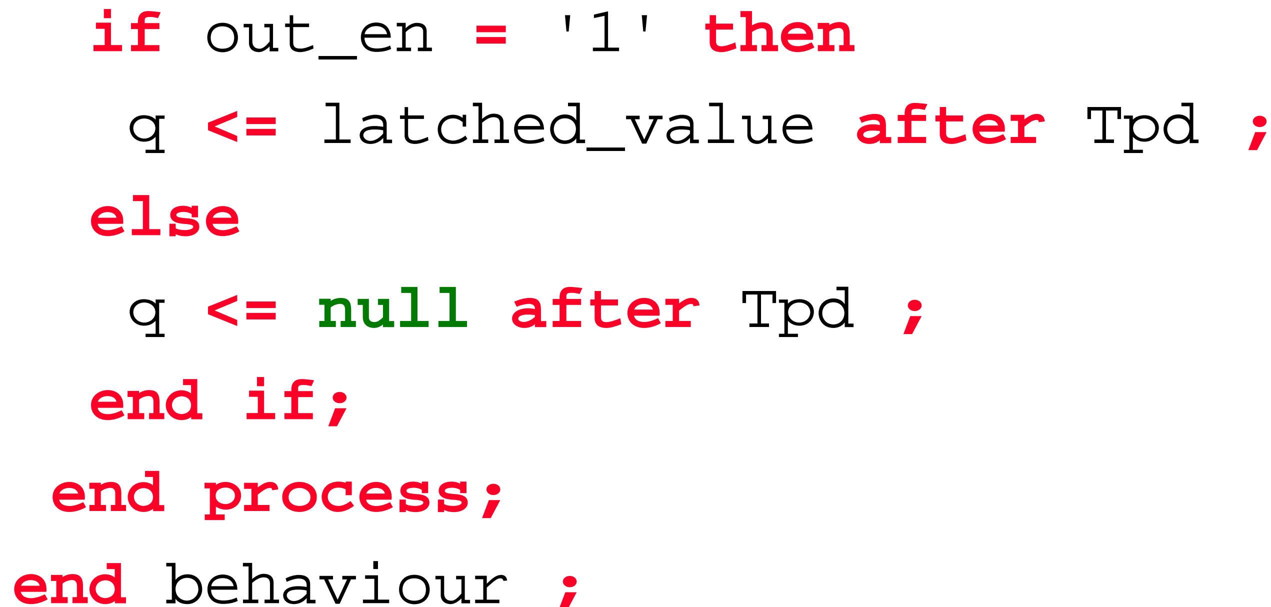
Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Latching Buffer

```
architecture behaviour of latch_buffer_32 is
begin
 process ( d, latch_en, out_en )
  variable latched_value : bit_32 ;
 begin
  if latch_en = '1' then
   latched_value := d ;
  end if ;
```

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Latching Buffer



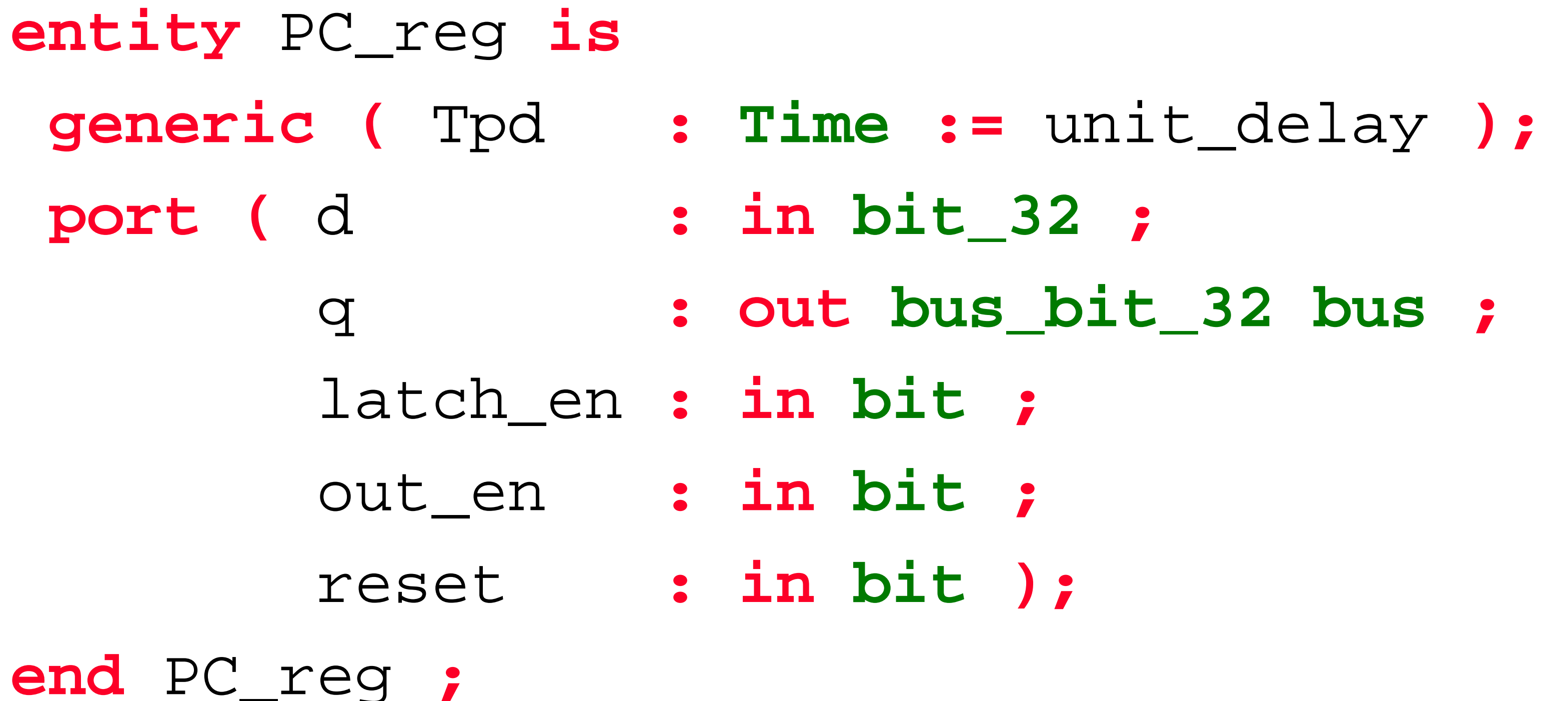
Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Program Counter Register

- Program Counter Register
  - Master/slave
  - Overriding reset on ‘1’
  - ***latch_en*** = '1'
    - » ***d*** input is stored in the variable ***master_PC***
    - » output (if enabled) is driven with value in ***slave_PC***
  - ***latch_en*** negative edge
    - » ***master_PC*** := ***slave_PC***
    - » Prevents race which would occur if normal register used

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Program Counter Register



```
entity PC_reg is
 generic ( Tpd        : Time := unit_delay );
 port ( d             : in bit_32 ;
        q             : out bus_bit_32 bus ;
        latch_en      : in bit ;
        out_en        : in bit ;
        reset         : in bit );
end PC_reg ;
```

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Program Counter Register

```
architecture behaviour of PC_reg is
begin
 process ( d, latch_en, out_en, reset )
 variable master_PC, slave_PC : bit_32 ;
 begin
  if reset = '1' then slave_PC := X"0000_0000";
  elsif latch_en = '1' then master_PC := d ;
  else
   slave_PC := master_PC ;
  end if;
```

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Program Counter Register

```
if out_en = '1' then
 q <= slave_PC after Tpd ;
else
 q <= null after Tpd ;
end if ;
end process ;
end behaviour ;
```

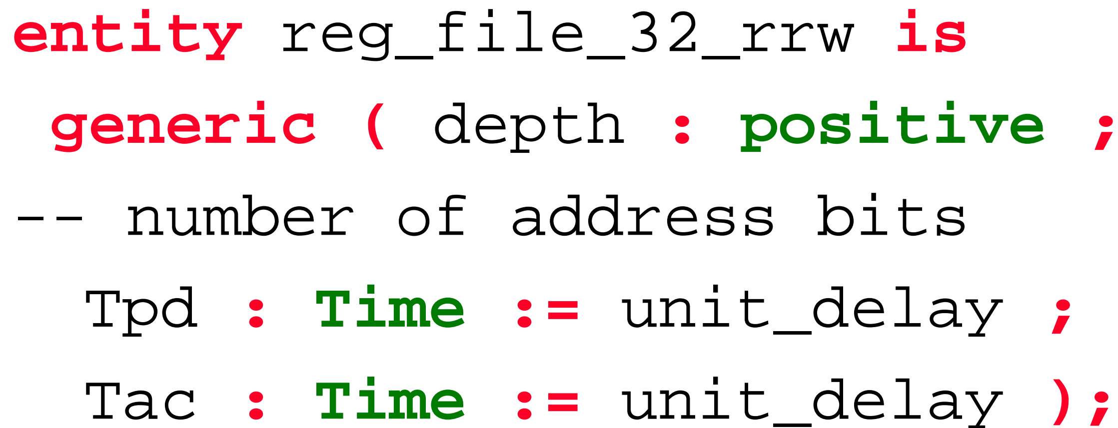
Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Register File

- 3 port register file
  - Two read ports
    - » data bus outputs (q1 and q2)
  - One write port
    - » address input (a1, a2 and a3)
      - number bits specified by the generic constant depth
    - » enable input (en1, en2 and en3)
    - » data input (d3)

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Register File



```
entity reg_file_32_rrw is
 generic ( depth : positive ;
-- number of address bits
  Tpd : Time := unit_delay ;
  Tac : Time := unit_delay );
```

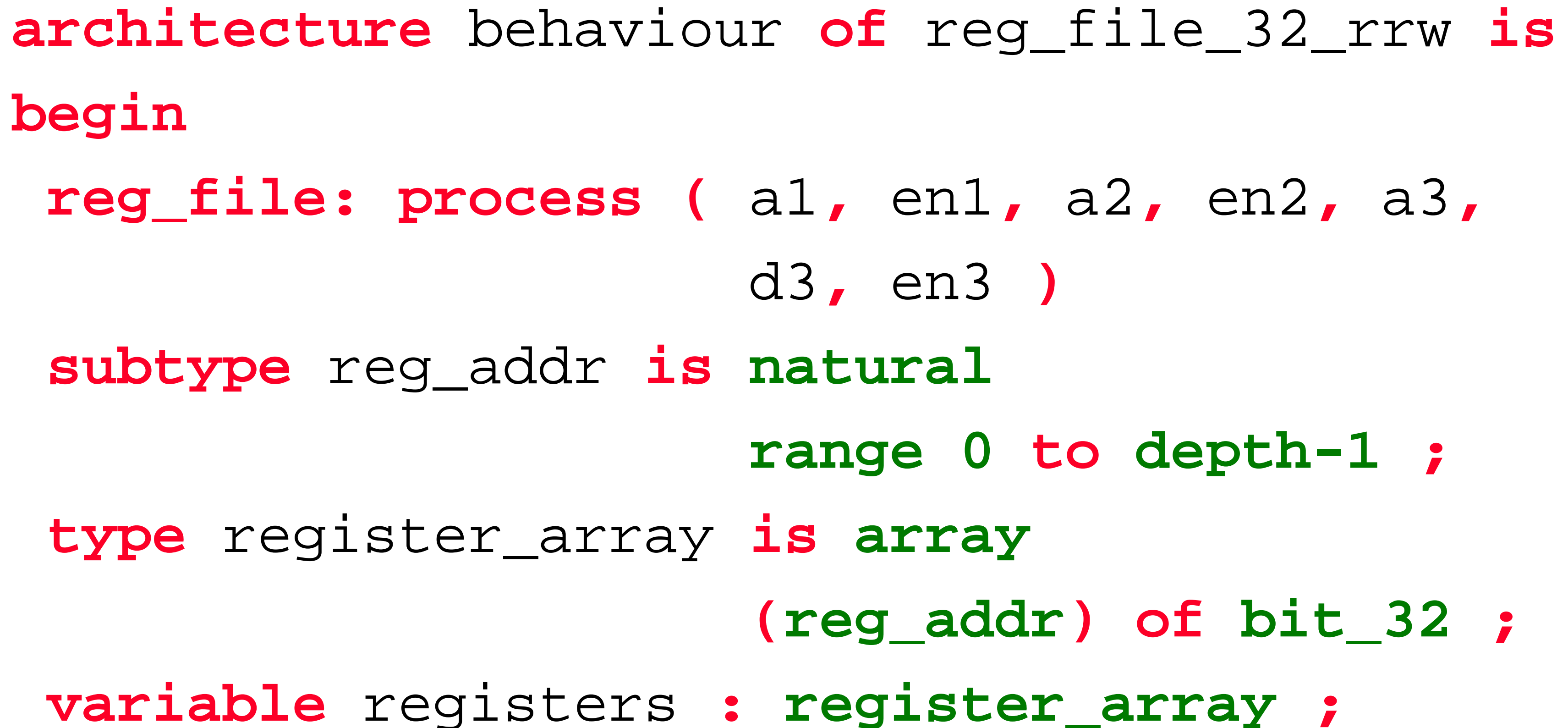
Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Register File

```
port ( a1   : in bit_vector(depth-1 downto 0);
       q1   : out bus_bit_32 bus ;
       en1 : in bit ;
       a2   : in bit_vector(depth-1 downto 0);
       q2   : out bus_bit_32 bus ;
       en2 : in bit ;
       a3   : in bit_vector(depth-1 downto 0);
       d3   : in bit_32 ;
       en3 : in bit );
end reg_file_32_rrw ;
```

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Register File



```
architecture behaviour of reg_file_32_rrw is
begin
 reg_file: process ( a1, en1, a2, en2, a3,
                     d3, en3 )
 subtype reg_addr is natural
                     range 0 to depth-1 ;
 type register_array is array
                     (reg_addr) of bit_32 ;
 variable registers : register_array ;
```

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Register File

```
begin
 if en3 = '1' then
  registers(bits_to_natural(a3)) := d3 ;
 end if ;
 if en1 = '1' then
  q1 <= registers( bits_to_natural(a1) )
  after Tac ;
 else
  q1 <= null after Tpd ;
 end if;
```

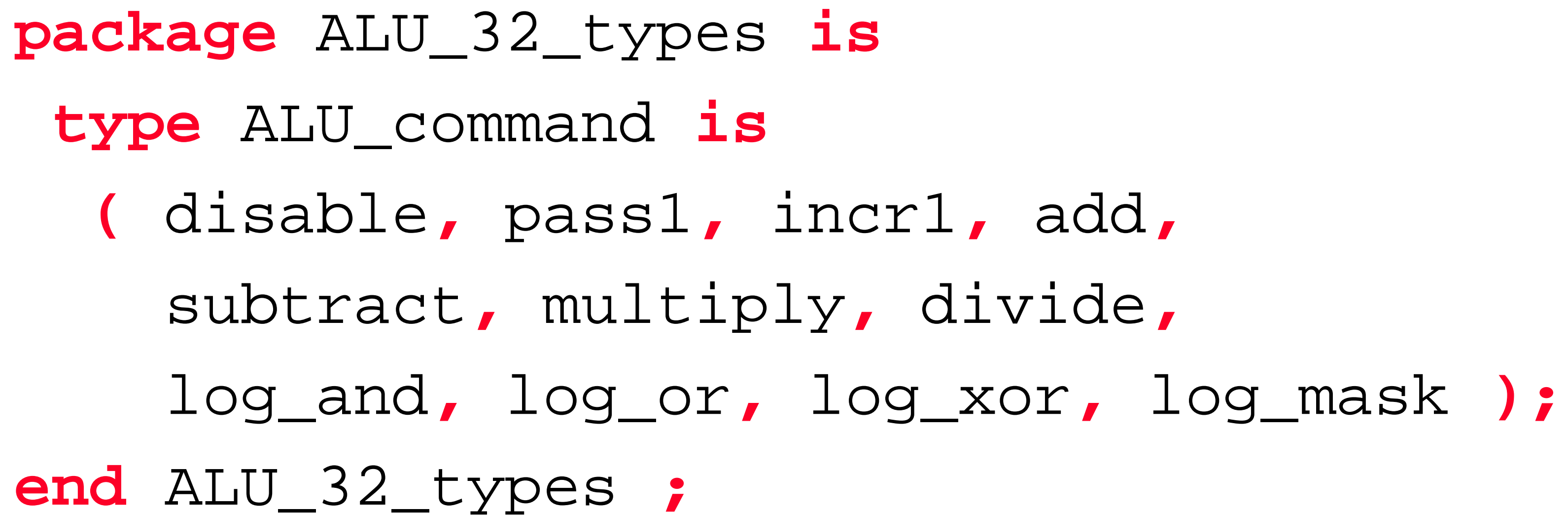
Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Register File

```
if en2 = '1' then
 q2 <= registers( bits_to_natural(a2) )
        after Tac ;
else
 q2 <= null after Tpd ;
end if ;
end process reg_file ;
end behaviour ;
```

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

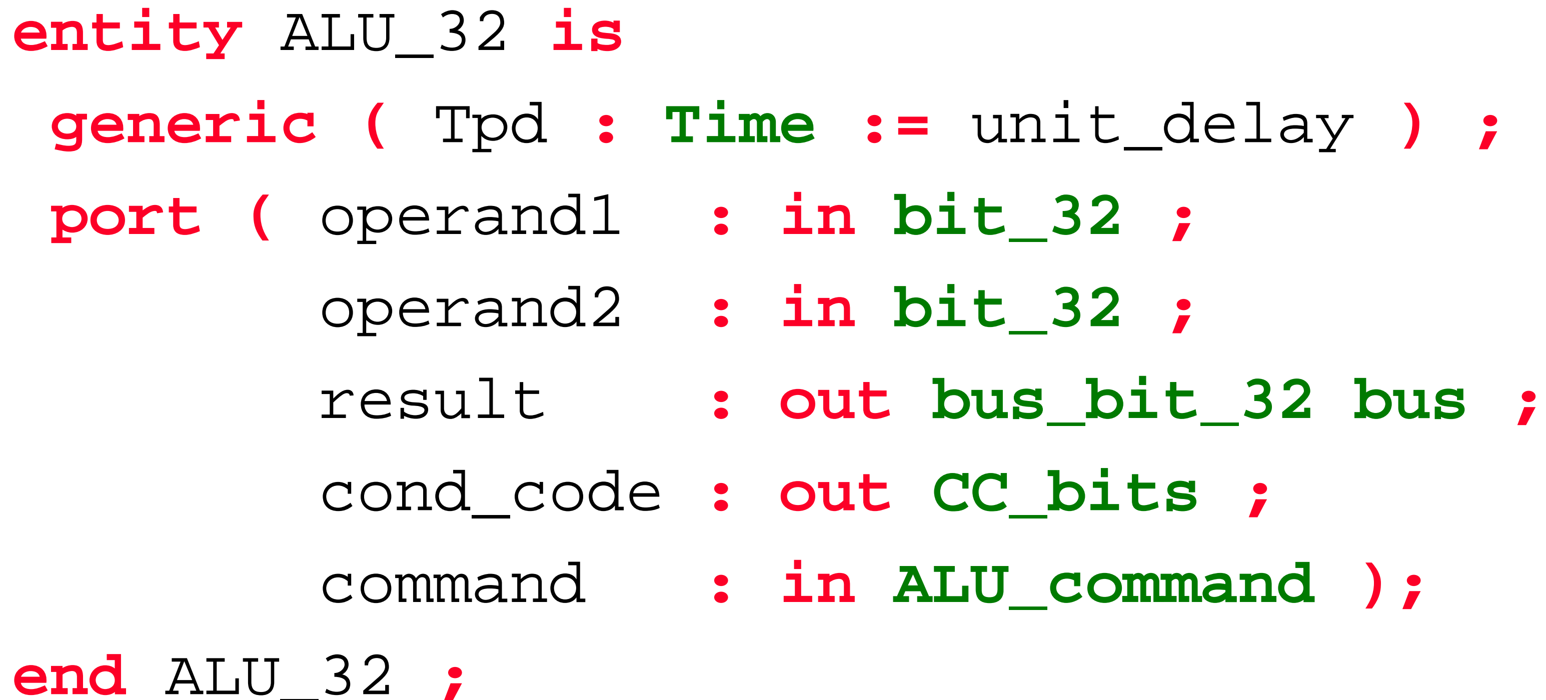
ALU



```
package ALU_32_types is
 type ALU_command is
  ( disable, pass1, incr1, add,
    subtract, multiply, divide,
    log_and, log_or, log_xor, log_mask );
end ALU_32_types ;
```

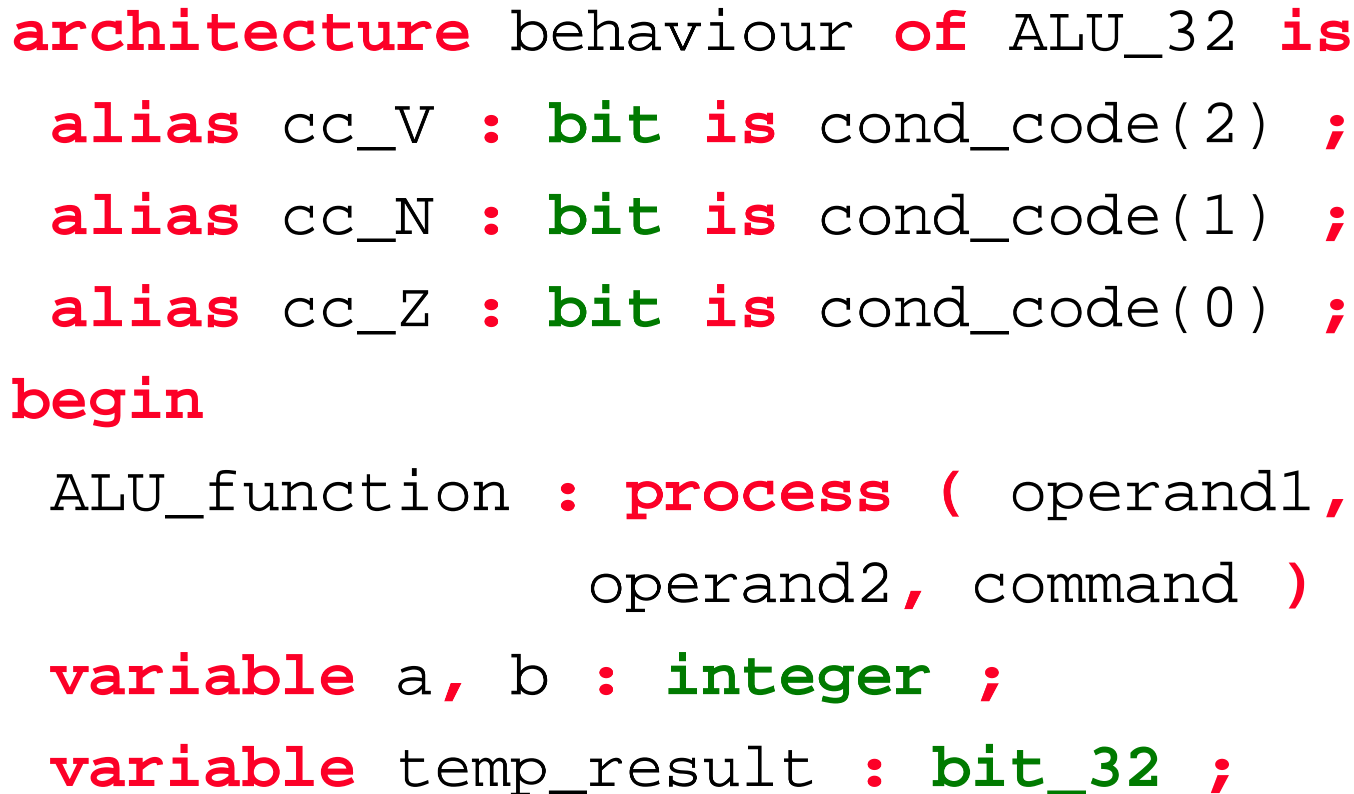
Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

ALU



Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

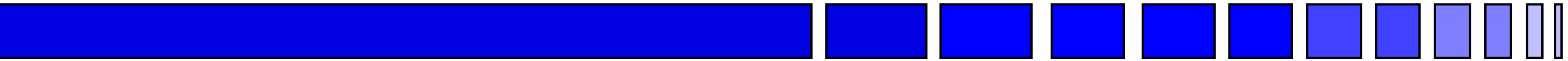
ALU



```
architecture behaviour of ALU_32 is
 alias cc_V : bit is cond_code(2) ;
 alias cc_N : bit is cond_code(1) ;
 alias cc_Z : bit is cond_code(0) ;
begin
 ALU_function : process ( operand1,
                  operand2, command )
 variable a, b : integer ;
 variable temp_result : bit_32 ;
```

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

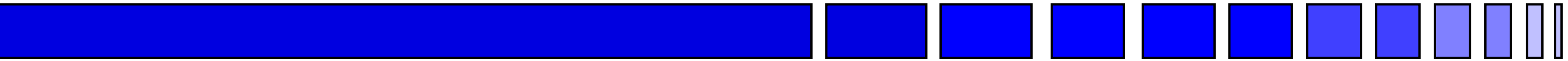
ALU



```
begin
case command is
 when add | subtract | multiply | divide =>
  a := bits_to_int(operand1) ;
  b := bits_to_int(operand2) ;
 when incr1 =>
  a := bits_to_int(operand1) ;
  b := 1 ;
 when others => null ;
end case;
```

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

ALU



```
case command is
  when disable => null ;
  when pass1    =>
      temp_result := operand1 ;
  when log_and =>
      temp_result := operand1 and operand2 ;
  when log_or   =>
      temp_result := operand1 or operand2 ;
```

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

ALU

```
when log_xor =>
  temp_result := operand1 xor operand2;
when log_mask =>
  temp_result := operand1 and not operand2;
when add | incr1 =>
  if b > 0 and a > integer'high - b then
-- positive overflow
  int_to_bits(((integer'low+a)+b)-integer'high-
     1, temp_result);
  cc_V <= '1' after Tpd;
```

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

ALU

```
elsif b > 0 and a < integer'low + b then
 elsif b < 0 and a < integer'low - b then
-- negative overflow
  int_to_bits( ( (integer'high+a) + b )-
           integer'low + 1, temp_result ) ;
  cc_V <= '1' after Tpd;
else
  int_to_bits( a + b, temp_result ) ;
  cc_V <= '0' after Tpd;
end if;
```

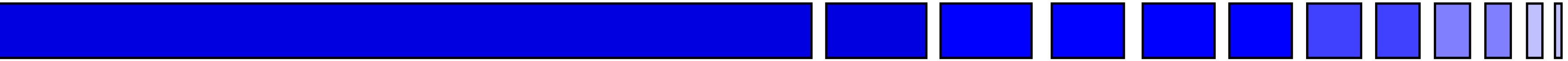
Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

ALU

```
when subtract =>
  if b < 0 and a > integer'high + b then
-- positive overflow
  int_to_bits( ( (integer'low + a) - b )-
                integer'high-1,
  temp_result) ;
  cc_V <= '1' after Tpd ;
```

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

ALU



```
-- negative overflow
  int_to_bits( ( (integer'high+a) - b )
                    - integer'low + 1 ,
                      temp_result) ;
  cc_V <= '1' after Tpd ;
else
  int_to_bits( a - b , temp_result) ;
  cc_V <= '0' after Tpd ;
end if;
```

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

ALU

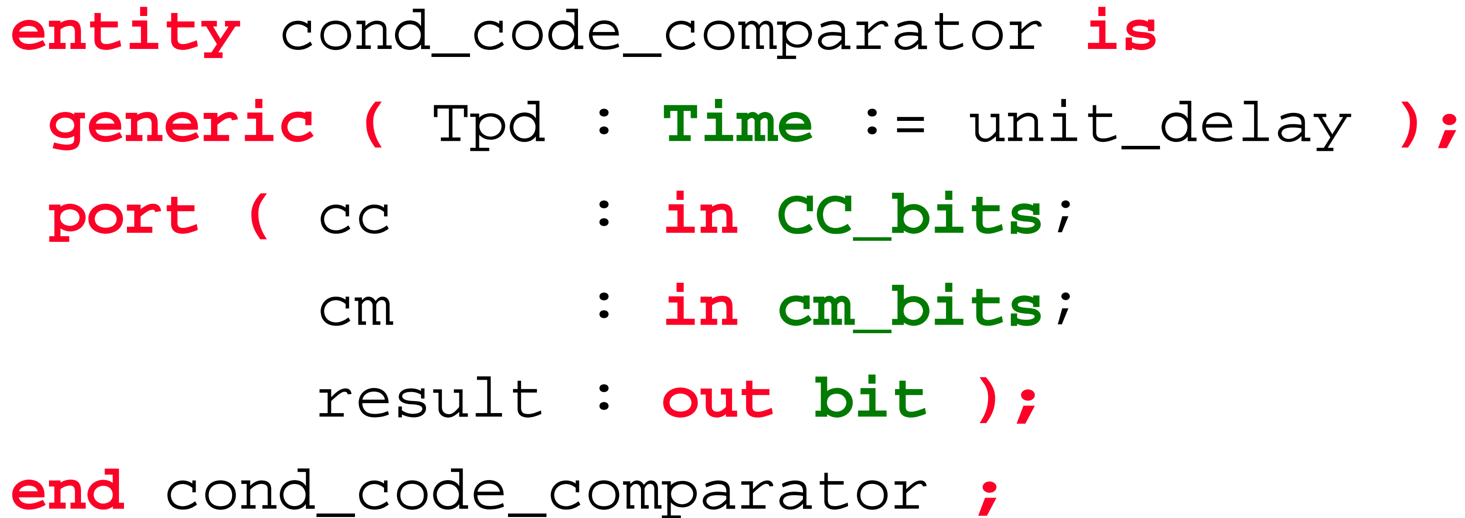
```
when multiply  => ...
when divide    => ...

  end process ALU_function;

end behaviour ;
```

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

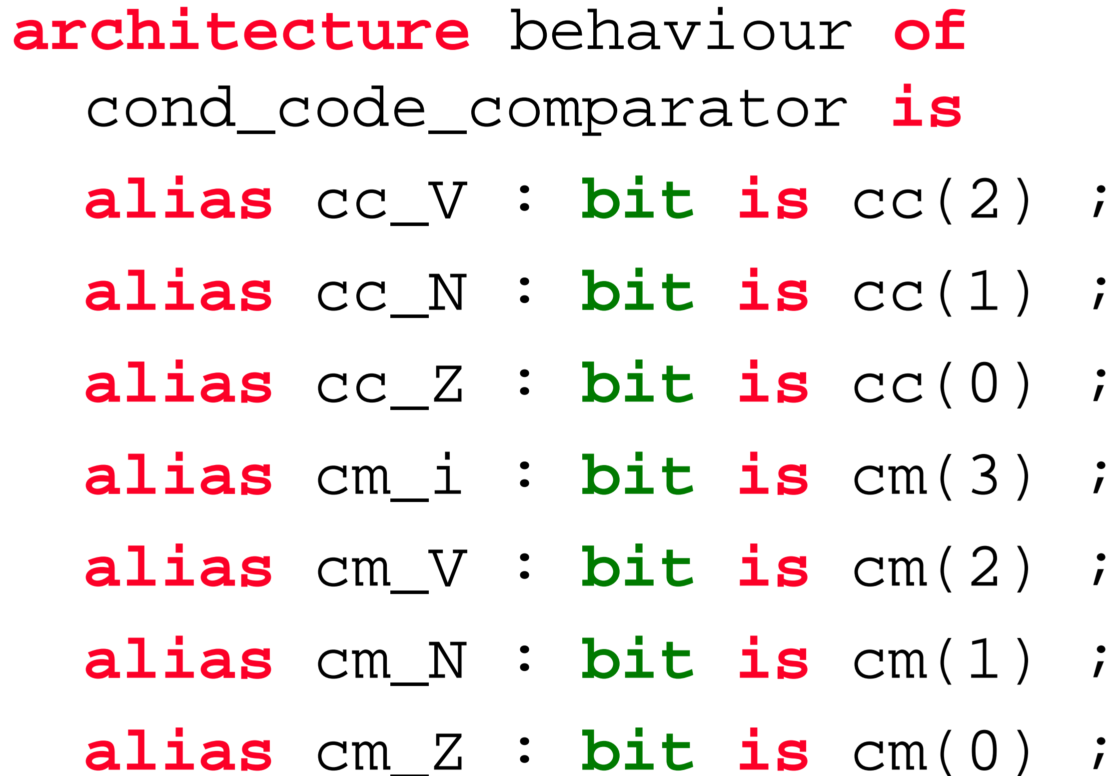
Condition Code Comparator



```
entity cond_code_comparator is
 generic ( Tpd : Time := unit_delay );
 port ( cc     : in CC_bits;
        cm     : in cm_bits;
        result : out bit );
end cond_code_comparator ;
```

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Condition Code Comparator



```
architecture behaviour of
  cond_code_comparator is
  alias cc_V : bit is cc(2) ;
  alias cc_N : bit is cc(1) ;
  alias cc_Z : bit is cc(0) ;
  alias cm_i : bit is cm(3) ;
  alias cm_V : bit is cm(2) ;
  alias cm_N : bit is cm(1) ;
  alias cm_Z : bit is cm(0) ;
```

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Condition Code Comparator

```
begin
 result <= bool_to_bit(
                                          (
    (cm_V and cc_V)
or  (cm_N and cc_N)
or  (cm_Z and cc_Z)
                                          ) = cm_i
                                        ) after Tpd;
end behaviour ;
```

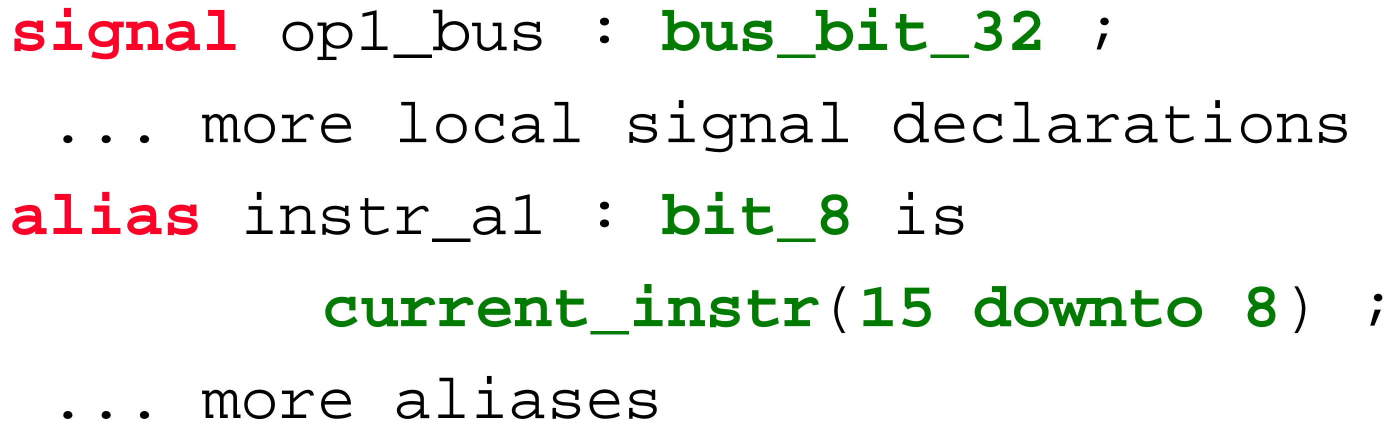
Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Condition Code Comparator

```
architecture RTL of dp32 is
  component reg_file_32_rrw
     generic ( ... ) ;
     port    ( ... ) ;
  end component ;
  ...rest of component declarations
```

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

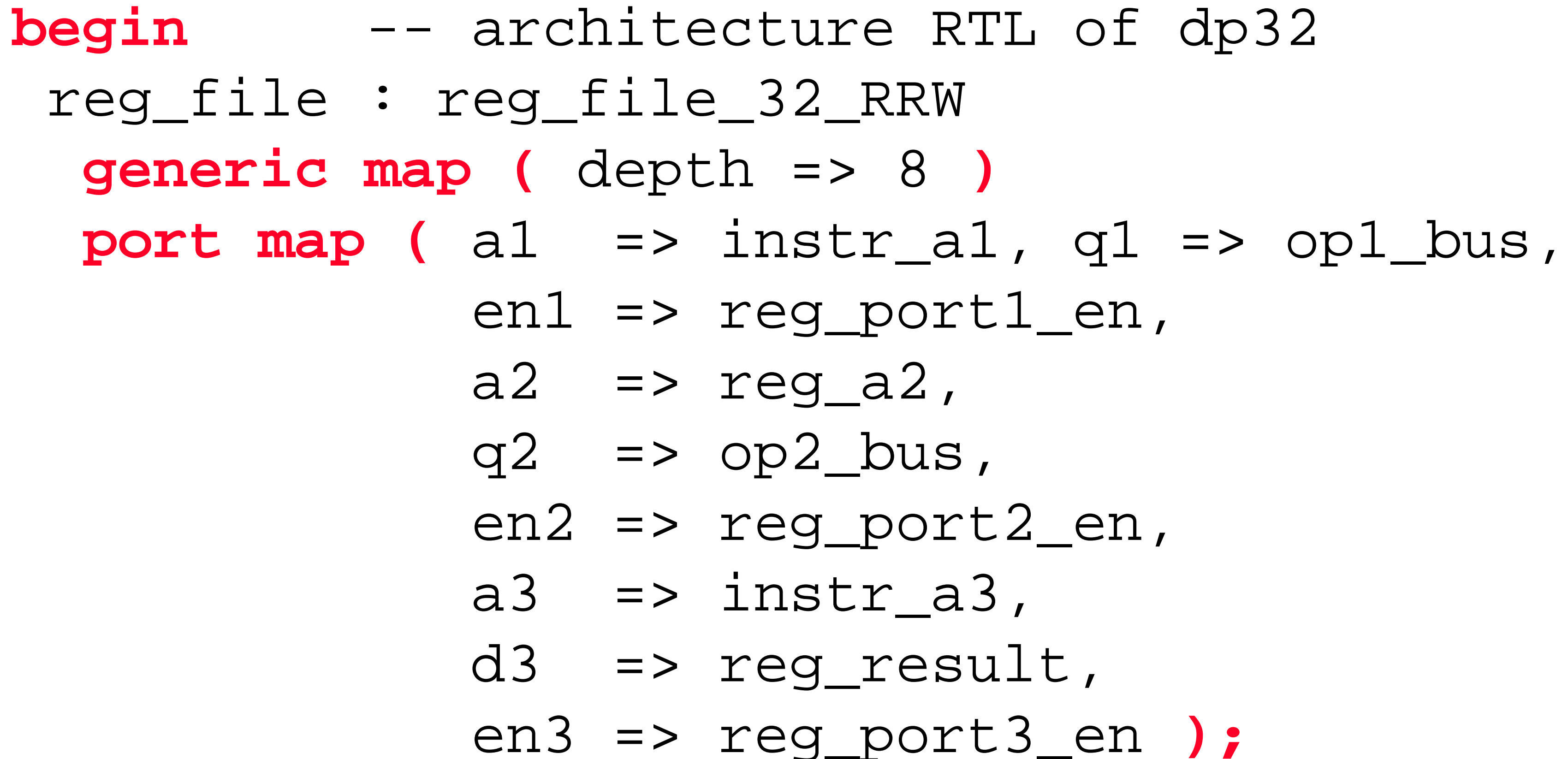
Condition Code Comparator



```
signal op1_bus : bus_bit_32 ;
 ... more local signal declarations
alias instr_a1 : bit_8 is
         current_instr(15 downto 8) ;
 ... more aliases
```

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Condition Code Comparator



```
begin      -- architecture RTL of dp32
 reg_file : reg_file_32_RRW
  generic map ( depth => 8 )
  port map ( a1  => instr_a1, q1 => op1_bus,
             en1 => reg_port1_en,
             a2  => reg_a2,
             q2  => op2_bus,
             en2 => reg_port2_en,
             a3  => instr_a3,
             d3  => reg_result,
             en3 => reg_port3_en );
```

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Condition Code Comparator

```
reg_port2_mux : mux2
     generic map ( ... )
     port map    ( ... );
```

```
... more instantiations of components
  with port maps...
```

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Controller

```
begin-- block controller
 state_machine: process
  type controller_state is
  ( resetting, fetch_0, fetch_1, fetch_2,
   decode, disp_fetch_0, disp_fetch_1,
   disp_fetch_2, execute_0, execute_1,
   execute_2 );
variable state, next_state : controller_state ;
variable write_back_pending : boolean ;
```

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Controller

```
type ALU_op_select_table is
 array (natural range 0 to 255) of ALU_command;
constant ALU_op_select : ALU_op_select_table :=
  ( 16#00# => add,        16#01# => subtract,
    16#02# => multiply,   16#03# => divide,
    16#10# => add,        16#11# => subtract,
    16#12# => multiply,   16#13# => divide,
    16#04# => log_and,    16#05# => log_or,
    16#06# => log_xor,    16#07# => log_mask,
    others => disable );
```

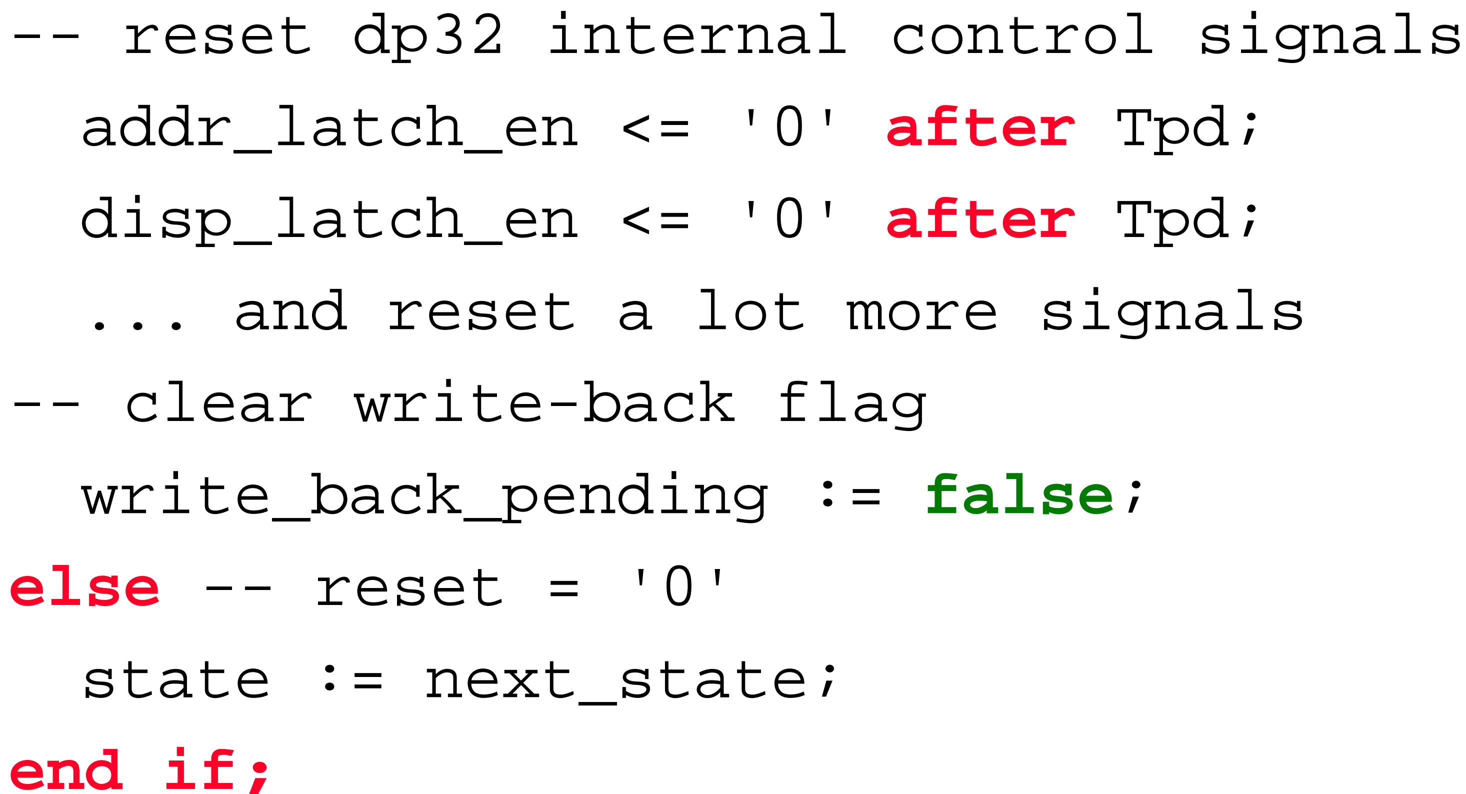
Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Controller

```
begin-- process state_machine
-- start of clock cycle
  wait until phi1 = '1';
-- check for reset
  if reset = '1' then
     state := resetting;
-- reset external bus signals
  read  <= '0' after Tpd;
  fetch <= '0' after Tpd;
  write <= '0' after Tpd;
```

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Controller



```
-- reset dp32 internal control signals
  addr_latch_en <= '0' after Tpd;
  disp_latch_en <= '0' after Tpd;
  ... and reset a lot more signals
-- clear write-back flag
  write_back_pending := false;
else -- reset = '0'
  state := next_state;
end if;
```

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Controller

```
-- dispatch action for current state
case state is
 when resetting =>
--ck for reset going inactive at end of clock
 wait until phi2 = '0' ;
 if reset = '0' then
  next_state := fetch_0;
 else
  next_state := resetting;
 end if;
```

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Controller

```
 when fetch_0 =>
-- clean up after previous execute cycles
    reg_port1_en <= '0' after Tpd;
  ... reset a lot more signals
-- handle pending register write-back
 if write_back_pending then
    reg_port3_en <= '1' after Tpd;
 end if;
```


Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Controller

```
-- enable PC via ALU to address latch
  PC_out_en <= '1' after Tpd;
-- enable PC onto op1_bus
  ALU_op     <= pass1 after Tpd;
-- pass PC to r_bus
  wait until phi2 = '1';
  addr_latch_en <= '1' after Tpd
-- latch instr address
  wait until phi2 = '0';
  addr_latch_en <= '0' after Tpd;
  next_state := fetch_1;
```

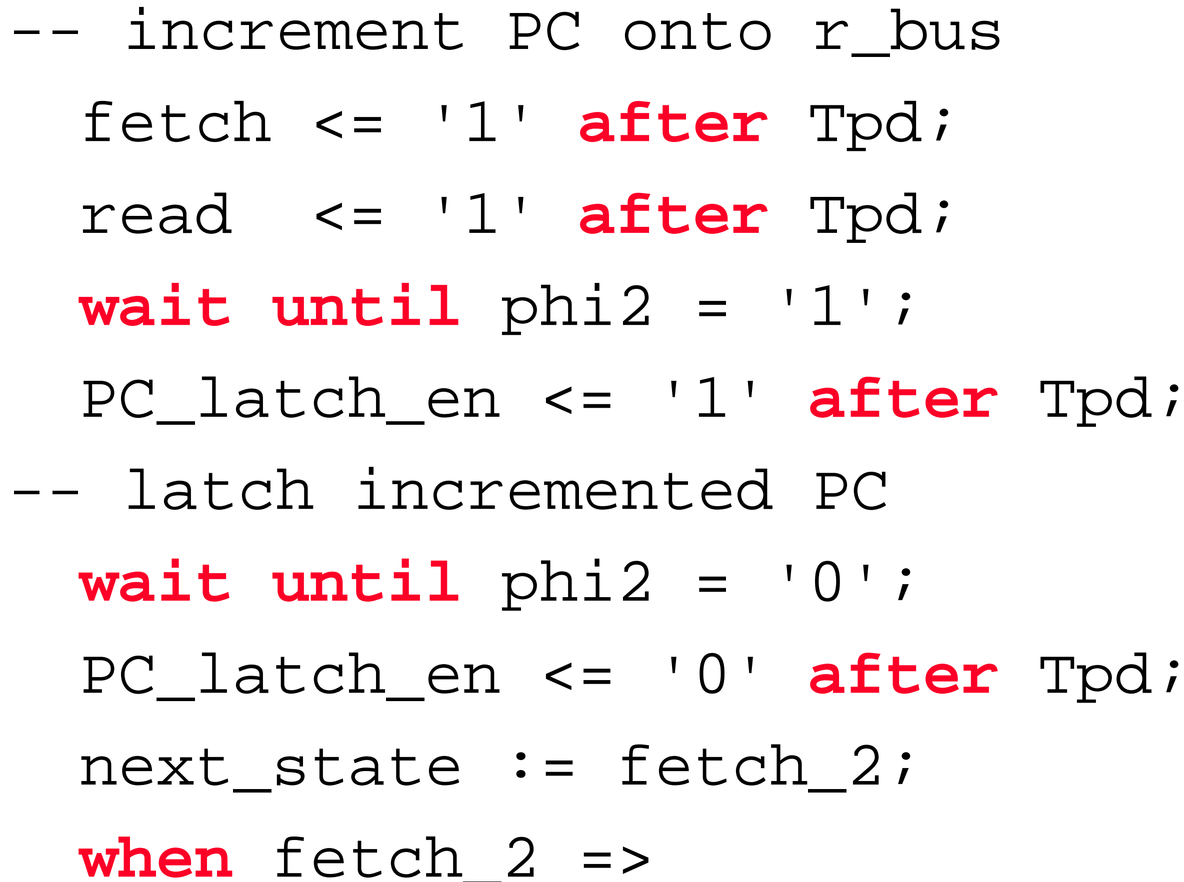
Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Controller

```
when fetch_1 =>
-- clear pending register write-back
 if write_back_pending then
 reg_port3_en <= '0' after Tpd;
 write_back_pending := false;
 end if;
-- increment PC & start bus read
 ALU_op <= incr1 after Tpd;
```

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

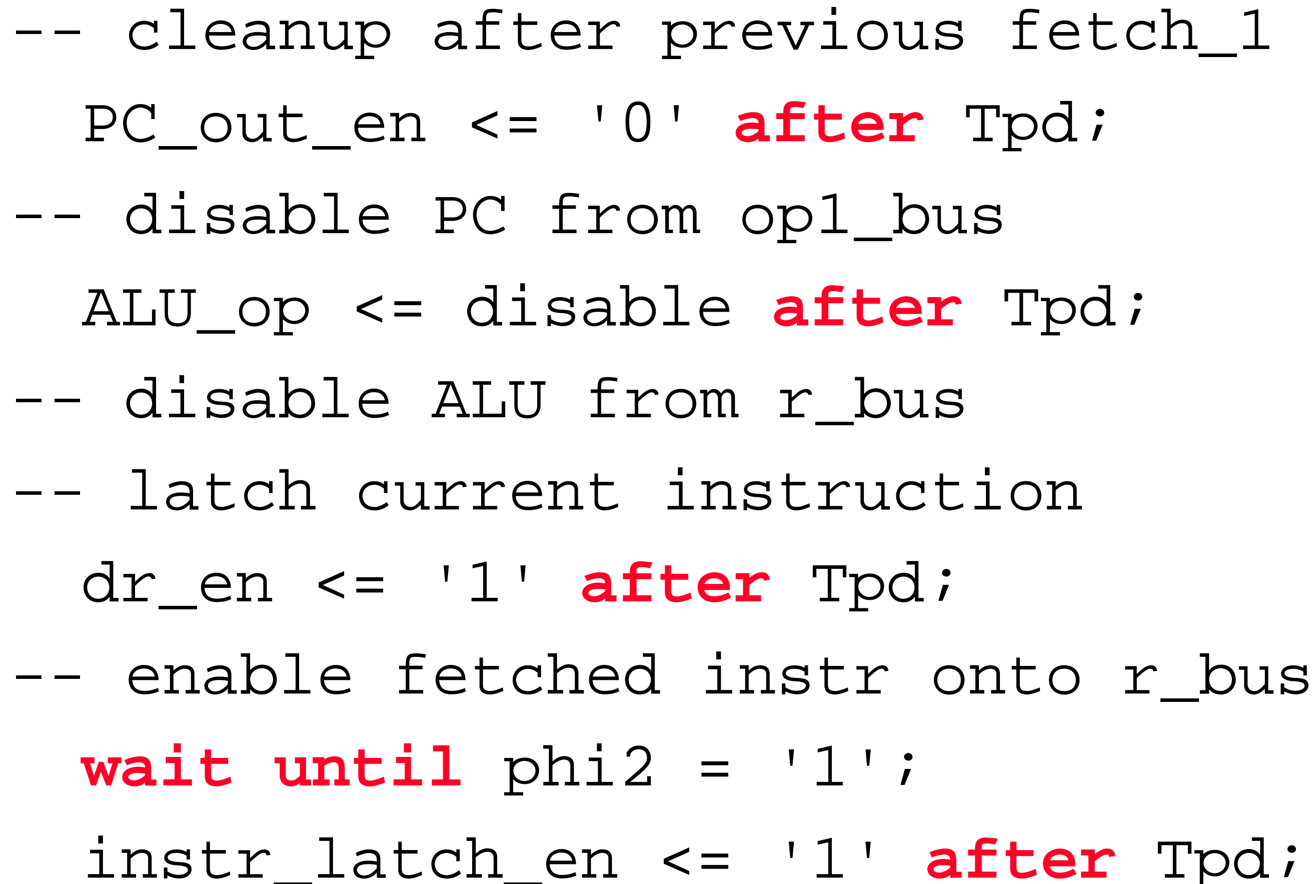
Controller



```
-- increment PC onto r_bus
  fetch <= '1' after Tpd;
  read  <= '1' after Tpd;
  wait until phi2 = '1';
  PC_latch_en <= '1' after Tpd;
-- latch incremented PC
  wait until phi2 = '0';
  PC_latch_en <= '0' after Tpd;
  next_state := fetch_2;
  when fetch_2 =>
```

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

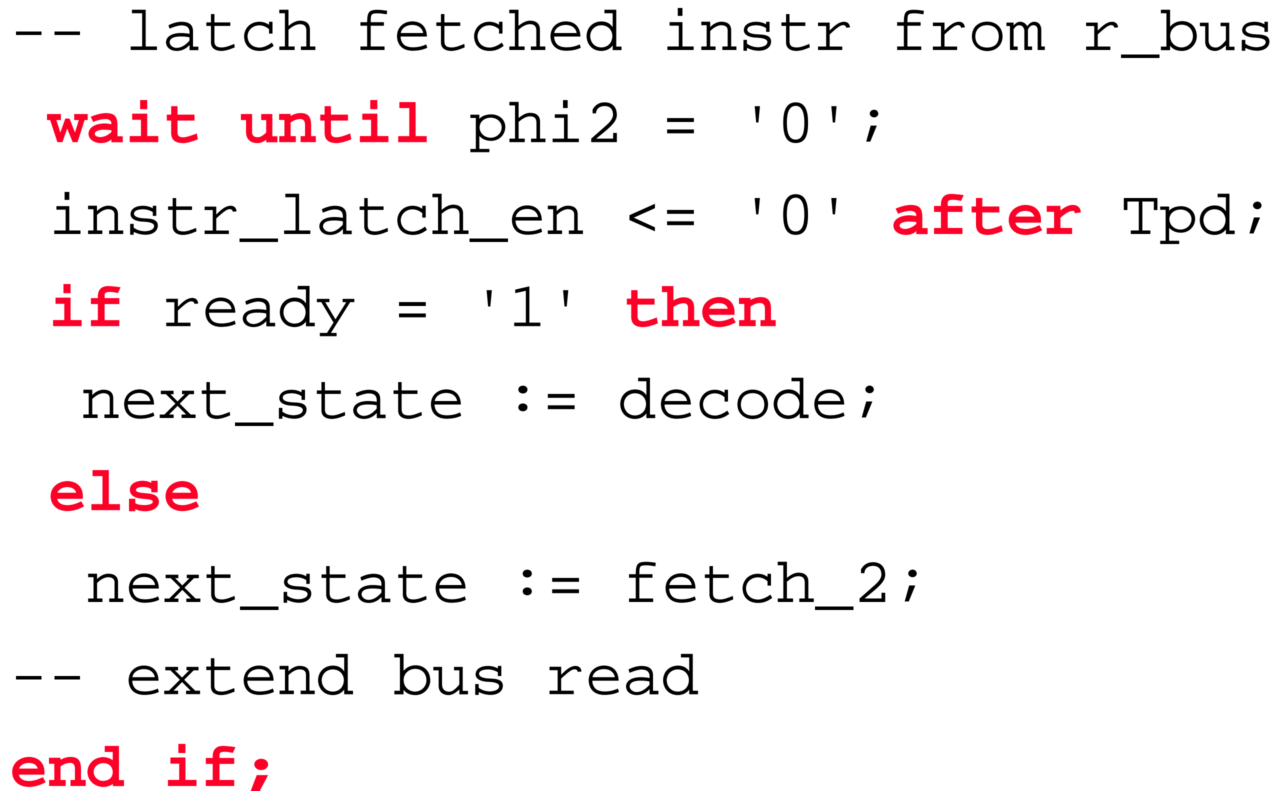
Controller



```
-- cleanup after previous fetch_1
  PC_out_en <= '0' after Tpd;
-- disable PC from op1_bus
  ALU_op <= disable after Tpd;
-- disable ALU from r_bus
-- latch current instruction
  dr_en <= '1' after Tpd;
-- enable fetched instr onto r_bus
  wait until phi2 = '1';
  instr_latch_en <= '1' after Tpd;
```

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Controller



Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Controller

```
when decode =>
-- terminate bus read from previous fetch_2
  ...
-- disable fetched instr from r_bus
-- delay to allow decode logic to settle
  ...
-- next state based on opcode of currect
  instruction
```

Controller

```
case opcode is
 when op_add | op_sub | op_mul | op_div
       | op_addq | op_subq | op_mulq
       |op_divq | op_land | op_lor
       |op_lxor | op_lmask
       | op_ldq | op_stq =>
   next_state := execute_0;
     ...
```

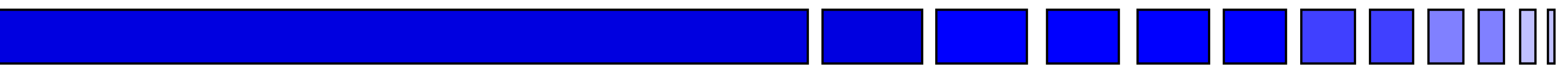
Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

Controller

```
-- fetch offset
  ...
-- if branch taken fetch displacement
  ...
-- else
  ...
-- if branch taken add immed displacement to PC
  ...
-- ignore and carry on
end case; -- op
```

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

RTL Testbench



```
configuration dp32_rtl_test of dp32_test is
 for structure
  for cg : clock_gen
   use entity work.clock_gen(behaviour)
   generic map( Tpw => 8 ns, Tps => 2ns);
  end for;
  for mem : memory
   use entity work.memory(behaviour);
  end for;
```

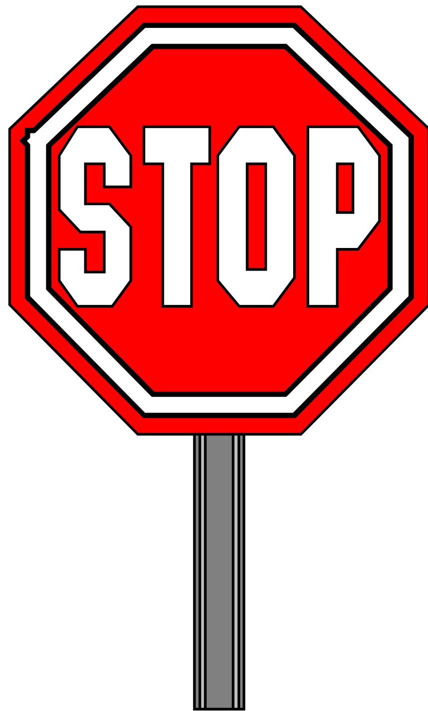
Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

RTL Testbench

```
for proc : dp32
   use entity work.dp32(rtl);
  for rtl
   for all : reg_file_32_rrw
    use entity
     work.reg_file_32_rrw(behaviour);
   end for;
     ... more "for"
  end for;
end dp32_rtl_test;
```

Copyright © 1997/8/9, KJH, 545_12, 5/8/2001

End of Lecture



Copyright © 1997/8/9, KJH, 545_12, 5/8/2001