LEFT SHIFT (MULTIPLY BY 4)

Verilog

```
module sl2(input  [31:0] a,
           output [31:0] y);

  // shift left by 2
  assign y = {a[29:0], 2'b00};
endmodule
```

VHDL

```
library IEEE; use IEEE.STD_LOGIC_1164.all;
entity sl2 is -- shift left by 2
  port(a: in  STD_LOGIC_VECTOR(31 downto 0);
       y: out STD_LOGIC_VECTOR(31 downto 0));
end;

architecture behave of sl2 is
begin
  y <= a(29 downto 0) & "00";
end;
```

SIGN EXTENSION

Verilog

```
module signext(input  [15:0] a,
               output [31:0] y);

  assign y = {{16{a[15]}}, a};
endmodule
```

VHDL

```
library IEEE; use IEEE.STD_LOGIC_1164.all;
entity signext is -- sign extender
port(a: in  STD_LOGIC_VECTOR(15 downto 0);
     y: out STD_LOGIC_VECTOR(31 downto 0));
end;

architecture behave of signext is
begin
  y <= X"0000" & a when a(15) = '0' else X"ffff" & a;
end;
```

RESETTABLE FLIP-FLOP

Verilog

```
module flopr #(parameter WIDTH = 8)
              (input                  clk, reset,
               input      [WIDTH-1:0] d,
               output reg [WIDTH-1:0] q);

  always @(posedge clk, posedge reset)
    if(reset) q <= 0;
    else      q <= d;
endmodule
```

VHDL

```
library IEEE; use IEEE.STD_LOGIC_1164.all; use
IEEE.STD_LOGIC_ARITH.all;
entity flopr is -- flip-flop with synchronous reset
  generic(width: integer);
  port(clk, reset: in   STD_LOGIC;
       d:          in   STD_LOGIC_VECTOR(width-1 downto 0);
       q:          out  STD_LOGIC_VECTOR(width-1 downto 0));
end;

architecture asynchronous of flopr is
begin
  process(clk, reset) begin
    if reset = '1' then q <= CONV_STD_LOGIC_VECTOR(0, width);
    elsif clk'event and clk = '1' then
      q <= d;
    end if;
  end process;
end;
```