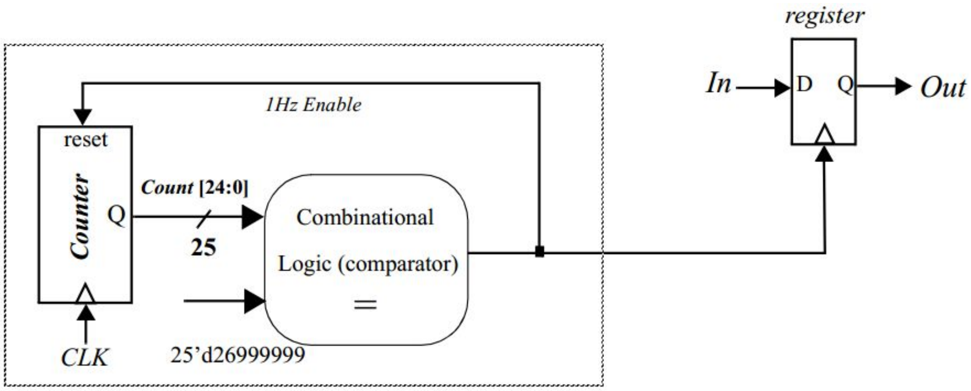
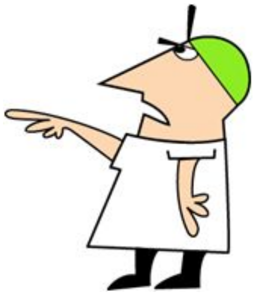
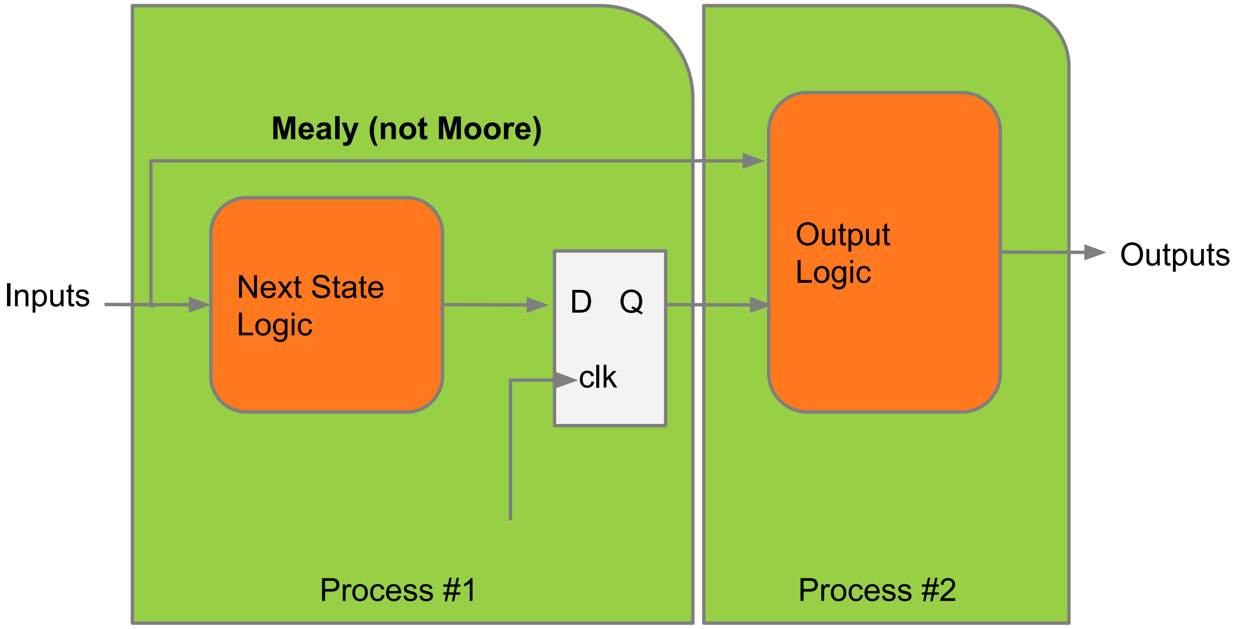
TIPs: Digital glitches!



- A 1 Hz signal is derived from a 27MHz to update a register
- Assume the counter reset is synchronous, a global reset (cnt=0) is not shown
- What can be wrong?



TIPs: State Machines



Mealy (not Moore)

Inputs

Next State Logic

D Q

clk

Output Logic

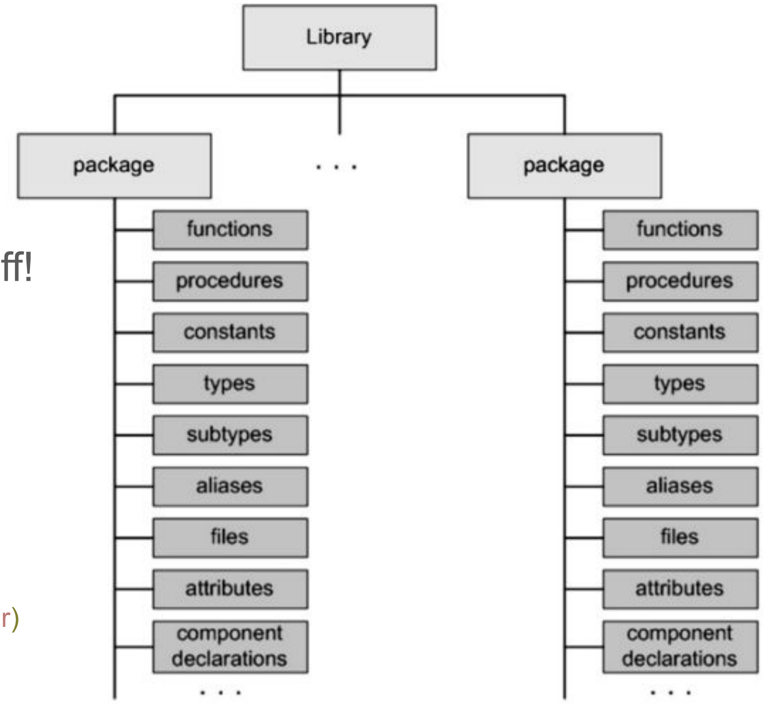
Outputs

Process #1

Process #2

VHDL Libraries and Packages

- Code reusability and Modularity
- Best Practices / Shorthands
- Investing in Packages really pays off!



Examples:

```
type ArrOfSTDV8_t is array (natural range <>) of
  std_logic_vector(7 downto 0);

function ilog2(val : integer) return integer ;

function terni(cond : boolean; res_true, res_false : integer)
  return integer;
```

VHDL Records

```
example_inst : axi_example_peripheral
    port_map(
        clk               => clk,
        data              => data,
        s_axi_gp0_aclk    => aclk,
        s_axi_gp0_arvalid => periph_axi_arvalid,
        s_axi_gp0_awvalid => periph_axi_awvalid,
        s_axi_gp0_bready  => periph_axi_bready,
        s_axi_gp0_rready  => periph_axi_rready,
        s_axi_gp0_wlast   => periph_axi_wlast,
        s_axi_gp0_wvalid  => periph_axi_wvalid,
        s_axi_gp0_arid    => periph_axi_arid,
        s_axi_gp0_awid    => periph_axi_awid,
        s_axi_gp0_wid     => periph_axi_wid,
        s_axi_gp0_arburst => periph_axi_arburst,
        s_axi_gp0_arlock  => periph_axi_arlock,
        s_axi_gp0_arsize  => periph_axi_arsize,
        ... MANY MANY LINES REMOVED!
      );
```

```
example_inst : axi_example_peripheral
    port_map(
        clk            => clk,
        data           => data,
        s_axi_gp0_aclk => aclk,
        s_axi_gp0_in   => periph_axi.from_master,
        s_axi_gp0_out  => periph_axi.to_master
    );
```

VHDL Records

```
library ieee;
use ieee.std_logic_1164.all;
package my_package is
   type from_AXI_master is record
        arvalid : std_logic;
        awvalid : std_logic;
         … removed!
    end record from_AXI_master;
    type to_AXI_master is record
        .. removed!
        rresp : std_logic_vector(1 downto 0);
        rdata : std_logic_vector(31 downto 0);
    end record to_AXI_master;
    type AXI_slave is record
        to_master : to_AXI_master;
        from_master : from_AXI_master;
    end record AXI_slave;
end my_package;
```

Packages can also include:

- Constants
- Types
- Common components
- Functions
- Procedures

Generate vs Loop!

Generate:

1. Replicating Logic in VHDL
2. Turning on/off blocks of logic in VHDL

```
GEN_ADD: for I in 0 to 7 generate

   LOWER_BIT: if I=0 generate
     U0: HALFADD port map  (A(I),B(I),S(I),C(I));
   end generate LOWER_BIT;

   UPPER_BITS: if I>0 generate
     UX: FULLADD port map (A(I),B(I),C(I-1),S(I),C(I));
   end generate UPPER_BITS;

 END generate GEN_ADD;

 COUT <= C(7);
```

Loop:

1. Must be in a process
2. Executes in a single clock (take care!)

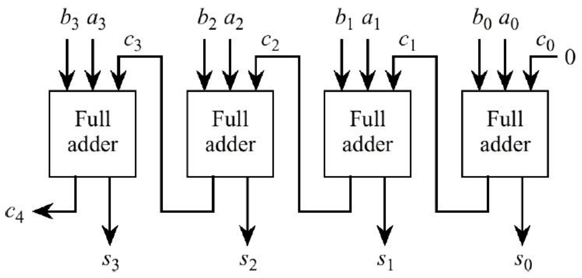
```
 process (A)
begin
  Z <= "0000";
  for i in 0 to 3 loop
    if (A = i) then
      Z(i) <= ‘1’;
     end if;
  end loop;
 end process;
```

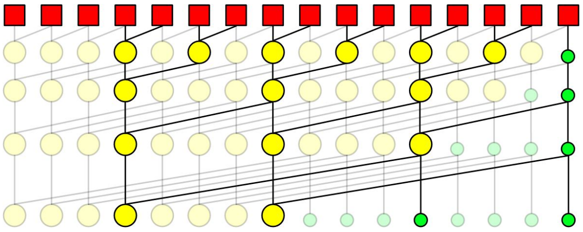
```
for i in mySrc’range loop
   my2Ddest(i) <= mySrc(i);
end loop;
```

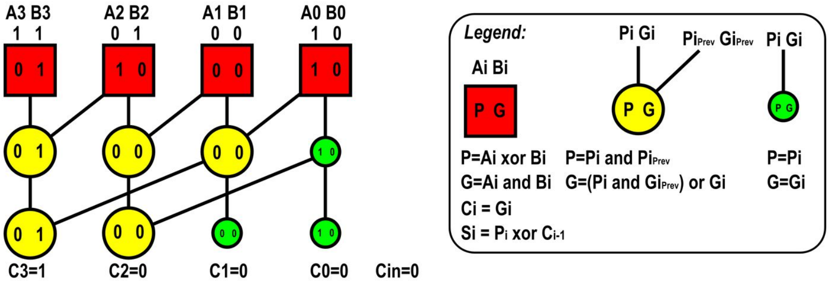
✓ ✗

TIPs: Simple things are not simple

- Ripple Adder: Delay is *O*(n)
- Kogge-Stone: Delay is *O*(Log_2(n))
  - ○ Area? Power?
  - ○ Mid-way trees?

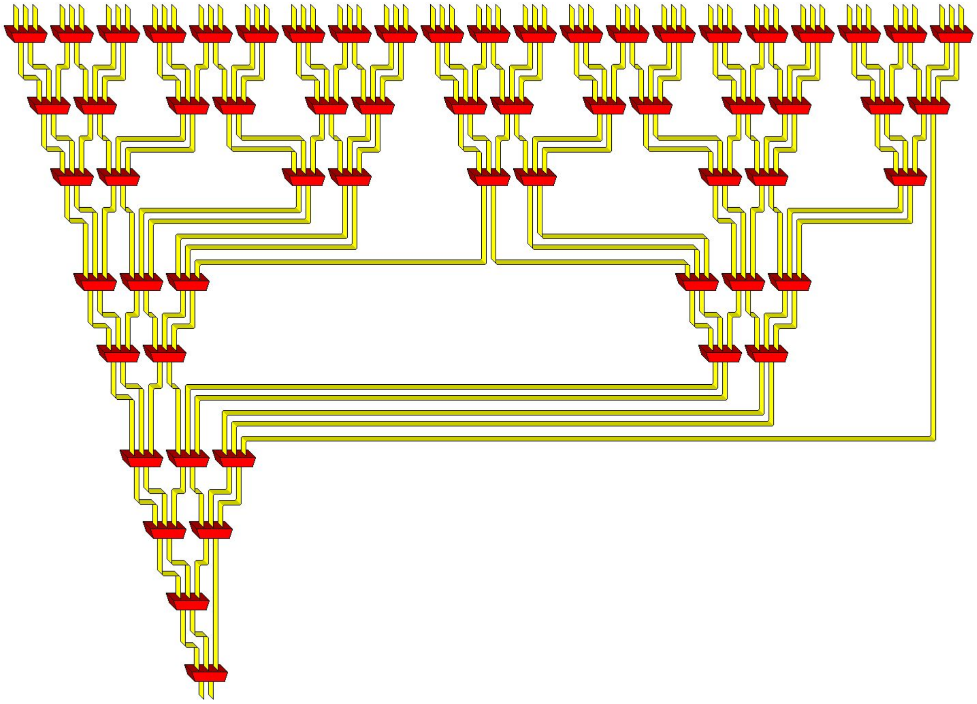






TIPs: Simple things are not simple

- Multiplication is easier ;-)
- Needs a CPA
- Half-width? Rounding?
- Fused Multiply-Add



TIPs: Simple things are not simple

Newton-Raphson: Division A/B

1. Look-up an estimate for the reciprocal of the divisor: X_0 [$\simeq$1/B]
2. Compute successively more accurate estimates of the reciprocal: X_i
3. Compute the quotient by multiplying the dividend by the reciprocal of the divisor Q= A x (1/B)

$$X_{i+1} = X_i - \frac{f(X_i)}{f'(X_i)} = X_i - \frac{1/X_i - D}{-1/X_i^2} = X_i + X_i(1 - DX_i) = X_i(2 - DX_i)$$

Newton-Raphson: Square-root

$$x_{i+1} = x_i(3 - bx_i^2)/2$$

Lab #1

1. Write a vhdl module that adds 4 x 18bit registered inputs, and deliver the result at the clock positive edge
2. Write a TB to check results
3. Analyse
   a. RTL Schematics
   b. Technology Schematics
   c. Post PAR static timing
4. Add pipeline stages, calculate on 2 steps
5. Do steps 2~3
6. ~~Add 3 shift registers at output~~
7. ~~Do steps 2~3~~
8. ~~Enable register balancing~~
9. ~~Do steps 2~3~~