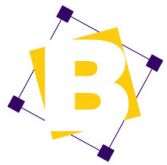
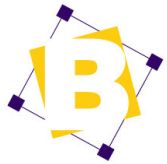


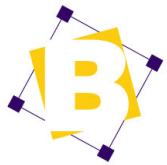
Clocking Domains

Clocking Domain and Reactive Region



- Clocking Domain important part of use of reactive region
- Failure to use clocking domain leads to pre vs post synthesis differences
- How do you get to reactive region?
  - Event/delay control inside special construct (program)
- How easy is it to accidentally use non-clocked version of signals?
  - Most common place for clocking domain is in program
  - Both clocking signal and regular signal available
  - Non-clocking signal **easiest** to use

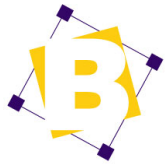
Syntax 3 – Forget “c.”




```
initial begin
  ##0; // start in reactive
  c.z <= c.x;
  @(myclking);
  c.z <= y;
  @(myclking);
  c.z = c.x;
  …
end
```

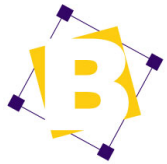
Ooops, I forgot to include clocking domain “c.” when sampling signal!

Clocking Domain and Extensibility / Reusability



- 24 Port Gigabit Ethernet
- Problem
  - Need 24 clocking domains – one for each port
- Current Solution
  - Put each port's clocking domain in interface
  - Instantiate clocking domain in interface
    - Have to use interface.clocking.signal – yuck!
    - I already disliked clocking.signal!
  - If you bury program in interface, you get same error prone problem (illustrated on previous slide)
- Proper Solution
  - Change syntax so clocking domain is just a view of signals in an interface
    - Only have to type interface.signal
      - **`Still would like enhancement in 3.2 to fix this`**
    - Can't accidentally get non-clocked signal!

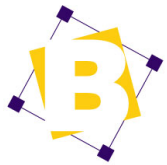
Current Syntax



Boyd Technology Inc.

```
module/program tb (
  logic req, gnt,
  logic [7:0] addr, data,
  logic [1:0] mode,
  logic start, rdy);

  clocking clk1 @(posedge clk);
    input #1step gnt, rdy, clk;
    output #2 req, addr, mode, start;
    inout data;
  endclocking



...

endmodule/endprogram
```

Proposed Syntax




```
interface simple_bus (
  logic req, gnt,
  logic [7:0] addr, data,
  logic [1:0] mode,
  logic start, rdy);

  modport tb @(posedge clk) (
    input #1step gnt, rdy, clk,
    output #2 req, addr, mode, start,
    inout data);
endinterface: simple_bus

module/program tb(simple_bus.tb bus);

...

endmodule/endprogram
```

Can't accidentally get version that didn't go through clocking