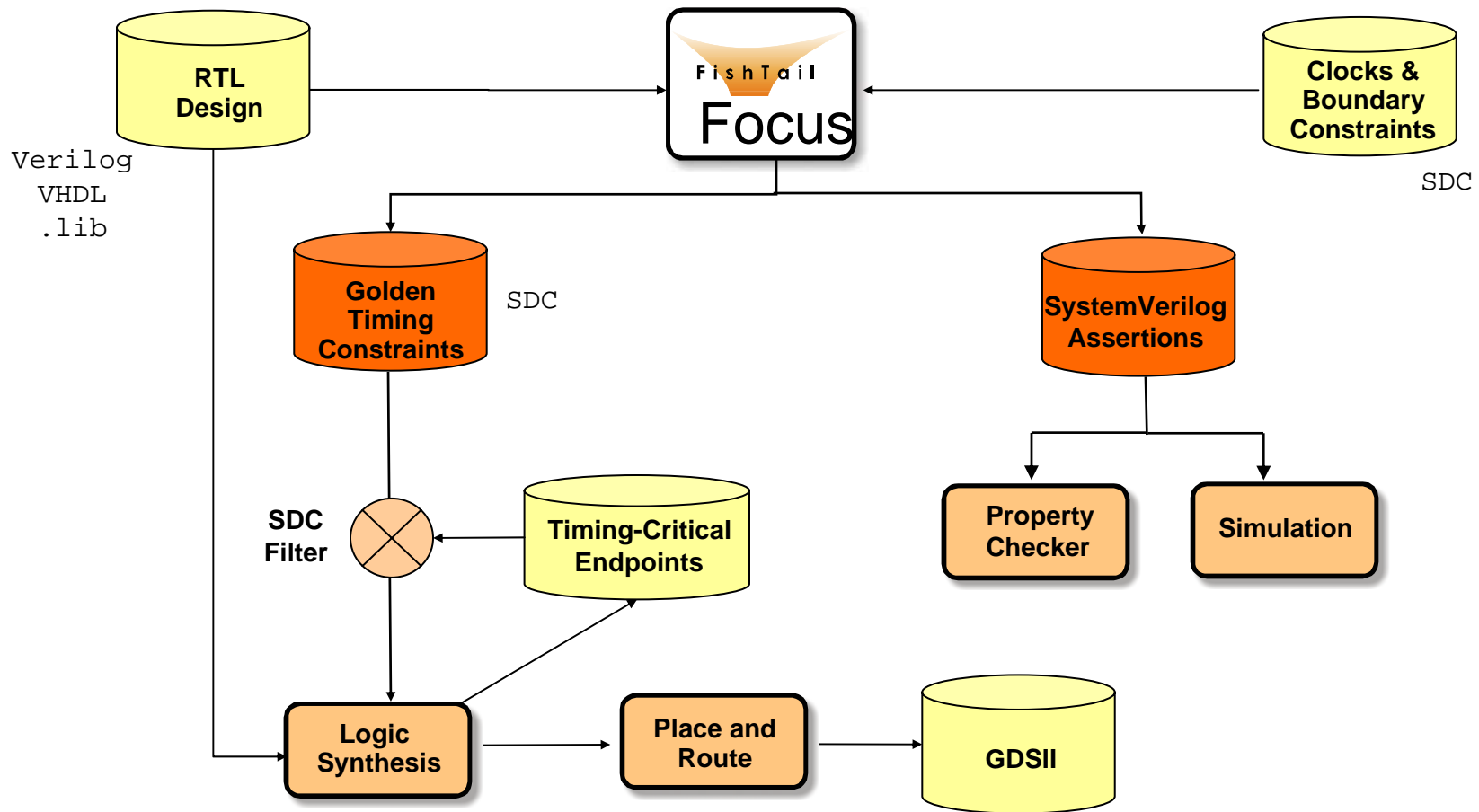


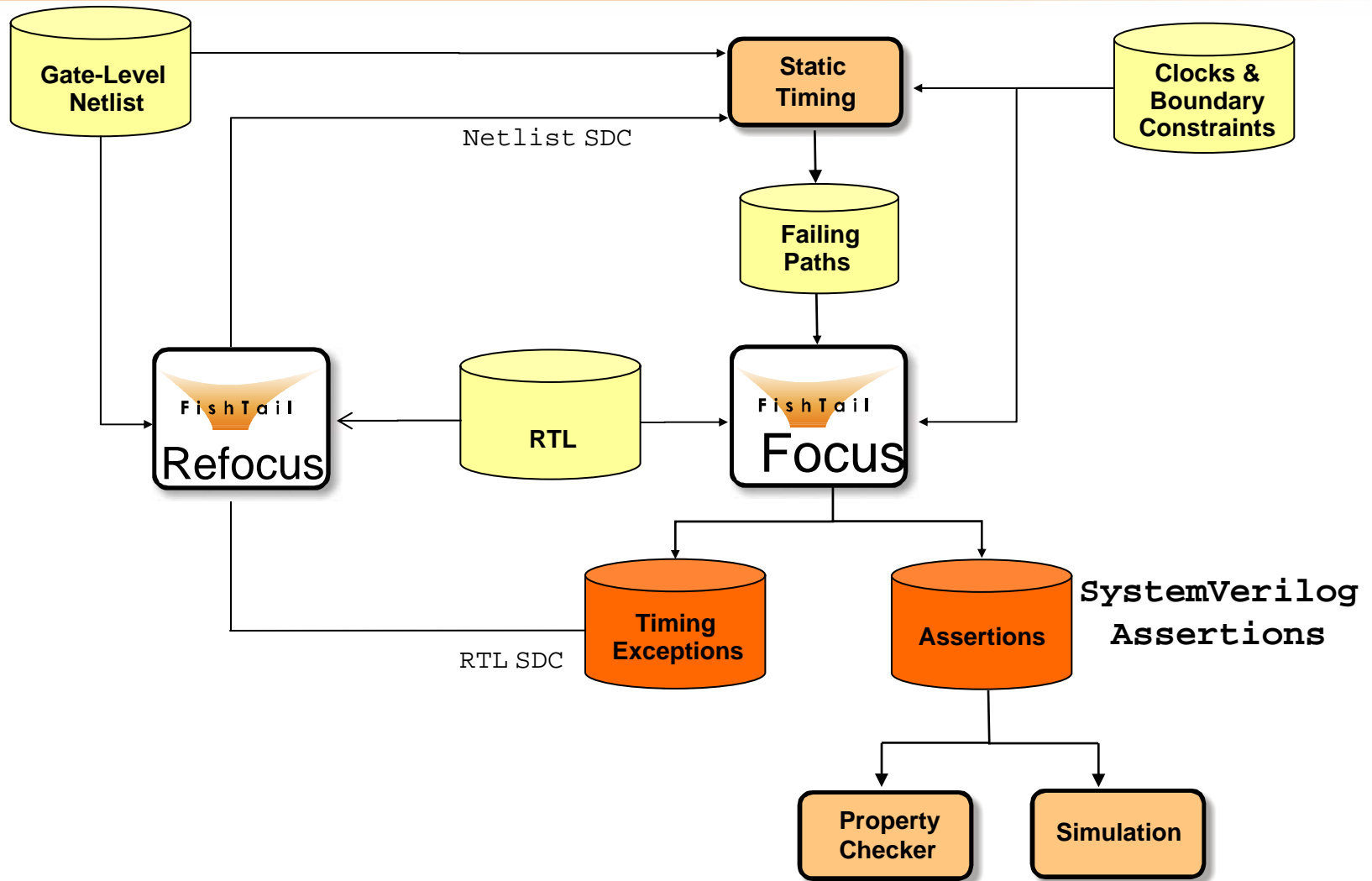
FishTail Value Proposition

- Reduction in design cycle
 - No need to manually specify & verify exceptions to clocking
 - No need to resort to timing accurate gate-level simulation
- Reduction in timing closure risks
 - Fewer iterations to converge on real timing challenges
- Improved QoR
 - Unnecessary logic optimization is not performed
- Plug and play in the design flow
 - Leverage standards
- Empower design handoff
 - Golden timing constraints are part of the handoff from design to implementation team

Timing Critical Exceptions for Synthesis

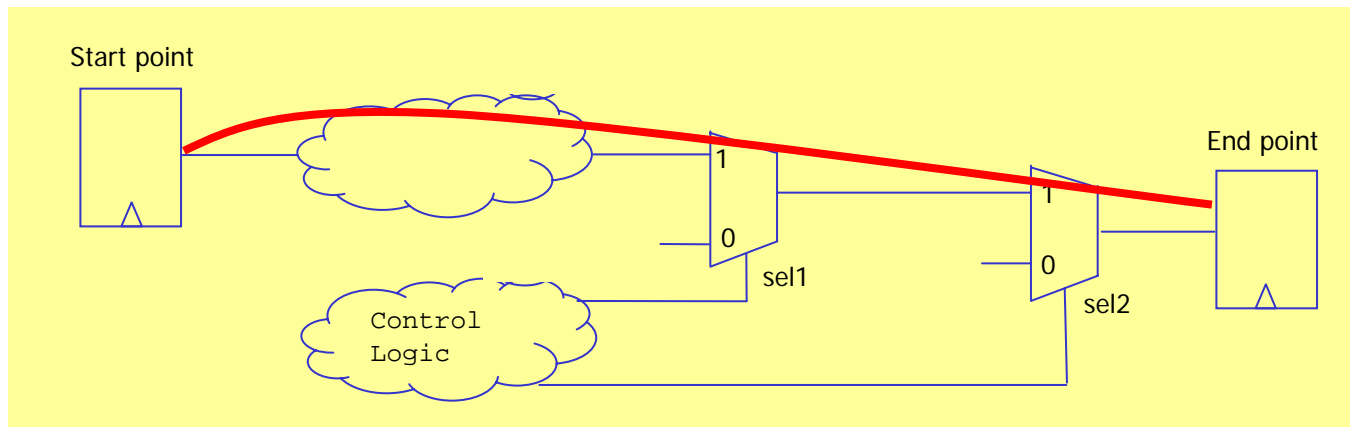


Timing Exceptions for Timing Closure



False-Path Generation and Verification

- Control Flow



- ```
set_false_path -from [get_cells startpoint_reg] \
 -through [get_pins pin] -to [get_cells endpoint_reg]
```

- Assertion

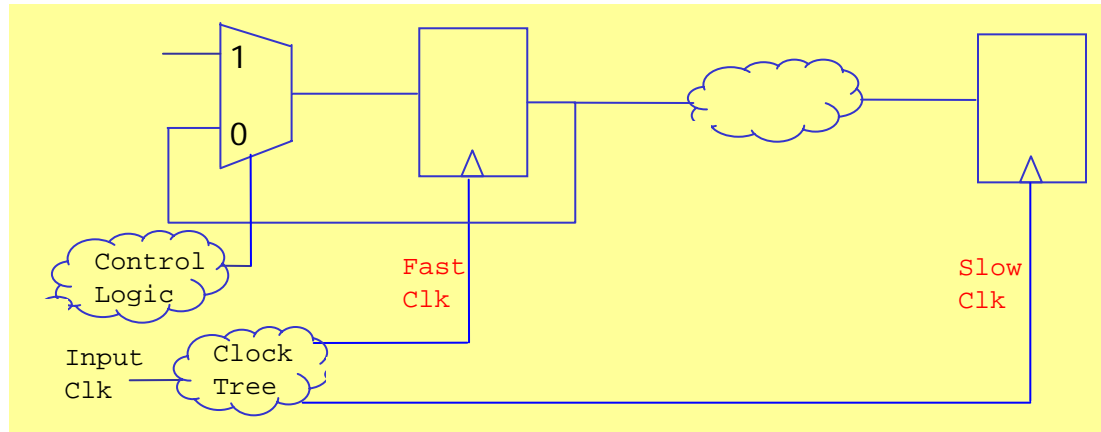
- ```
always @ (posedge clk)
    !(sensitization condition);
```

Example False Path Assertion in SVA Format

```
// fp1
// Sensitization condition:
//   ( r_e && notE )
// Startpoint clock: CLK
// Endpoint clock: CLK
// Assertion:
module u_fp1 (input clk, input rst, input v3, input v4);
  property e_fp1;
    @(posedge clk) !( v3 && v4 ) && rst;
  endproperty
  fp1: assert property(e_fp1);
endmodule
bind test u_fp1 sva_u_fp1(clk, rst, r_e, notE);
```

Multi-Cycle Path Generation & Verification

Different Launch and Capture Clocks



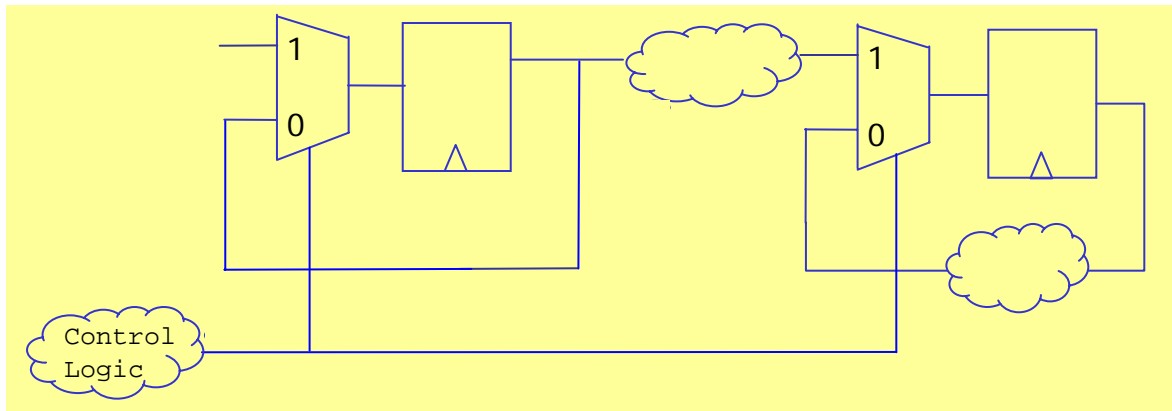
- `set_multicycle_path -from [get_cells startpoint_reg] \`
`-to [get_cells endpoint_reg] -start -setup <shift>`
- `set_multicycle_path -from [get_cells startpoint_reg] \`
`-to [get_cells endpoint_reg] -start -hold <shift - 1>`

Assertion

- `always @ (posedge fastclk)`
`if (startpoint != prev(startpoint)) ->`
`next (endpoint == prev(endpoint));`

Multi-Cycle Path Generation & Verification

- Same Launch and Capture Clocks



- ```
set_multicycle_path -from [get_cells startpoint_reg] \
```
- ```
-to [get_cells endpoint_reg] -end -setup <shift>
```
- ```
set_multicycle_path -from [get_cells startpoint_reg] \
```
- ```
-to [get_cells endpoint_reg] -end -hold <shift - 1>
```

Assertion

- ```
always @ (posedge fastclk)
```
- ```
if (startpoint != prev(startpoint)) ->
```
- ```
!(sensitization condition);
```

# Example MCP Assertion in SVA Format

```
// mcp1
// Sensitization condition:
// (uCounter/countX[1] && !uCounter/countX[0])
// Startpoint clock: clk
// Endpoint clock: clk
// Assertion:
module u_mcp1 (input clk, input [7:0] from_reg,
 input [15:0] to_reg,
 input rst, input v7, input v8);

 property e_mcp1;
 @(posedge clk)
 if (from_reg != past(from_reg) && rst) |->
 ##0 (!(v7 && !v8))*[2];
 endproperty

 mcp1: assert property(e_mcp1);
endmodule

bind top u_mcp1 sva_u_mcp1(clk2, regm[7:0], p[15:0],
 resetn, uCounter.countX[1],
 uCounter.countX[0]);
```

# Summary

- Standards have accelerated the development and deployment of our plug-and-play point tool, Focus
  - All inputs conform to standards
    - Verilog, VHDL, .lib, SDC
  - All outputs conform to standards
    - SDC
    - SystemVerilog Assertions (SVA)
- Standard interfaces are simpler & more powerful than proprietary interfaces
  - We use off-the-shelf building blocks for Verilog, VHDL, SDC, .lib readers
  - We integrate with multiple simulation and property checking tools using standard assertion formats
- Roadmap
  - Support for synthesizable SystemVerilog Q1'06