Reconfigurable Architecture (5)

osana@eee.u-ryukyu.ac.jp

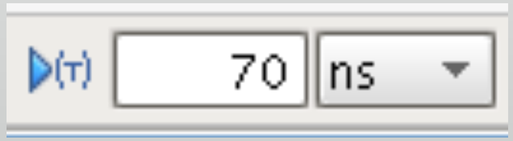
Review: Testbench

- `timescale, initial, always #
  - System tasks begins with “$”
  - Generate input signals to UUT
  - instance.signal to refer signals in UUT and its submodules

Review: Vivado

- Create project with FPGA order #
  - Keep source files in separate folder to protect (but not mandatory)
  - Add RTL files as “Design Source”
  - Add testbenches as “Simulation Source”

Review: Vivado Simulator

- Basic usage:
  - Set runtime to 0 in simulation settings
  - Advance simulation for Δt 

  - Changes in signal list:  Changes in source code: 
- Example source codes on the web
  - http://mux.eee.u-ryukyu.ac.jp/lecture.html.en

Today’s goal

- To make the board work
  - LED flashing, then 7 segment LED display
  - Write RTLs, simulate, then synthesize + place & route
- Using “parameter” in simulation / implementation
- Then, first assignment in this course

Implementation Flow

- Logic synthesis
- Technology mapping
- Place and route
- Bitstream generation
  - Details in hands-on

Today’s hands-on

- Synthesize, place & route, bitstream generation and FPGA programming
  - All Source files available on the web
    - Including the code on the slide

First goal of assignment

- Making a stopwatch
  - Push buttons + 7 segment LED display controller
  - Decimal counter
- Can be achieved using most of today’s source code

Push buttons

- Chattering filter
  - Last week’s example emits 5Hz pulse while keep pushed down
  - Optional task in the assignment

7 segment LED display

- Common anode
  - Column to illuminate: Anode=L
  - Segment to illuminate: Cathode=L
  - Quickly scanning all columns, show all figures with spectrum (dynamic drive)

Eight-digit Seven Segment Display

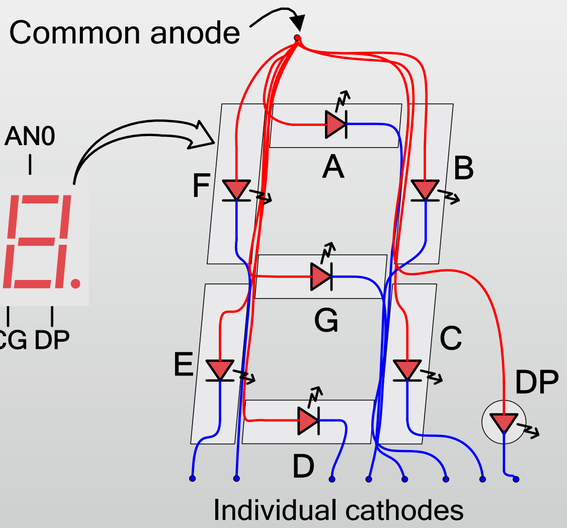


Figure 18. Common anode circuit node

Timescale: Human and Circuit

- FPGA runs at 100MHz, but we don't
  - LED at 100MHz is invisible, counters to make slower signals
  - 24bits counter is about 6Hz: 2^{24}=16M
  - But simulating up to 16M is not realistic
    - Better if different # bits can be used to simulate / implement

Led flashing, again

- Red figure is to be changed
  - 23 to implement
  - 2 to simulate: enough fast to trace on waveform

```
module led_kurukuru
    ( input  wire CLK, RST,
      output reg [15:0] LED );

  reg [23:0] CNT;
  wire STROBE = &CNT;

  always @ (posedge CLK) begin
     if (RST) begin
        CNT <= 0;
        LED <= 16'b1000_0000_0000_0000;
     end else begin
        CNT <= CNT+1;
        if (STROBE)
          LED <= {LED[0], LED[15:1]}
    end
  end
endmodule
```

New syntax: Parameter

- 2 ways to declare:
  - in module
  - or before port definitions
- Both requires default value
- The latter can change port width

Style 1:

```
module led_kurukuru
    ( input wire CLK, RST,
      output reg [15:0] LED );

  parameter CounterBits = 24;
  reg [(CounterBits-1):0] CNT;
  (no changes below)
```

Style 2:

```
module led_kurukuru #
    ( parameter CounterBits = 24 )
    ( input CLK, RST,
      output reg [15:0] LED );

  reg [(CounterBits-1):0] CNT;
  (no changes below)
```

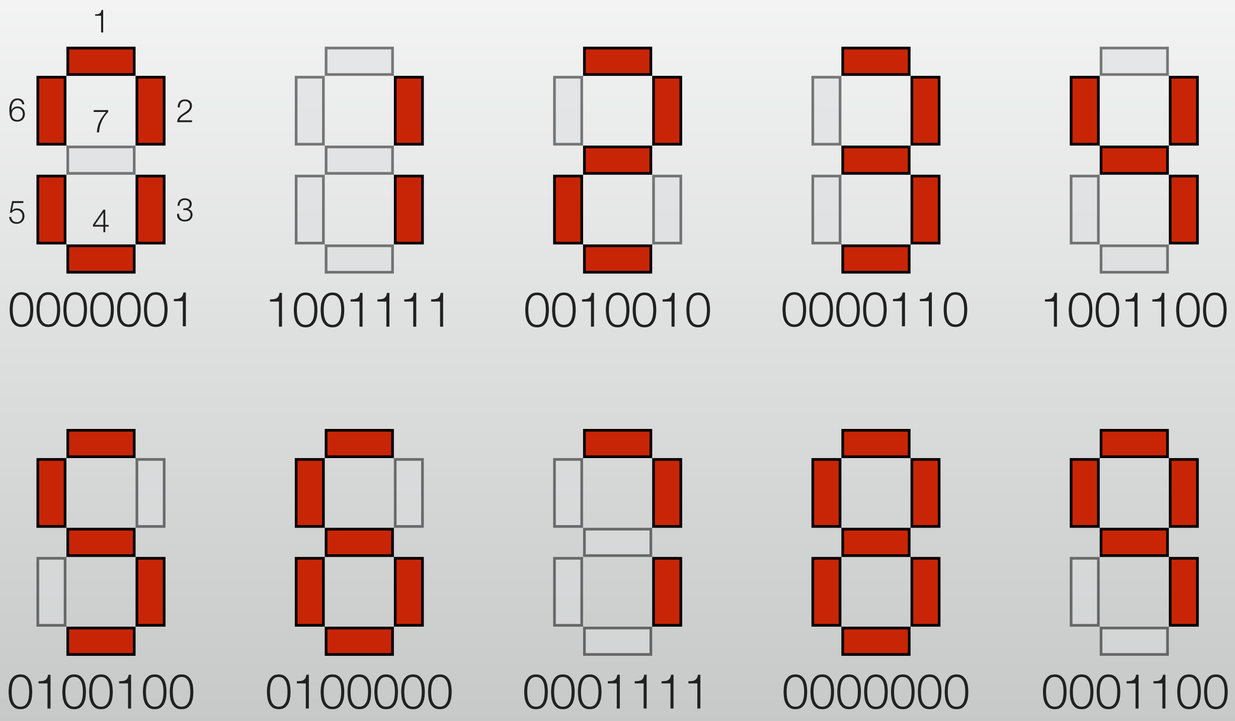
Changing parameters

- Do nothing for 24bits
  - Parameters can be overwritten on instantiation
  - kuru2 has a 2 bit counter
- Handy for faster simulation

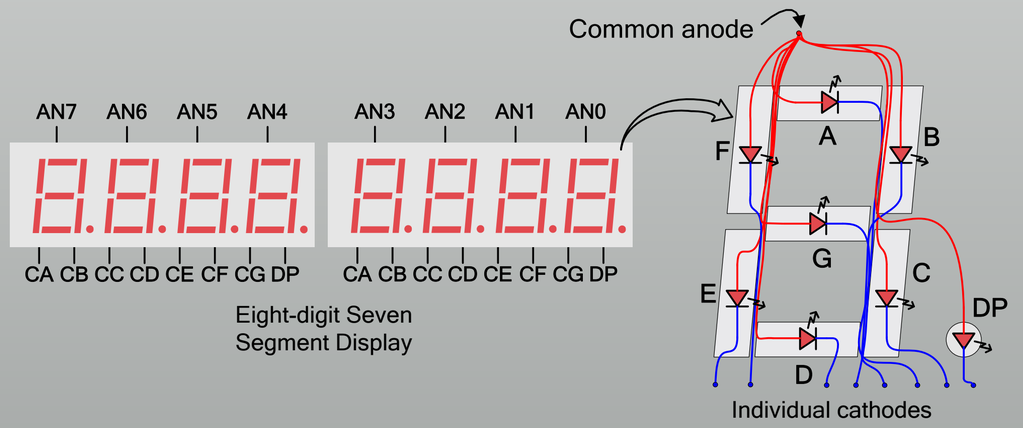
```
led_kurukuru
  kuru1(.CLK(CLK), .RST(RST), .LED());

led_kurukuru  # ( .CounterBits(2) )
  kuru2(.CLK(CLK), .RST(RST), .LED());
```

Lab

7-seg LED

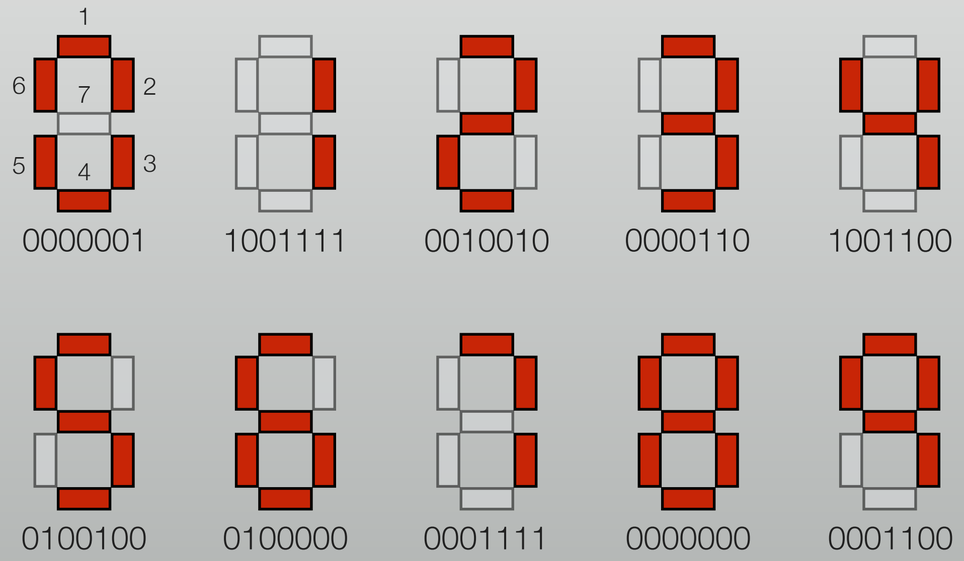


- 0~9 figures and their pattern
  - 7bit (without dot) for cathode, 0 to illuminate
- Anode (column) later



Single-column driver

- 4bit in → 7bit out

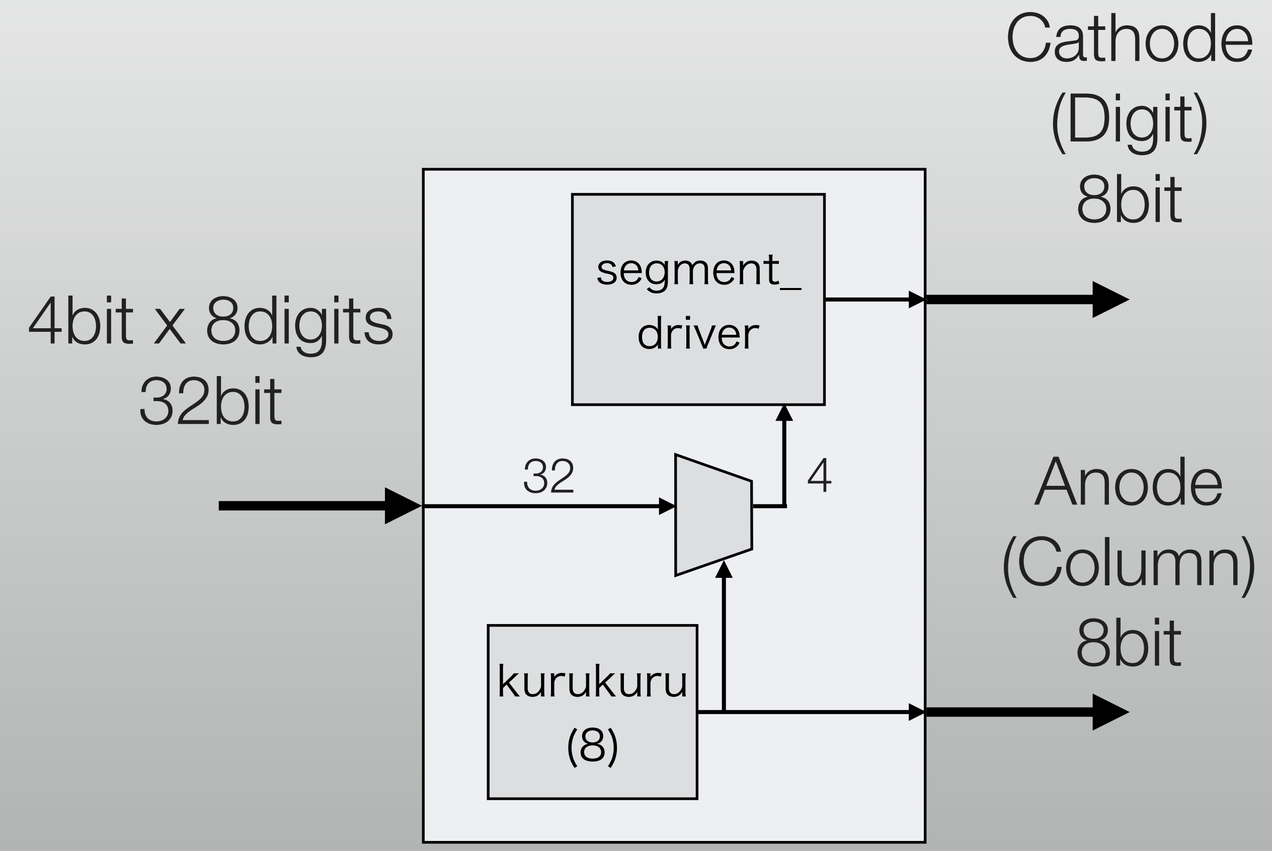



```
module segment_driver
  ( input  wire [3:0] VAL,
    output wire [6:0] CATHODE );

  assign CATHODE =
      (VAL==4'h0) ? 7'b000_0001 :
      (VAL==4'h1) ? 7'b100_1111 :
      (VAL==4'h2) ? 7'b001_0010 :
      (VAL==4'h3) ? 7'b000_0110 :
      (VAL==4'h4) ? 7'b100_1100 :
      (VAL==4'h5) ? 7'b010_0100 :
      (VAL==4'h6) ? 7'b010_0000 :
      (VAL==4'h7) ? 7'b000_1111 :
      (VAL==4'h8) ? 7'b000_0000 :
      (VAL==4'h9) ? 7'b000_1100 : 7'h0;
endmodule
```

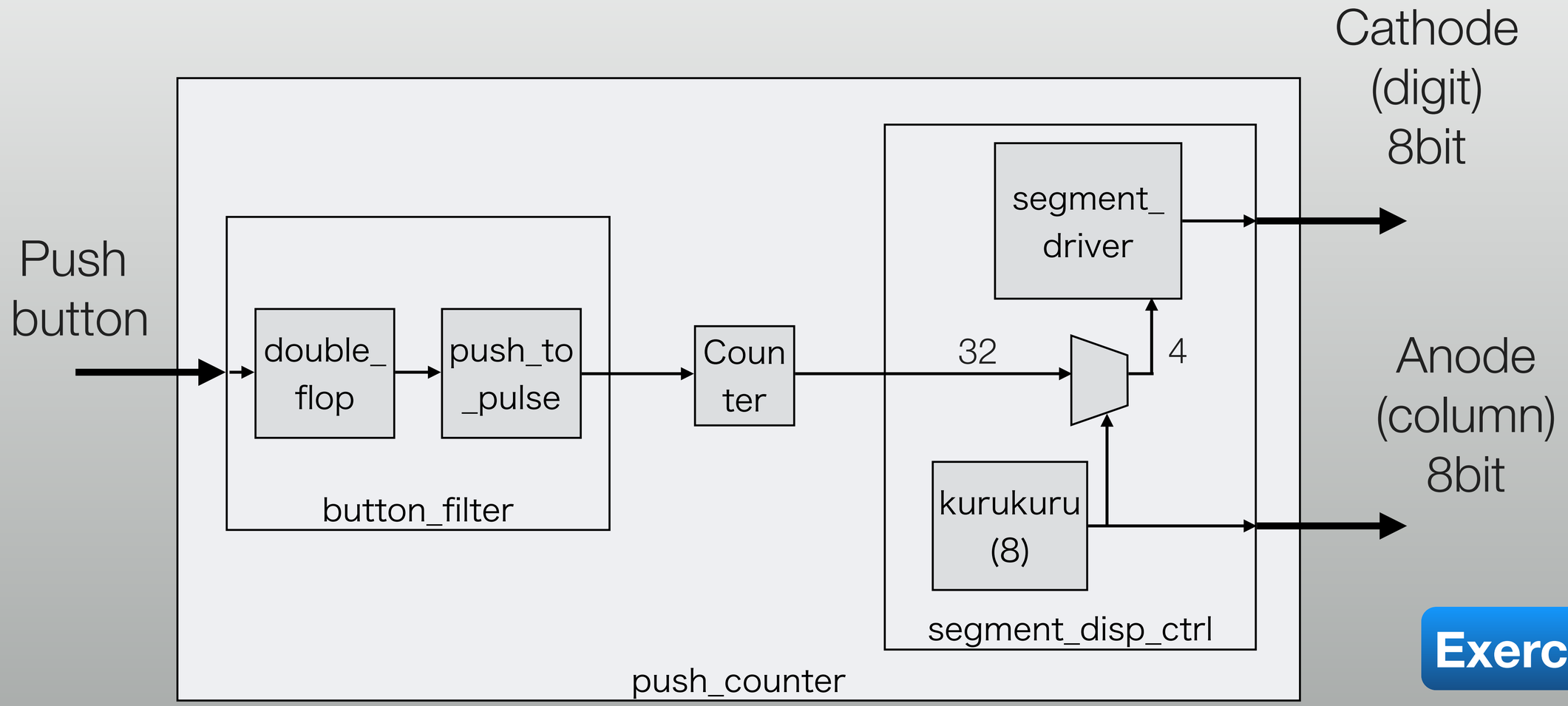
Logic compression done by Vivado

Overall structure

- 8 columns = 32bits in
  - 4bits for each column
  - Hexadecimal is also possible with modified segment_driver module



Make it work on FPGA board



Exercise

Today's Assignment

- Make a stopwatch
  - Minimum 1/10000s
  - Maximum 9999.9999s
- Buttons: RST and START-STOP, latter requires chattering removal
- Suppose that the board's 100MHz clock is 100% reliable

FPGA boards for lab/assignment

- Digilent Nexys4 (discontinued)
- Digilent Nexys4 DDR and Nexys A7-100T (they’re same)
  - Pin assignments differ between Nexys4 <-> DDR / A7
  - **Choose the right XDC file, or you’ll break the FPGA**

Lab 1: Flashing LEDs

Goal of the lab

- Different # bits to simulate and implement
  - Change the parameter in simulation
  - Source files are available on the web
- Understand Vivado usage in its implementation flow

Source codes

- Available from http://mux.eee.u-ryukyu.ac.jp/lecture.html.en
  - Download and unzip lab-05.zip
  - Constraint: nexys4.xdc or nexys4ddr.xdc
  - Design Source: led_kurukuru.v
  - Simulation Source: led_kurukuru_test.v

Make a project

- Extract the zip file in working directory: 05-flash-led + 05-btn-cnt
- Make a Vivado project in 05-flash-led/vivado
  - Device: XC7A100T-1CSG324 (xc7a324tcsg324-1 in vivado)
  - 3 source files: RTL, TB and constraint
- **Important:** Choose the right constraint file for your board!

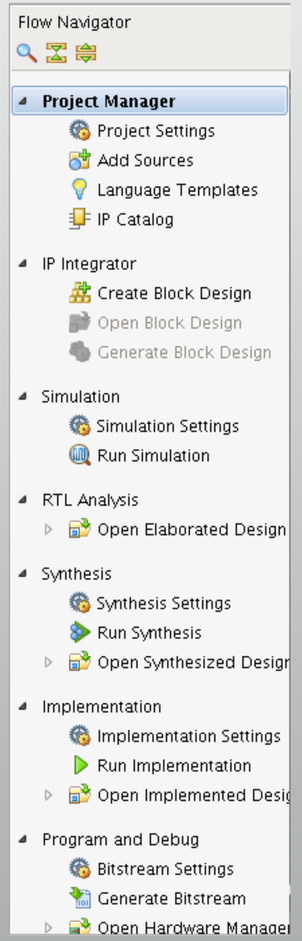
Run simulation

Flow Navigator

- Project Manager
- Simulation
- RTL Analysis
- Synthesis
- Implementation: technology mapping + P&R
- Program and Debug: Bitstream generation and programming

Implementation flow (up to down)

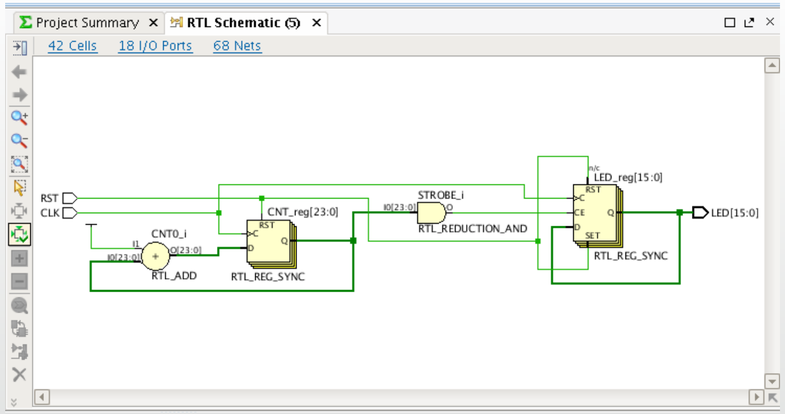
(IP Integrator is not in today's scope)



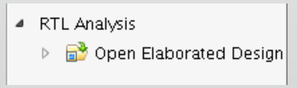
Project Manager

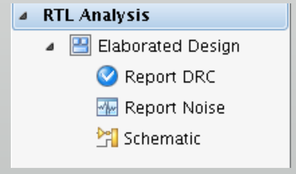
- is the initial screen
  - Target device and project summary
- On right bottom:
  - Messages / Log: check if you’ve got errors
  - Design Runs: progress of Implementation flow tasks

RTL Analysis

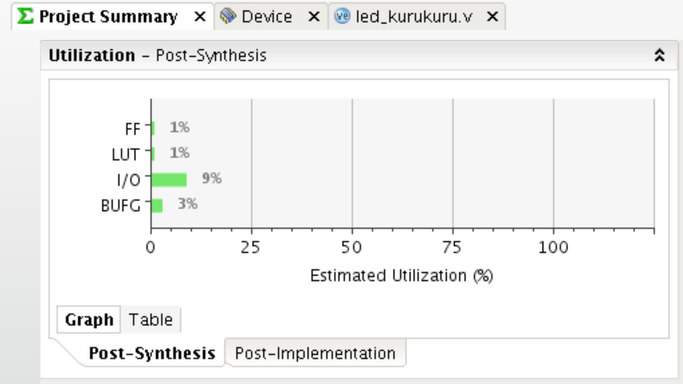


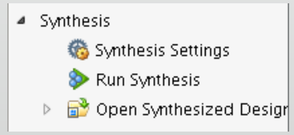
- “Open Elaborated Design” to initiate
  - Launched automatically on synthesis and later commands
- Schematic is linked to RTL, jump on right click
  - Not very important, just for check

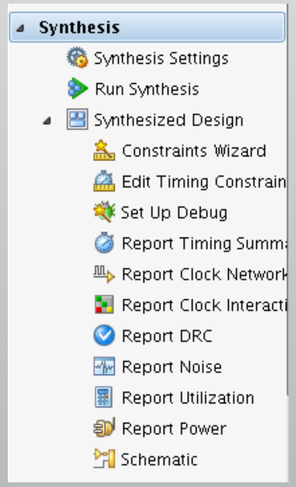




Synthesis



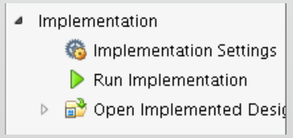


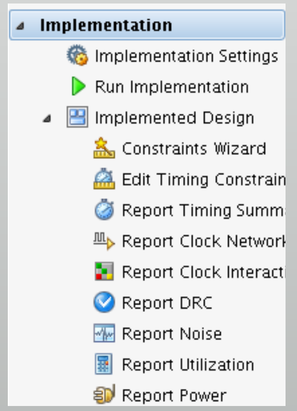


- “Run Synthesis” to launch
  - Open Synthesized Design to see reports/schematics
    - Only I/Os shown in device view (because the circuit isn’t placed yet)
  - Resource estimate in Project Summary
    - Fixed after implementation, just an estimation at this point

Implementation

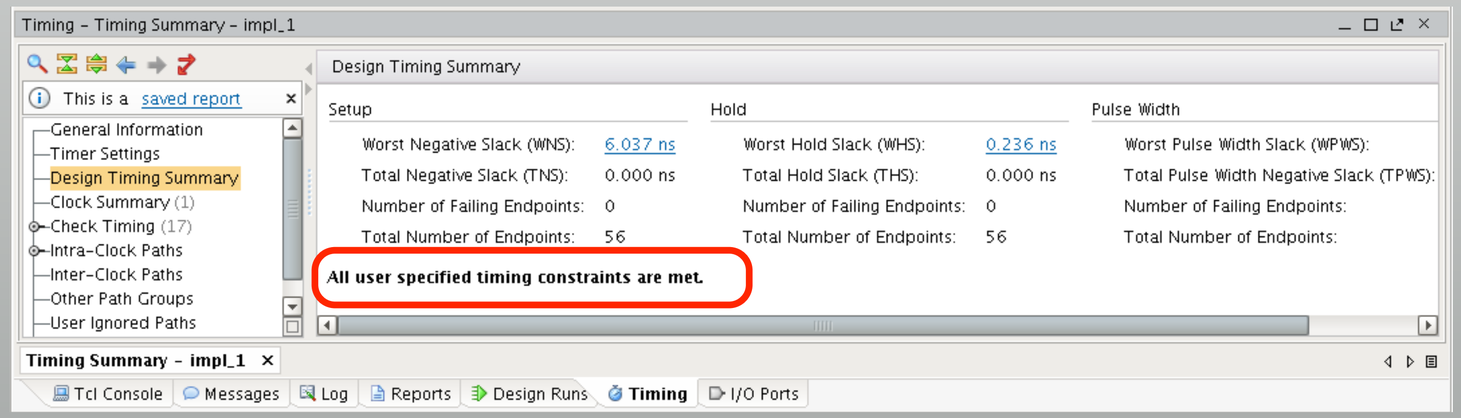
- “Run Implementation” to launch
  - Whole circuit is placed and routed
  - Resource usage in “Project summary” is fixed
  - Timings and I/O results are **very** important





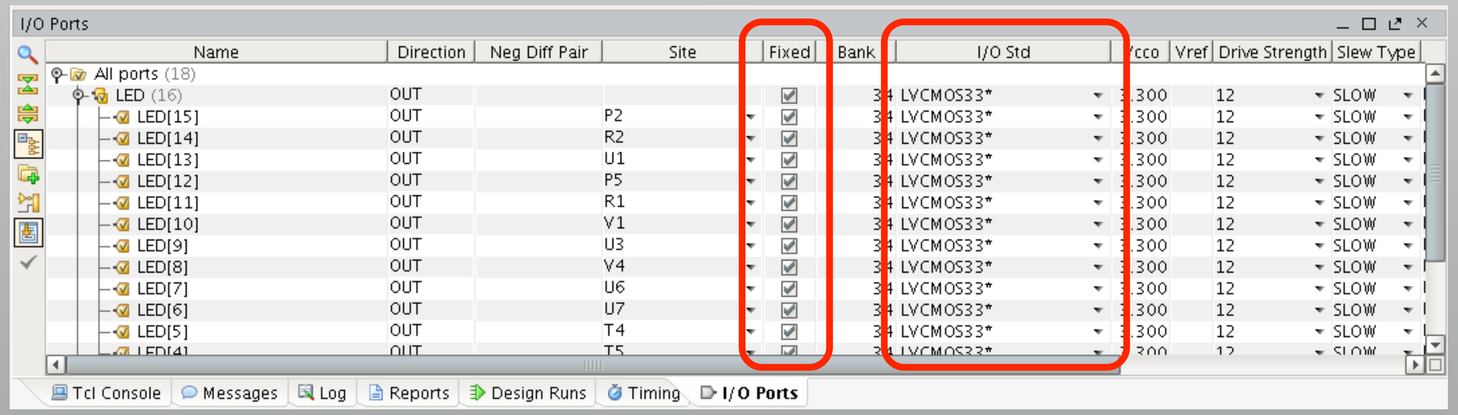
Implementation: Timing

- Available after implementation
  - Open Implemented Design → Window → Timing
  - “All user specified timing constraints are met” is what we want

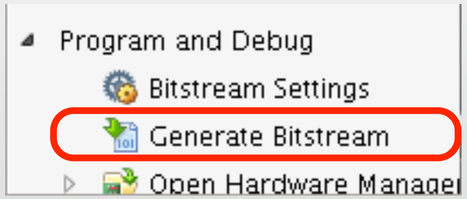


Implementation: I/O Ports

- Open Implemented Design → Window → I/O Ports
  - All ports must be “Fixed” and their I/O Standard must be correct
  - If not, the board/FPGA may be broken



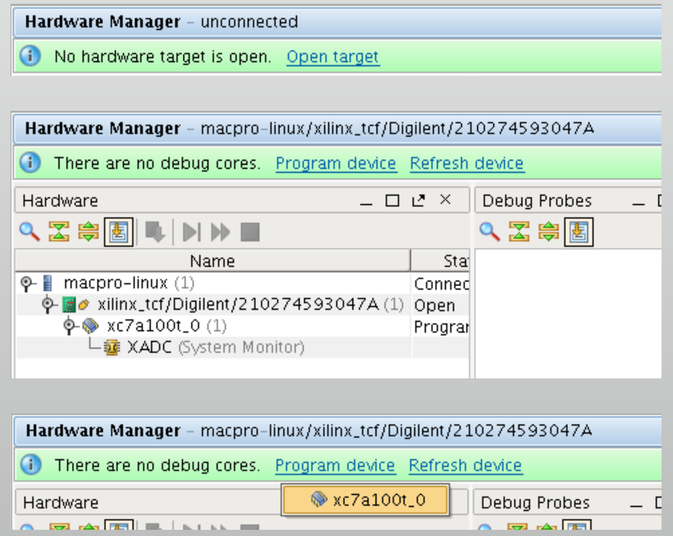
Bitstream generation



- Place & Route result to be written on an FPGA
  - Just click on “Generate Bitstream” to synthesize, implement and generate bitstream
  - This is handy, but will be sometimes too time consuming

Hardware Manager

- Connect the board and turn power on
- “Open Target” to find the FPGA
- “Program device” to program
  - Default filename is good in typical use case



Check it:

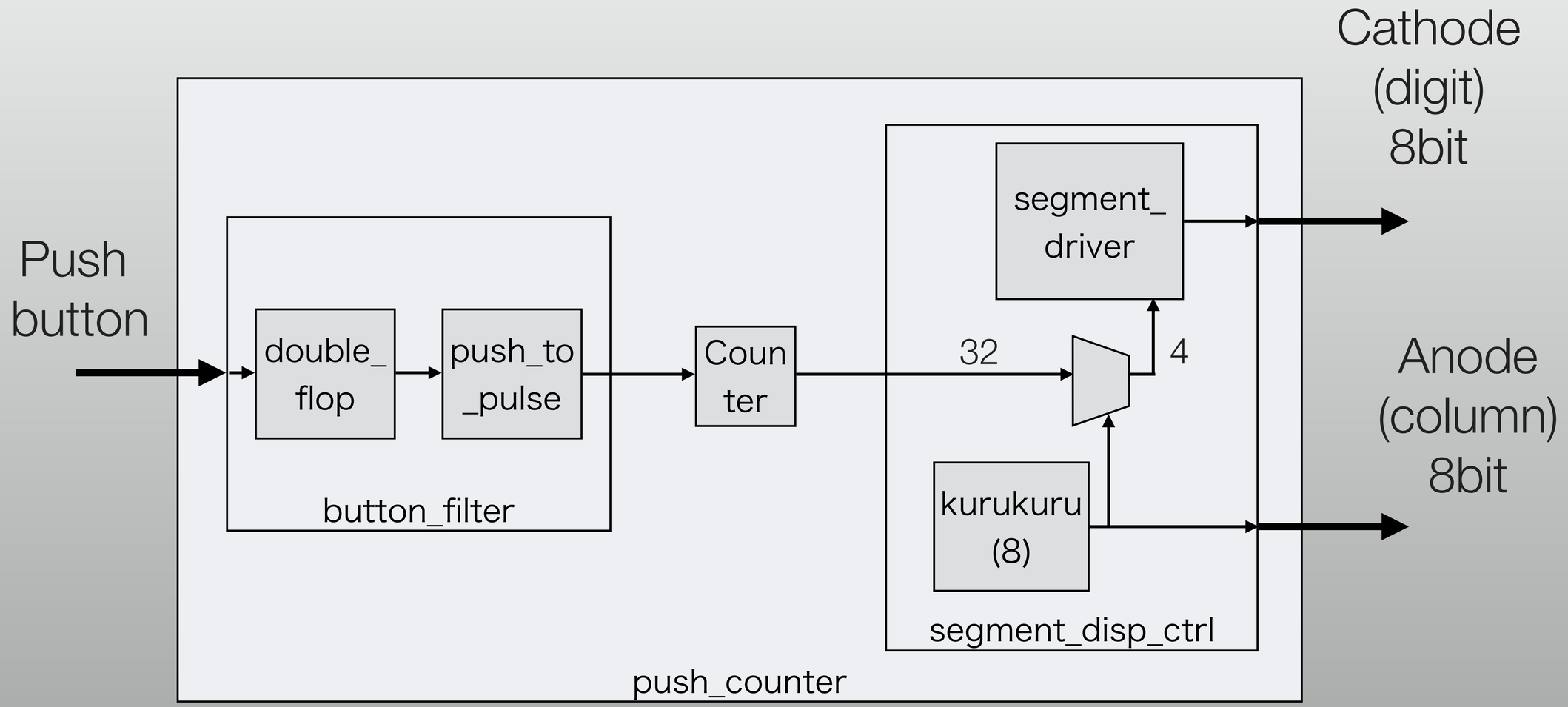
* LED flashes from left to right, in different speed with simulation
  * Also try to change parameter for simulation

Lab 2:
Hexadecimal LED counter

Make a project

- File → New Project
- Sources in 05-btn-cnt/src
  - push_counter_test.v: TB,
    nexys4.xdc or nexys4ddr.xdc: constraints
  - All others as design sources

Organization



Button assignments

- Left (BTNL) : ADV
  - Increment the counter
- Center (BTNC): RST
  - Reset

Parameters

- LED column switching speed (ColumnCounterBits)
  - 10bit counter = 1/100 ms (may be too fast…)
- Chattering filter counter (ButtonFilterCount)
  - 20×10^6 = 1/5sec

Try it

- Keep pressing down left button
  - Counter++ at 5Hz
- ColumnCounterBits = 24 in push_counter.v
  - Slower LED column scanning, same to exercise 1

Assignment

- Stopwatch
  - You can use any of today's source codes
- Deadline Nov.20
  - Send me zipped all source codes (*.v and *.xdc),
    with a description of the design's organization in PDF (1 or 2 pages in A4)

Note for remote programming

- Sharing an FPGA card in the lab
  - Connect the card to a host
  - Launch “hw_server” on the host

    (no full version of Vivado is required, included in Vivado Lab edition)
- Choose “Remote server” on launching HW manager, “Auto connect” is not available for remote use