Common digital structures

Lecture 5

Common digital structures

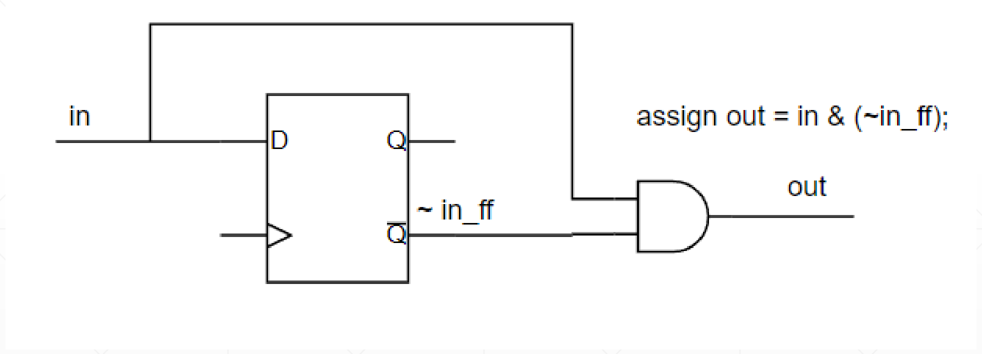
- Digital designs are made up of simple logic cells.
- But those cells can make larger structures that are just as commonly used in digital design.
- In industry you will often find yourself re-using previously designed blocks or parts of blocks

Level to pulse

- Often control signals can be input as a constant level to indicate status.
- But often our logic expects a single clock pulse to trigger events.
- How can we make this conversion?

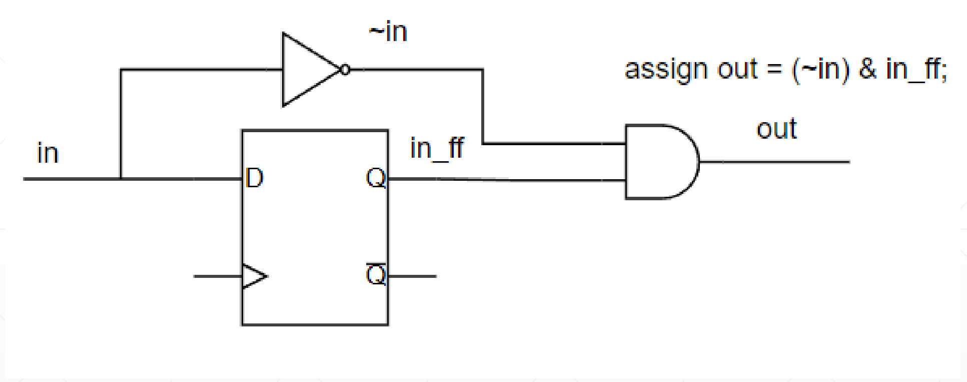
Level to Pulse

- Rising edge detection
- Input has gone high but delayed version is still low.



assign out = in & (~in_ff);

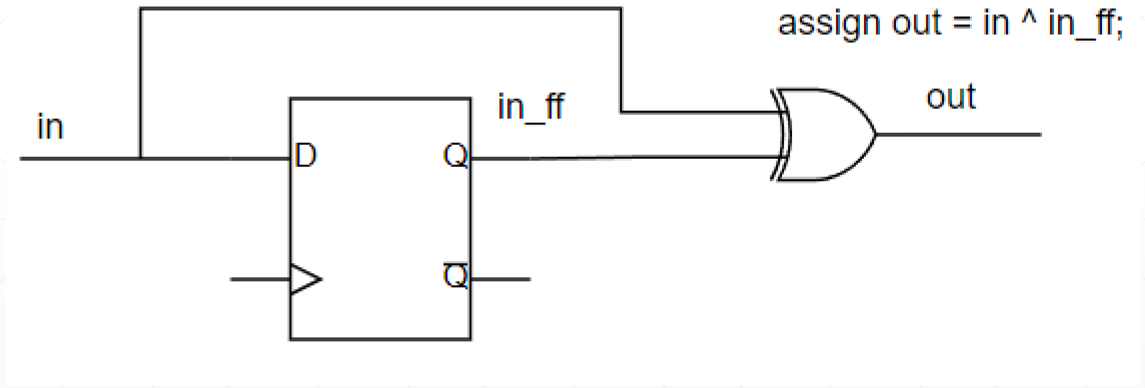
- Falling edge detection
- Input has gone low but delayed version is still high.



assign out = (~in) & in_ff;

Level to Pulse

- Any edge detection
- Input and delayed version are different



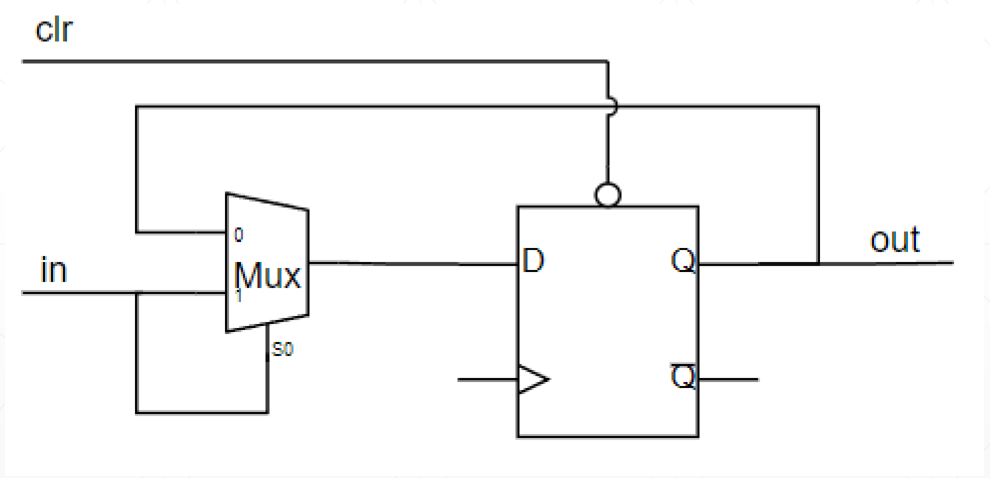
Real world scenarios in the digital domain

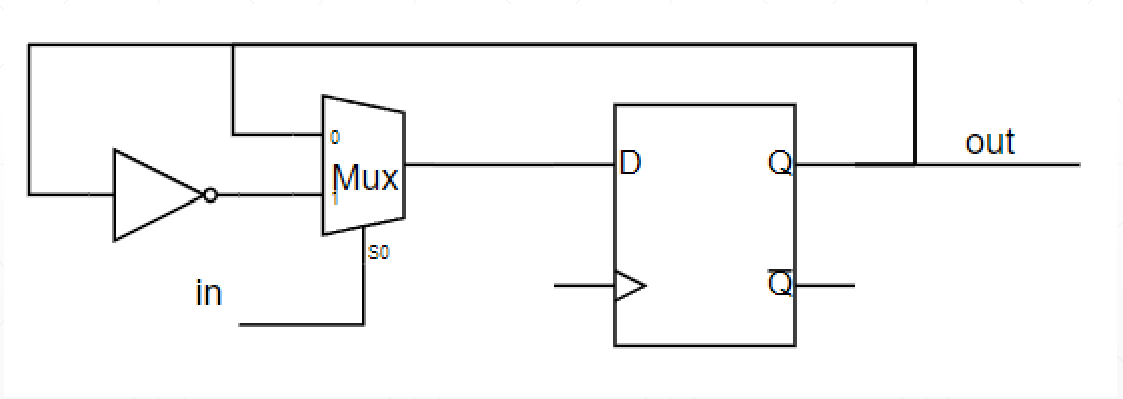
- Most of what we think of in digital design is idealized
- RTL simulations have zero delay and even asynchronous signals will driven in a synchronous way.
- This leads to a situation where poor design can generate very serious bugs which may not be found in simulation.
- So what sort of scenarios can cause these issues, and how do we catch them/ design around them?

Pulse to level

- Input is “latched”* into register.
- Clear signal is required to drop level

- Input pulse toggles register.
- No clear signal required



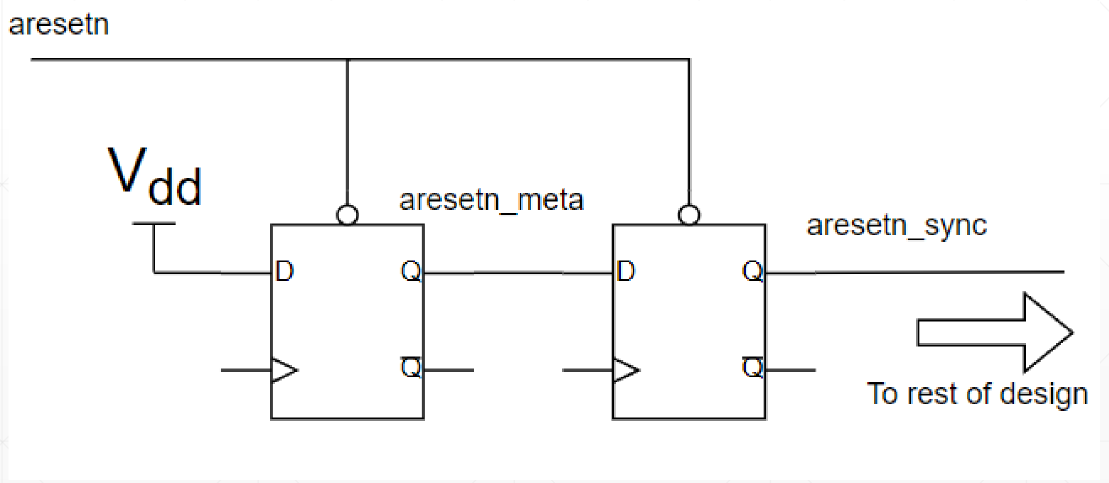


Reset synchronization

- In ASICs we use asynchronous resets
- Does this cause any sort of issue?

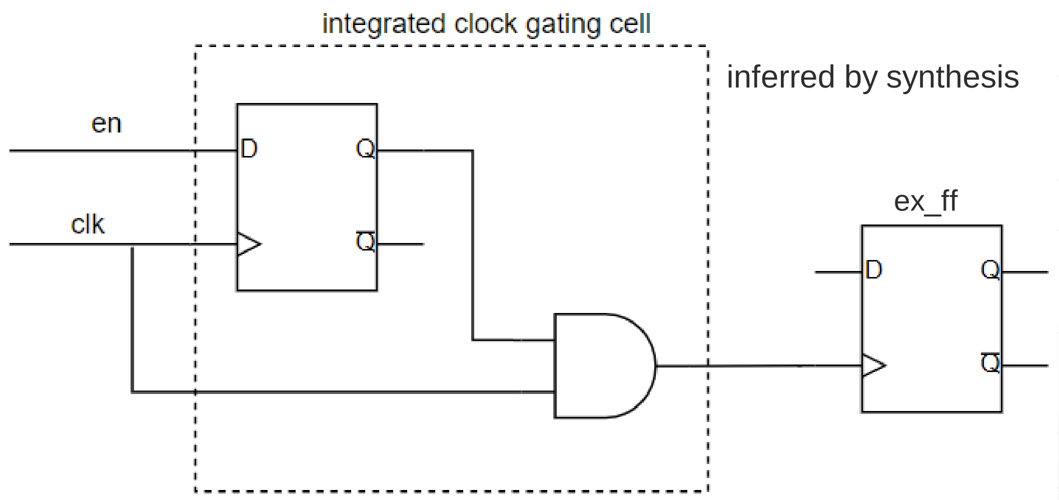
Reset synchronization

```
reg [3:0] aresetn_meta, aresetn_sync;
always @ (posedge clk or negedge aresetn) begin

    if (!aresetn) begin
       aresetn_meta <= ‘b0;
       aresetn_sync <= ‘b0;
    end else begin
       aresetn_meta <= 1‘b1;
       aresetn_sync <= aresetn_meta;
    end

end
```



ICGC (Integrated clock gating cell)

```
always @ (posedge clk) begin

    if (reset) begin
         ex_ff <= ‘b0;
    end else begin
       if(en)
         ex_ff <= ex_in;
       else
         ex_ff <= ex_ff;
    end

end
```

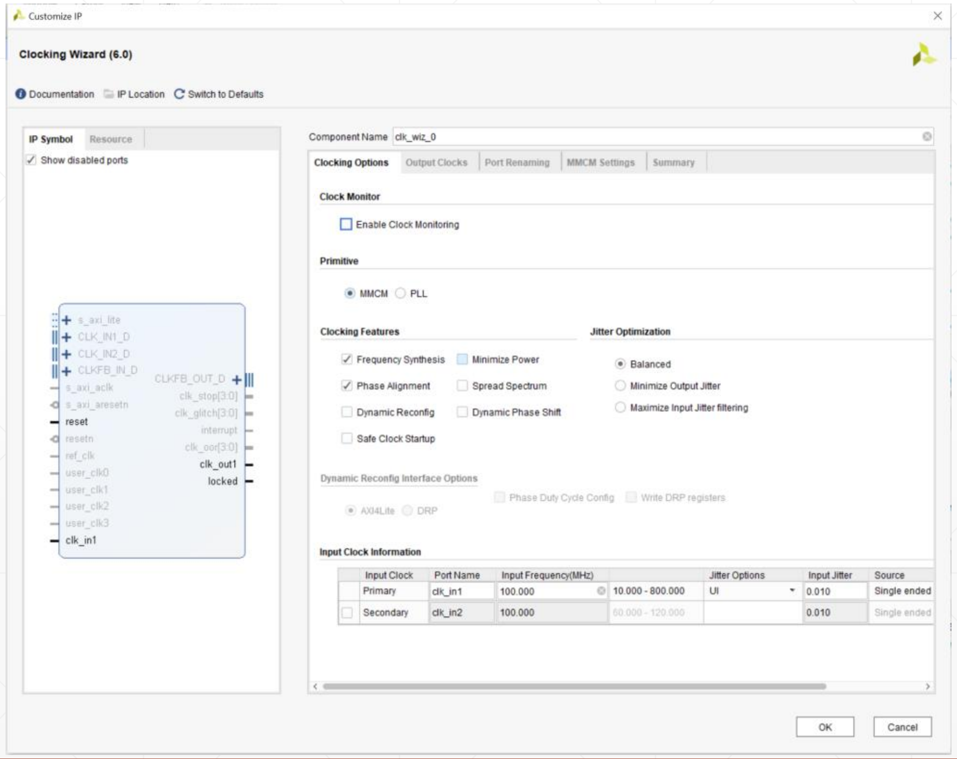


Clock generation

- Many ICs utilize an internal oscillator, Implemented in analog which will supply the digital logic.
- Other options are externally supplied clock references.
- Often we may wish to generate extra clocks that operate at different frequencies.
- How can this be done, and how does it look on FPGA vs ASIC?

Clock Generation (FPGA)

- This can be done using the built in MMCM or PLL resources.
- Can be instantiated directly, but much easier to use clocking wizard IP in Vivado. https://www.xilinx.com/support/documentation/ip_documentation/clk_wiz/v6_0/pg065-clk-wiz.pdf



Component Name: clk_wiz_0

Clocking Options | Output Clocks | Port Renaming | MMCM Settings | Summary

Clock Monitor

- ☐ Enable Clock Monitoring

Primitive

[✓ MMCM] PLL

Clocking Features

- ✓ Frequency Synthesis
- ☐ Minimize Power
- ✓ Phase Alignment
- ☐ Spread Spectrum
- ☐ Dynamic Reconfig
- ☐ Dynamic Phase Shift
- ☐ Safe Clock Startup

Jitter Optimization

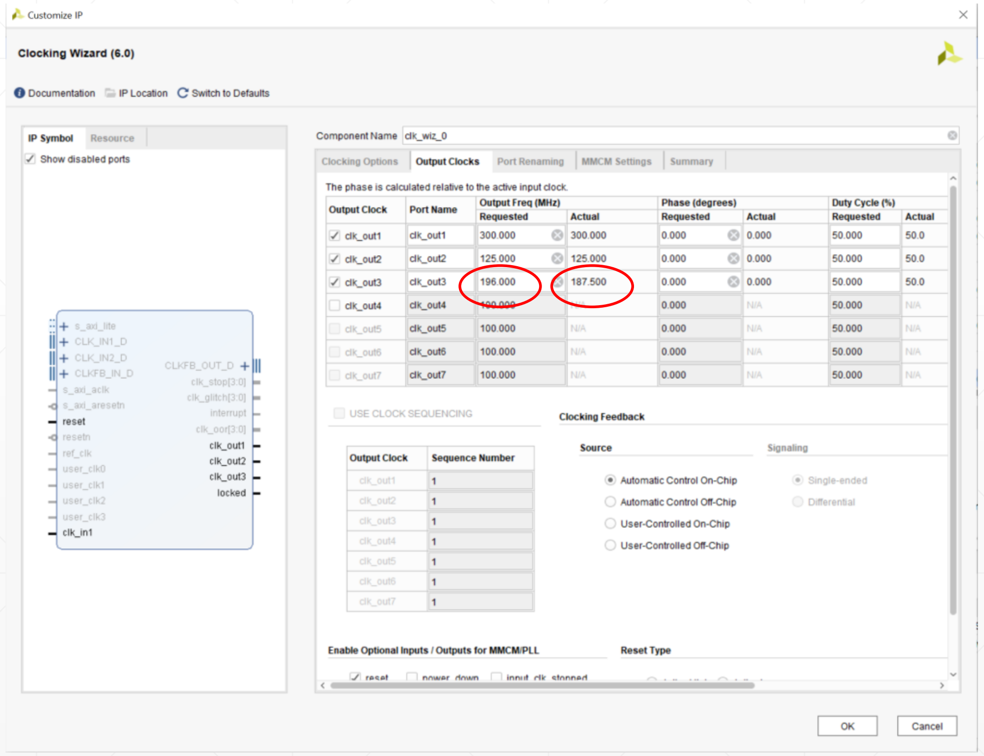
[✓ Balanced] Minimize Output Jitter / Maximize Input Jitter filtering

Dynamic Reconfig Interface Options

[✓ AXI4Lite] DRP; ☐ Phase Duty Cycle Config; ☐ Write DRP registers

Input Clock Information

	Input Clock	Port Name	Input Frequency(MHz)		Jitter Options	Input Jitter	Source
	Primary	clk_in1	100.000	10.000 - 800.000	UI	0.010	Single ended
☐	Secondary	clk_in2	100.000	60.000 - 120.000		0.010	Single ended



Component Name: clk_wiz_0

Clocking Options | Output Clocks | Port Renaming | MMCM Settings | Summary

The phase is calculated relative to the active input clock.

Output Clock	Port Name	Output Freq (MHz)		Phase (degrees)		Duty Cycle (%)	
		Requested	Actual	Requested	Actual	Requested	Actual
✓ clk_out1	clk_out1	300.000	300.000	0.000	0.000	50.000	50.0
✓ clk_out2	clk_out2	125.000	125.000	0.000	0.000	50.000	50.0
✓ clk_out3	clk_out3	196.000	187.500	0.000	0.000	50.000	50.0
☐ clk_out4	clk_out4	100.000	N/A	0.000	N/A	50.000	N/A
☐ clk_out5	clk_out5	100.000	N/A	0.000	N/A	50.000	N/A
☐ clk_out6	clk_out6	100.000	N/A	0.000	N/A	50.000	N/A
☐ clk_out7	clk_out7	100.000	N/A	0.000	N/A	50.000	N/A

☐ USE CLOCK SEQUENCING

Output Clock	Sequence Number
clk_out1	1
clk_out2	1
clk_out3	1
clk_out4	1
clk_out5	1
clk_out6	1
clk_out7	1

Clocking Feedback

Source: [✓ Automatic Control On-Chip] Automatic Control Off-Chip / User-Controlled On-Chip / User-Controlled Off-Chip

Signaling: [✓ Single-ended] Differential

Enable Optional Inputs / Outputs for MMCM/PLL

✓ reset ☐ power_down ☐ input_clk_stopped

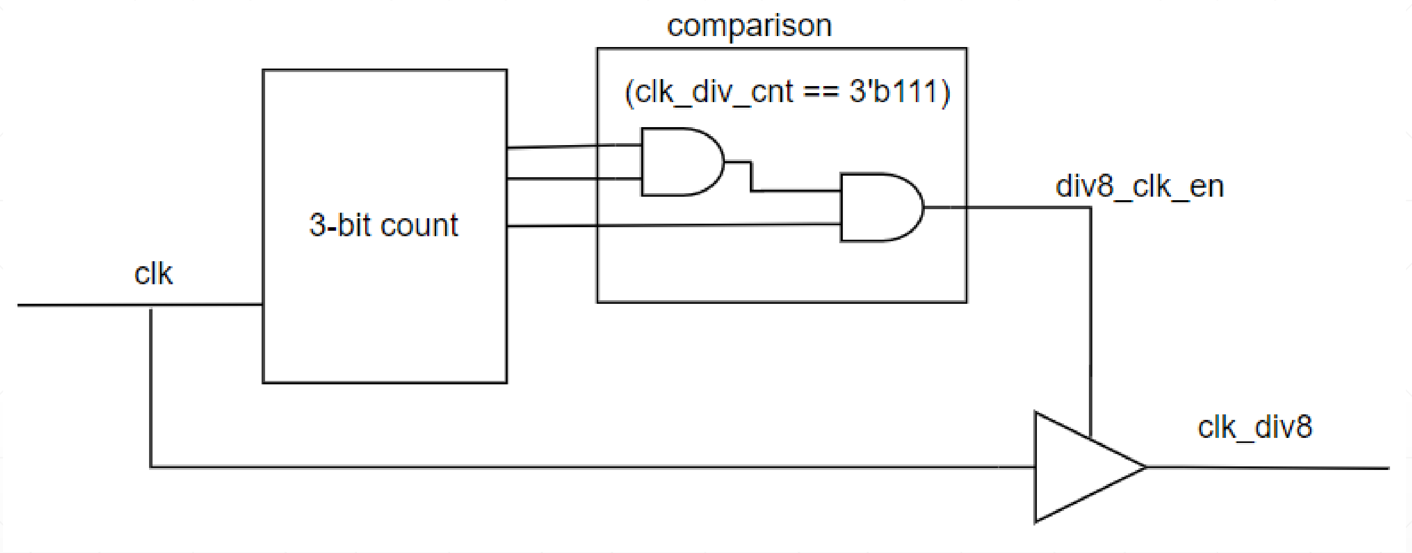
Reset Type

Cock generation (ASIC)

- Clock divider circuit
- Clock multiplication is also possible. Requires highly characterized logic circuits.

```
reg [2:0] clk_div_cnt , aresetn_sync;
always @ (posedge clk or negedge aresetn) begin

    if (!aresetn) begin
       clk_div_cnt<= 3‘b000;
    end else begin
       clk_div_cnt <= clk_div_cnt + 1;
    end

end

div8_clk_en = (clk_div_cnt == 3’b111);

clk_gating_cell xclk_gating_cell(.clk_in(clk), .clk_out(clk_div8), .en(div8_clk_en) );
```

Cock generation (ASIC)


comparison
(clk_div_cnt == 3'b111)
3-bit count
clk
div8_clk_en
clk_div8