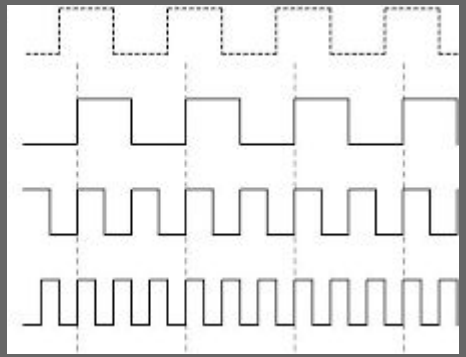
# **CPE 470 -** Clock Domains



# Clock Domains

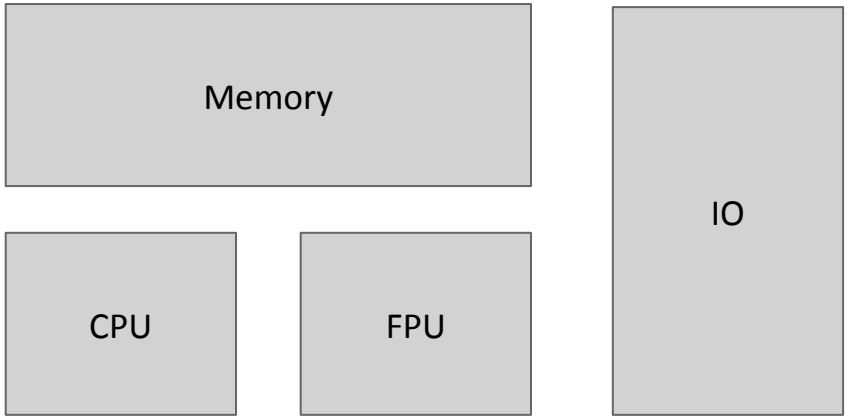
**Glossary**

**Clock Domain:** all the synchronous elements controlled by one specific clock signal

- Different parts of a digital system have different logic depths and clock needs
  - Floating Point Unit will have deeper logic than CPU's integer ALU
  - IO might need to be slow (I2C) or extremely fast (PCIe)
- If all components share the same clock, they are limited by slowest component

- Solution → Separate **Clocks Domains**
  - Deeper logic, Memory, SRAM gets slower clocks
  - Cheap computation gets fast clock

Where do all these clocks come from?
How do we navigate between them?



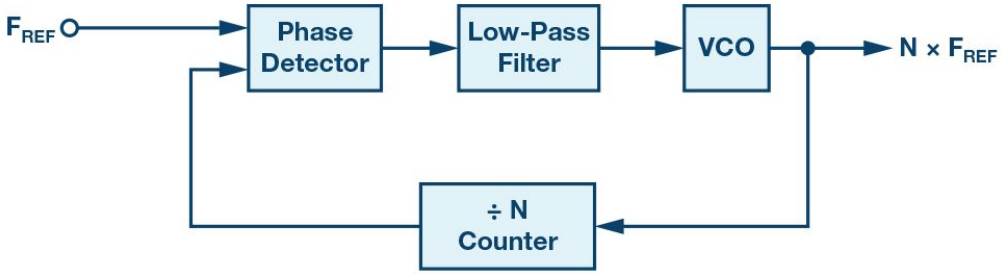
# Clock Generation & PLLs

- Two ways to get clocks
  - Off-chip ⟶ crystal oscillator
    - Higher frequencies (50+ Mhz) are hard to pcb layout
  - On-chip ⟶ clock divider
  - On-chip ⟶ generate with a **PLL**

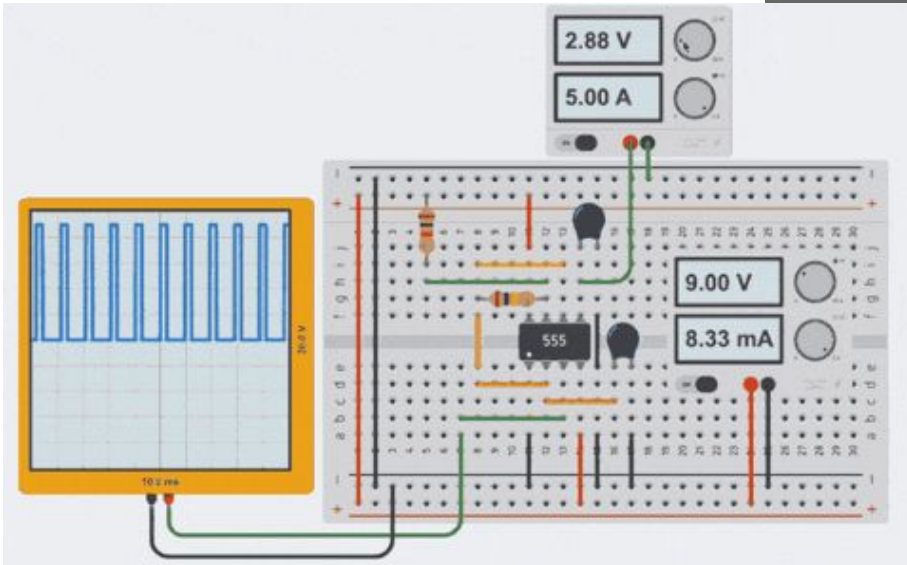
**Glossary**

**PLL:** phased-locked loop; an analog circuit that uses feedback to multiply a clock
**VCO:** voltage-controlled oscillator changes output frequencies based on input voltage

- PLLs act as clock multipliers
  - Take in a reference clock and multiply it
  - Uses a **VCO** to generate an arbitrarily fast clock
  - Uses feedback to tune VCO to correct frequency




555 Timer as a basic VCO

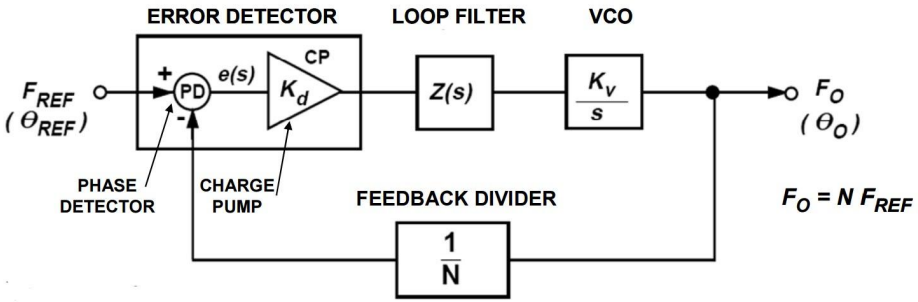


# PLL Types

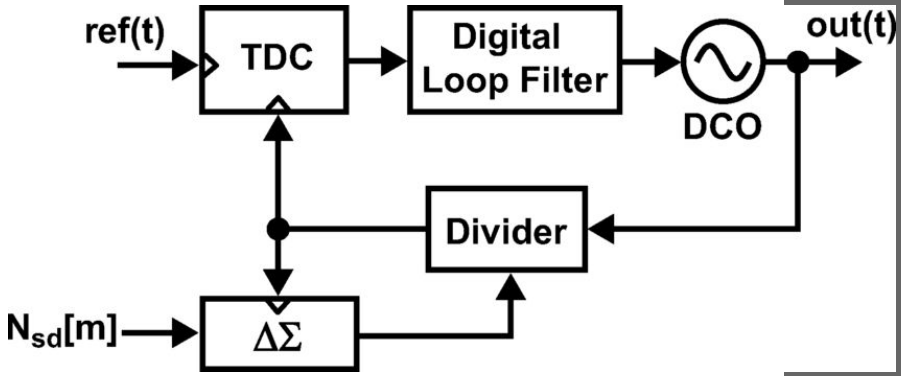
**Glossary**
**APLL:** analog PLL
**DPLL:** digital PLL

- **DPLL**s: digital logic for phase detection, low pass, etc.
  - Still need analog for the oscillator output
  - Smaller area than full analog
  - Digital quantization limits upper frequency
- **APLL**s: fully analog, including phase detect and filtering
  - Higher accuracy than digital allows higher frequencies
  - Analog is more complicated and higher area usage

Analog PLL uses charge pump to regulate VCO voltage




Digital PLL uses DCO (DAC+VCO) to generate output

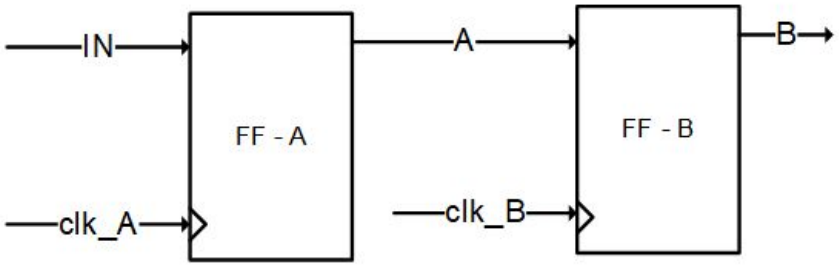


**Glossary**

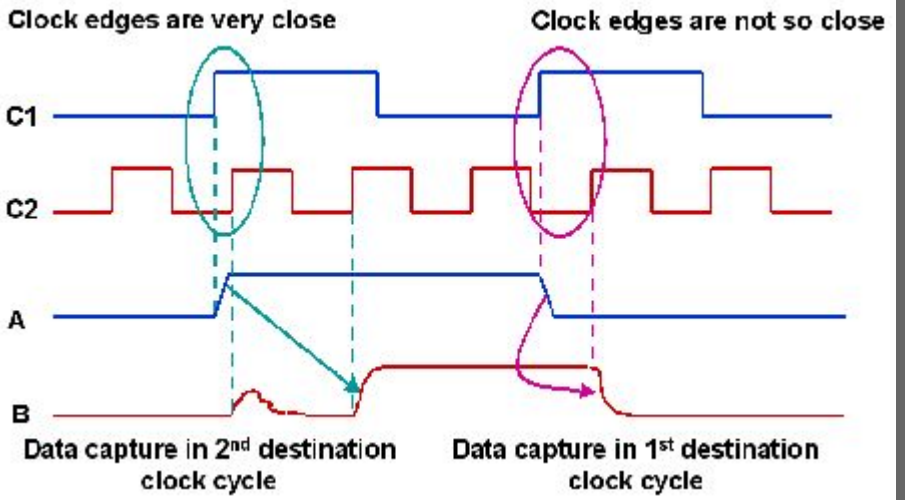
**CDC:** clock domain crossing

# Crossing Clock Domains

- Signals will have to cross between clock domains → Use a **CDC**



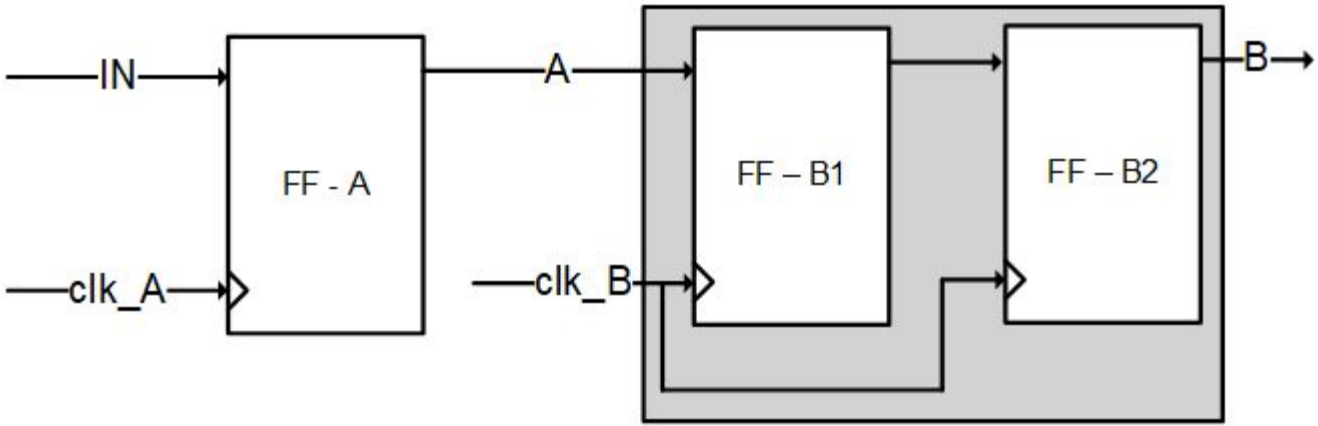

- Why not just go through a flip flop?
  - Clocks are not synchronized
  - Signal could change during destination clock transition
    - Metastability!




- There is ALWAYS a chance of metastability on a CDC
  - With only one flop, metastability will propagate through system :(

# Flip Flop Synchronizer

- 2 Flip flops highly reduces chance of metastability propagating
  - First flip flop will often be metastable, but low odds of propagating to second
  - Does not guarantee correctness!
    - In the case of metastability, the second flop will resolve to unknown value




Dual FF Synchronizer

**Glossary**

**MTBF:** mean time between failures

# Failure Rates

- Having 2 flip flops significantly metastability at the output
  - Does not eliminate it
- Chance for metastability to spread from first flop to second

$$MTBF = \frac{e^{S/\tau}}{WF_cF_d}$$

$$\begin{aligned} Where: S &= \text{Settling time (s)} \\ \tau &= \text{Settling time constant (s)} \\ F_c &= \text{Sampling clock frequency (Hz)} \\ F_d &= \text{Change frequency (Hz)} \\ W &= \text{Metastability window (s)} \end{aligned}$$

- Use **MTBF** as metric for how unlikely failures are
  - Generally, MTBF should be longer than the lifetime of the product
- Gets worse the more CDCs there are
  - With 10 CDCs and 100 year lifespan, MTBF should be 1000 years

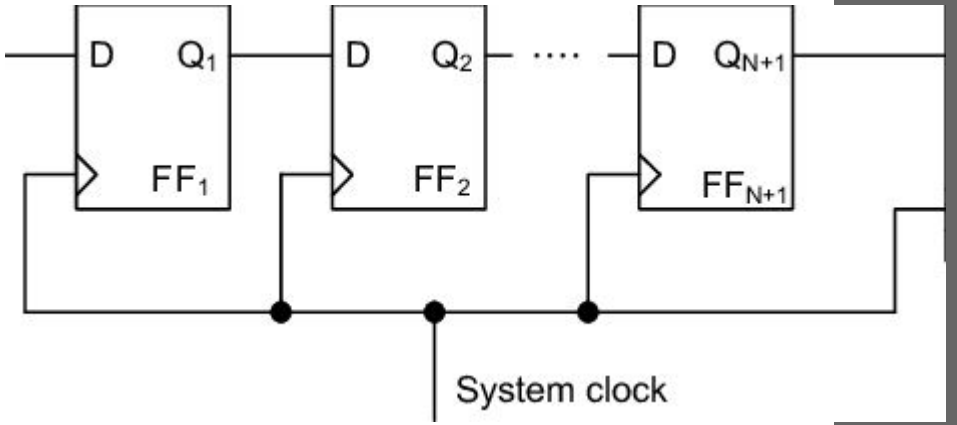
$$MTBF_{n,H} = \frac{MTBF_{n,1}}{H}$$

H = # of CDCs

# N-Flip Flop Synchronizer

- Adding more stages of flip flops significantly increases MTBF
  - At the cost of one extra cycle of latency
    - Throughput remains the same because pipelined
- Higher frequency designs require more stages
  - 2 stages is plenty for Skywater designs on the order of 100 MHz
  - High speed designs can use anywhere from 3-30 flip flops

| Stages | MTBF (s) | ln(MTBF) |
|---|---|---|
| 2 | 111.08E+72 | 170.50 |
| 3 | 154.00E+111 | 260.62 |
| 4 | 212.07E+150 | 350.74 |
| 5 | 292.85E+189 | 440.87 |



# Navigating Non-Correctness

- When metastability occurs on flop 1, flop 2's output will (usually) be 0 or 1
  - This only occurs on a transition at the input

- Two cases can occur:
  1. Get Lucky: Flop 2's output resolves to the **most recent** value at the CDC
  2. Get Delayed: Flop 2's output resolves to the **previous** value at the CDC
     a. Eventually metastability at flop 1 will end
     b. At this point, Flop 2 will update to **most recent** value

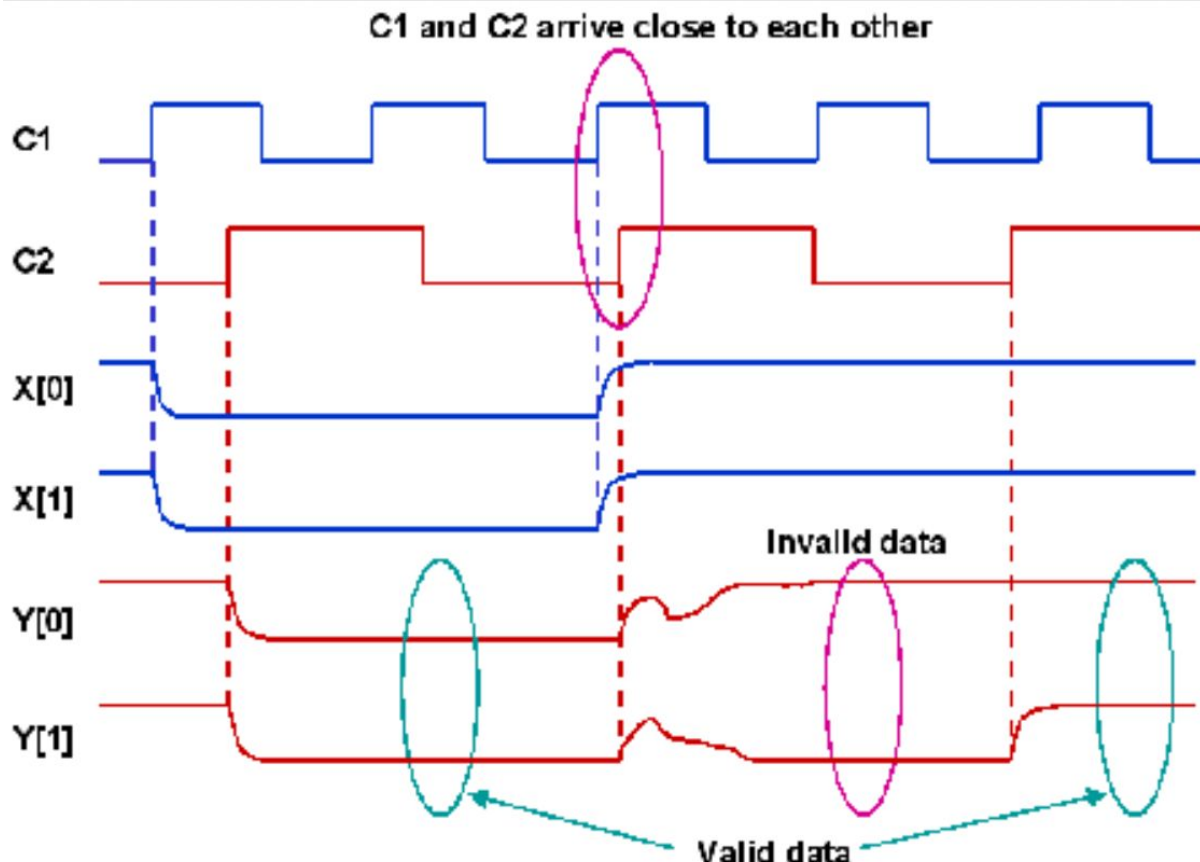
- For a **one bit signal**, an "incorrect" value only causes a delay in correctness
  - As long as CDC input signal is held, guaranteed to eventually get a correct value

# Multi-Bit Synchronization

- Correctness ended up being less of a problem for one bit signals
- Still a huge problem for multi-bit signals due to synchronization

- What if one bit is correct and another is delayed?
  - Multi Bit ⟶ Worst case is complete incorrectness
  - Single Bit ⟶ Worst case was delayed correctness

- Generally, we cannot pass arbitrary multi-bit signals thru CDC

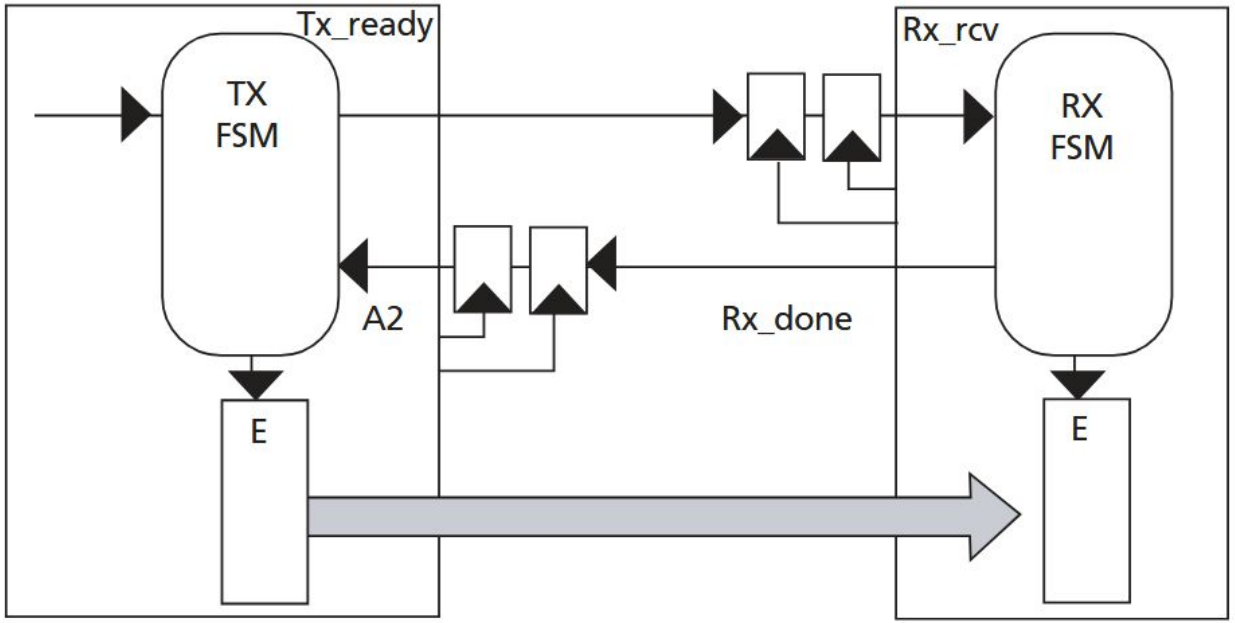


# Handshaking

- Can use single bit control signals to facilitate transfer of multi-bit data

1. Send multi-bit data without CDC and wait for it to propagate
2. Send ready signal through CDC, which will arrive later
3. Wait for response from CDC, which will arrive even later
4. Ready to stop holding data line

\+ Now we can send multi-bit data
\- High latency, low throughput due to control signal delays
\- Requires stateful FSM on either side



# Gray Encoding

**Glossary**

**Gray Encoding:** encoding scheme where one bit transitions between consecutive numbers

- Generic multi-bit signals can't go thru CDC
  - Multiple Bits transition at the same time and could get unsynchronized

- How can we encode data such that it can go thru?
  - Ensure that only one bit will transition at a time
- Gray Encoding ensures that only one bit will change for an increasing sequence
  - Only works for consecutive data, like a counter
  - The binary transition from 111 to 000 is instead 100 to 000
    - Go from n bits transitioning at once to only 1

- Can now sync counters across clock domains
  - What can we do with this?

| A B C | X Y Z |
|---|---|
| 0 0 0 | 0 0 0 |
| 0 0 1 | 0 0 1 |
| 0 1 0 | 0 1 1 |
| 0 1 1 | 0 1 0 |
| 1 0 0 | 1 1 0 |
| 1 0 1 | 1 1 1 |
| 1 1 0 | 1 0 1 |
| 1 1 1 | 1 0 0 |

# References


- https://anysilicon.com/clock-domain-crossing-cdc/
- https://web.archive.org/web/20070125072758/http://www.cadence.com/whitepapers/cdc_wp.pdf
- https://blog.abbey1.org.uk/index.php/technology/managing-mean-time-between-failure-in-xilinx-devices
- https://www.eetimes.com/understanding-clock-domain-crossing-issues/