Digital Performance: Synthesis, Power and Timing

Lecture 6

Digital Design So Far…

- Mostly discussing our RTL
  - Verilog code written at the “Register Transfer Level”
  - Behavioural description of design
- We know how to design “what it should do”, but what about…
  - How large will the device be? How much will it cost?
  - How fast will it be able to operate? Will it always behave consistently?
  - How much power does it consume? How much heat will need to be dissipated?
  - Many other considerations…
- These questions can’t be answered with behavioural description and simulation
  - Need information from synthesis

Synthesis

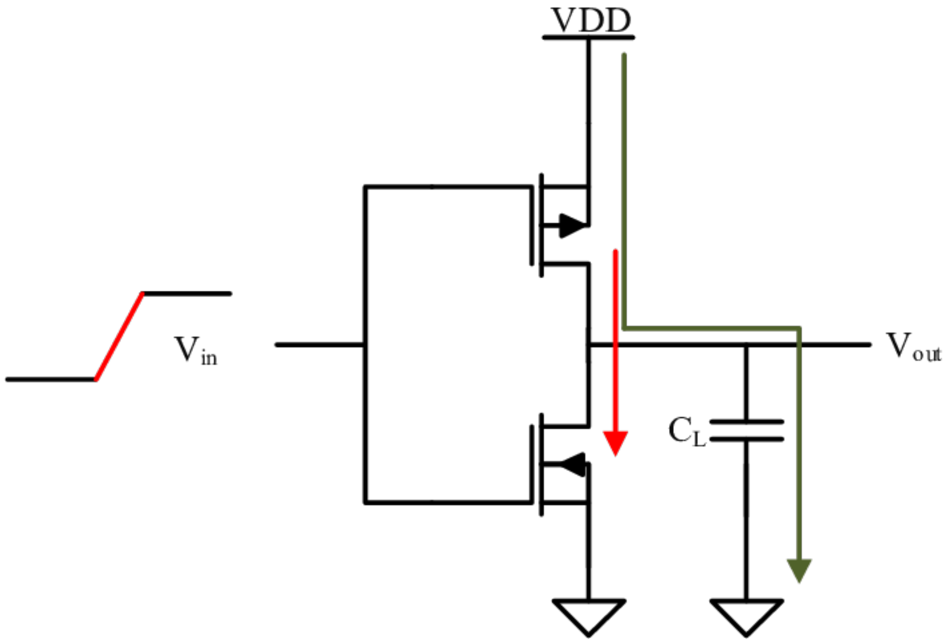
- Turns behavioural RTL description into gate-level netlist
  - Infers required digital architecture to provide desired behavior
- Chooses optimal architecture
  - ASIC synthesis might consider hundreds of different configurations for e.g. multiplier
  - FPGA synthesis will try to make best use of resources, e.g. infer DSP slice
- Optimises design
  - e.g. a multiply by 2 won't infer a full multiplier, just a shift
- Ensures chosen architecture meets timing requirements
- ASIC: Clock tree synthesis, routing (floorplanning) done at later stage
  - Don't have complete picture but knowing what cells we have in the design, we can make estimates for power & area, design for safe timing while leaving margin
- ASIC: Back-end design flow vs sign-off flow

Power

- A chip that consumes more power…
  - Costs more due to higher electricity bills
  - Reduces battery life in mobile devices
  - Produces more heat
- At advanced technology nodes, measuring and reducing power is becoming more and more important
  - Power estimation is a key part of design flow
  - An RTL designer must be able to anticipate how design choices impact power

Power Consumption in CMOS Circuits

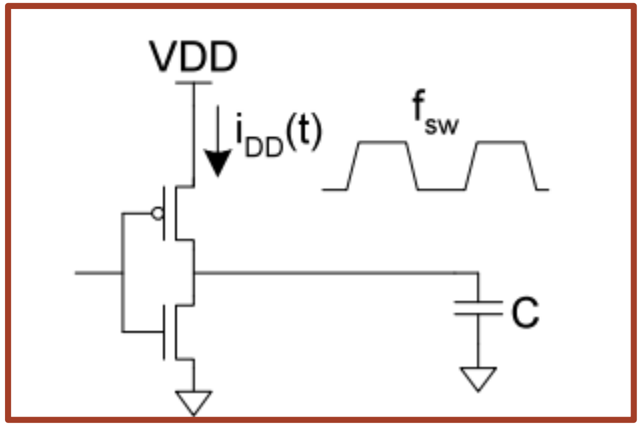
- $P_{total} = P_{dynamic} + P_{static}$
  - $P_{dynamic} = P_{switching} + P_{short\ circuit}$
- P_{static} due to leakage current
  - Determined largely by technology used
  - Worse at high temperatures
- $P_{dynamic}$ is power consumed when switch state changes
  - $P_{switching}$ from charging / discharging load C
  - $P_{short\ circuit}$ while PMOS and NMOS partially on during switch



source

Power Measurement: Dynamic Power

- Switching power is related to voltage supply, load capacitance and switching frequency
- $P_{switching} = CV_{DD}^2 f_{sw}$
- How to determine f_{sw}?
- May have real simulation data
- Otherwise, estimate with activity factor α
  - $f_{sw} = \alpha f_{clk}$
  - $\alpha = 0.5$ if f_{sw} changes once per clock cycle
- $P_{switching} = \alpha CV_{DD}^2 f_{clk}$



Power Measurement: Vectorless vs Vectored

Vectorless Power Estimation

- Can make computation from previous slide for every component in the circuit
  - Automate calculation using synthesis of course!
- Tool examines netlist to estimate load capacitance for each cell
- Assumes an average level of activity for every component (configurable)

Vectored Power Estimation

- If have real simulation data, can get real activity factor!
- More accurate than estimated
  - But slower to run
  - Typically available later in design flow
- Relies on appropriate selection of simulation data
  - Should have realistic activity levels

Low Power Design

- Low power design is becoming critical for advanced designs
  - More on this in next lecture!
- What can RTL designers do to reduce power consumption?
- Reduce α:
  - Disable block when not in use (“clock gating”, “data gating”)
  - Quantisation for arithmetic blocks
  - Consider alternative architectures, can do the same thing with fewer toggles?
- Reduce C:
  - Avoid many levels of combinational logic, or high fan-out
  - Reducing length of timing paths on ASIC means smaller cells can drive the signal
- Most importantly… Measure power and plan from the beginning!

Static Timing Analysis Review

- In pure behavioural simulations, there is zero-delay through the system
  - In reality, all cells have a delay associated with them
- A register (D flip-flop) has **setup** and **hold** time requirements
  - Data must be ready and stable T_{setup} **before** the clock edge
  - Data must be kept stable for T_{hold} **after** the clock edge
- We must consider delay across timing path, along with setup and hold time requirements, to avoid metastability
- Static Timing Analysis determines if a circuit meets timing constraints during and after synthesis

Static Timing Analysis

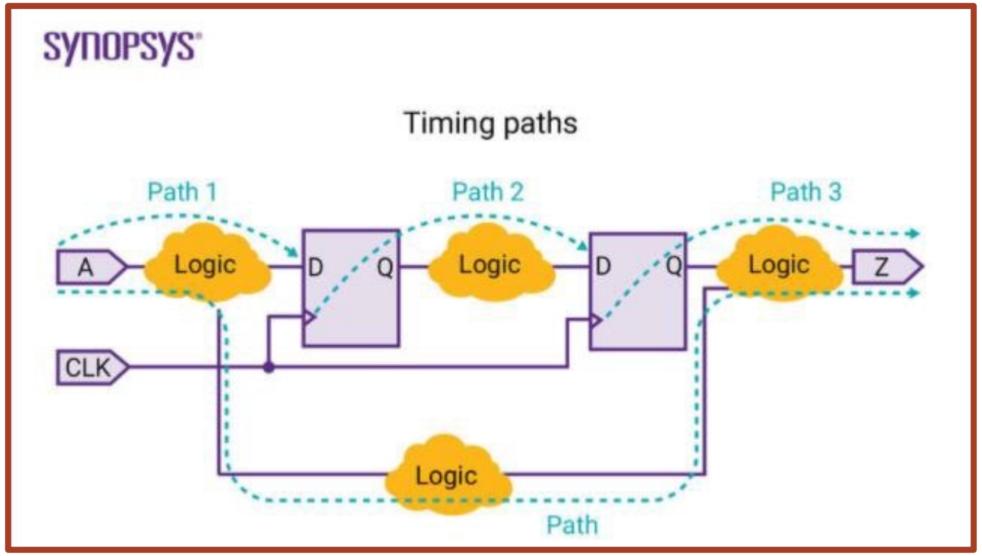
“STA is a method of validating timing performance of design by checking all possible paths for timing violations”

- STA tool follows these steps:
    1. Breaks entire design down into different **timing path**
    2. Calculates signal and clock **propagation delay** along each timing path
    3. Checks for **timing violations** of **timing constraints** along each path
- “Static”, i.e. no simulation, analysis only
- Thorough analysis, ALL timing paths are checked
- Does NOT check functionality

Timing Paths

- Start point:
  - Where data is launched, must be **input port** or **register clock pin**
- Combinational logic cloud:
  - AND, OR, XOR, NOT etc.
  - No memory (no flip flop or RAM)
  - May contain multiple possible paths between start and end points
- End point:
  - Where data is captured, must be **output port** or **register data input pin**
  - Why not end at register data output pin?

Timing Paths



https://www.synopsys.com/glossary/what-is-static-timing-analysis.html

- 4 Different types of timing paths…
  - Input ➔ Register
  - Register ➔ Register
  - Register ➔ Output
  - Input ➔ Output
- What types are paths 1-4?
- Other types of paths are
  - clock paths,
  - clock-gating paths,
  - asynchronous paths

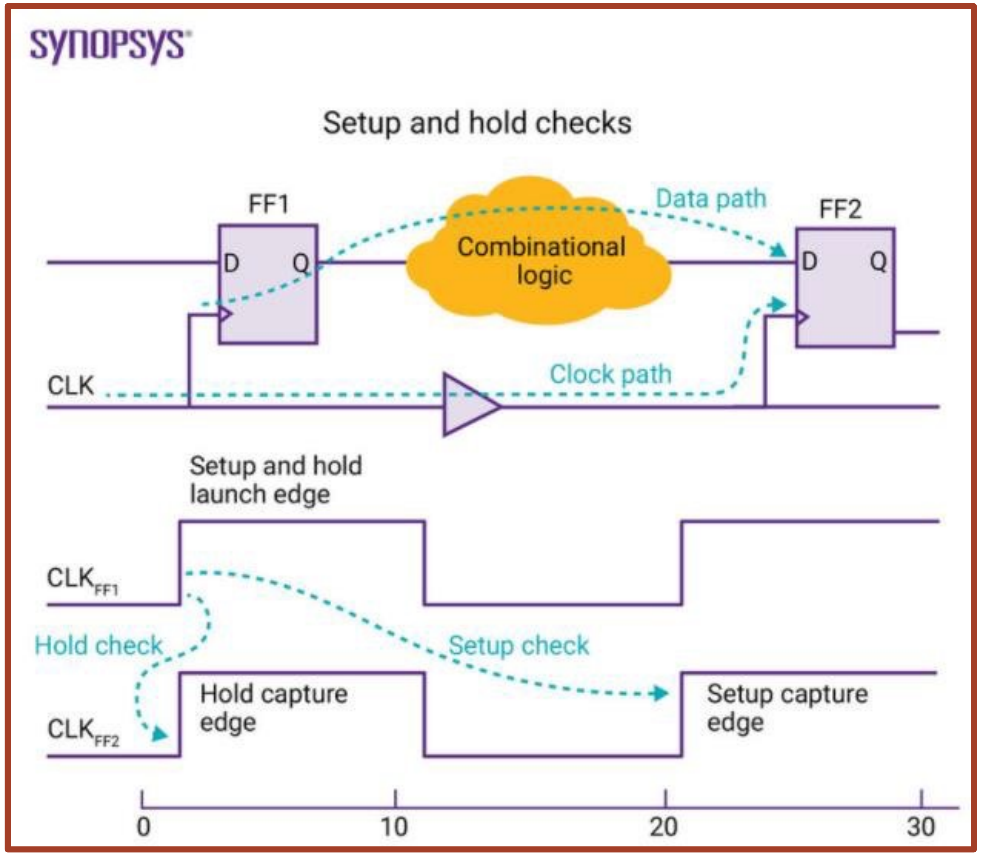
Propagation Delay

- The total delay through the timing path is the sum of the **cell delays** and **net delays**
- Cell delay
  - **Timing arc** from input port to output port of cell (logic gate or sequential cell)
  - Affected by input transition time, output load, PVT
- Net delay
  - From output of one cell to input of next cell along timing path
  - Time to charge / discharge parasitic capacitances
- Technology library contains tables of information for all cells
  - Timing arc delays, setup & hold time for flip-flops etc

Propagation Delay – Timing Arc Examples

Timing Constraints

- Setup constraint:
  - How much time does data need to be available before the clock edge
  - Specifies the **max delay** on a timing path
- Hold constraint:
  - How much time does data need to be stable after the clock edge
  - Specifies the **min delay** on a timing path
- Register setup and hold times are derived from library data



Timing Constraints – Clock Definition

- Timing checks consider data arrival and removal wrt clock edge
  - Must define the clock!
- Recall from lab assignments, Basys3_Master.xdc contains:

```
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]
```

- This tcl code creates a 100 MHz clock with 50% duty cycle, and connects it to the top-level input port named “clk”
- But there’s a lot more to clock definition!
- Constrain to allow for non-instantanous change clock paths
  - `set_clock_uncertainty`: Adds margin for error
  - `set_clock_latency`: Models delay from clock source to register clock input pin
  - `set_clock_transition`: Models the rise and fall times of clock waveform at clock input pins

Timing Constraints

- With clock defined, setup and hold checks can be performed for Register ➔ Register paths
  - What about Input ➔ Register, Register ➔ Output or Input ➔ Output?
- Must define **input delay** and **output delay**
  - Input delay is the latest data arrival time wrt clock (after launching clock edge)
  - Output delay is the latest data arrival time wrt clock (before capture clock edge)
- So we've defined our clock, input delay, output delay, and we have setup and hold times…
  - But what if we know we don't need to meet setup and hold times on some paths??

More info on setup/hold checks for each timing path type:
https://anysilicon.com/the-ultimate-guide-to-static-timing-analysis-sta/

Timing Exceptions

- False Path, if:
  - Change in source register is never going to impact or change destination register
  - OR, source and destination registers are not on the same clock domain
    - Safe clock domain crossing must be handled separately (next topic)
- Multi-cycle path
  - Change in source register won’t be needed at capture register for multiple clock cycles
  - Define a fixed number of clock periods to perform setup and hold checks over

More detail on false path, multi-cycle path:
https://www.edn.com/basics-of-multi-cycle-false-paths/

Metastability

- Ideal simulation shows instant signal transitions
  - In reality, all changes have a finite slope, what if clock arrives during change?
  - Flip-flop captures intermediate voltage, $\neq 1, \neq 0$
- Metastable FF will eventually resolve to either 0 or 1
  - Don't know which, don't know when
  - If Q is not resolved before next clock edge, metastability propagates through design!
- If source and destination registers are on asynchronous clock domains, cannot guarantee that setup times will always be satisfied
  - Metastability must be avoided using "Clock Domain Crossing" techniques

Synchronizer

- Synchronizer is a circuit used to minimize probability of metastability
- Basic sync is two FFs
  - Output of first FF is possibly metastable
  - Assumed to resolve before second clock edge
  - Only take data from second FF
- Sync often made from low V_T FF
  - Better setup/hold time
- Second FF could still be metastable…
  - Adding third FF further reduces probability

Mean Time Between Failures

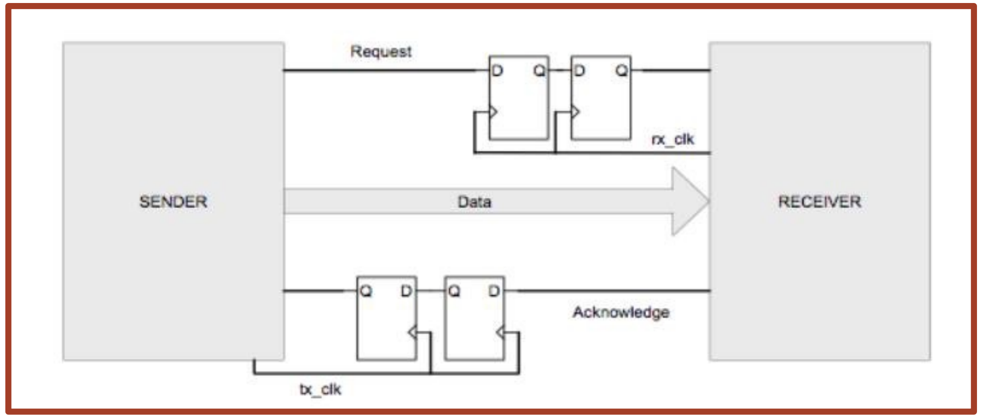
- Synchronizer does not guarantee no metastability
  - Need to know if it makes it sufficiently unlikely
- Must compute the average time between metastability events, MTBF

$$MTBF \propto \frac{1}{f_{clk} f_{data}}$$

- MTBF smaller for higher speed designs, i.e. metastability more likely
- Also affected by number of flip-flops used in sync
  - MTBF also highly process dependent

Other Clock Crossing Techniques: Handshake

- What if we want to send more than one bit?
  - Data incoherence
- What if launch clock frequency is much faster than capture clock?
  - Pulse may be missed completely
- Answer: Handshake!
  - (Or Dmux, FIFO…)
- New data to be held constant until acknowledge is received
  - Req, ack, must be synchronized



[source]

Other Clock Crossing Techniques…

- Gray code
- Asynchronous FIFOs
- DMUX
- … and many more!