ELEX 2117 : Digital Techniques 2
2022 Winter Term

Timing Analysis

This lecture describes how static timing analysis is used to ensure timing constraints for a digital design are met.
After this lecture you should be able to be able to: identify features and specifications on a timing diagram, identify a specification as a requirement or guaranteed response, apply the terms defined in this lecture, and do calculations involving clock rate, propagation delays and setup/hold time requirements.
Revised Mar. 11 - revised section on Metastability and STA, added synchronizers, removed sections on PVT corners and Timing Simulations

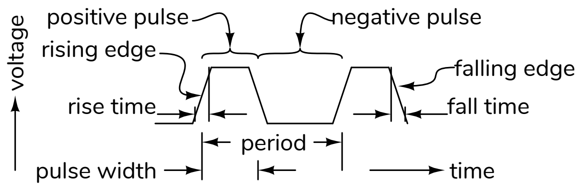
Introduction

Timing constraints are requirements that a design must meet. A typical example is that the design must operate at a specified minimum clock frequency. Meeting these constraints is often as difficult as ensuring that a design is logically correct.

A designer must correctly specify timing constraints such as clock periods and external circuit delays. This allows the design software to check if a given design will meet the device's internal timing requirements such as flip-flop setup and hold times. If not, the designer must change the design or relax the constraints in order to meet the timing requirements.

The performance of ICs varies from device to device and with changes in temperature and voltage. Building a circuit that appears to work properly is not enough to ensure that a design will work reliably. This is because the same design may fail on a different device, at a different temperature or with a different supply voltage.

Digital Waveforms



A transition from low to high is called a rising edge. The time it takes is called the rise time. A transition from high to low is called a falling edge. The time it takes is called the fall time. Two adjacent edges define a pulse, which can be negative or positive. The time between these is called the pulse width.

Rise and fall times are typically measured between 10% and 90% of the swing. Other measurements are typically made between 50% levels. But there are exceptions.

A signal consisting of periodic pulses is a clock. The inverse of the clock period is the clock frequency.

Timing Specifications

Timing specifications can be:

requirements: the manufacturer requires that these specifications be met for a device to operate properly, or

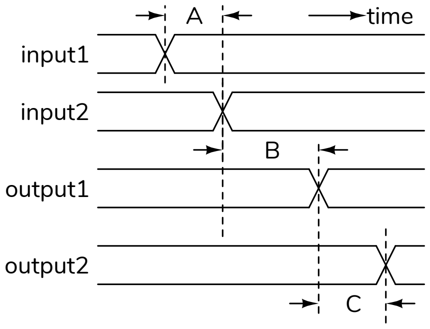
guaranteed responses: the manufacturer guarantees that these specifications will be met.

Since the device manufacturer cannot control the timing on inputs but can guarantee the timing of outputs, a simple rule to distinguish requirements from responses is:

- requirements are measured *ending on a transition on an input.*
- guaranteed responses are measured *ending on a transition on an output.*

Timing diagrams show the relationship between transitions on inputs and outputs. However, they are not drawn to scale and transitions may not always happen in the order shown.

Other conventions used in timing diagrams include: both high and low levels are shown when the specification applies to both, shading between two levels indicates that the value is allowed to change during this time, a line half-way between the two logic levels indicates that the signal is in high-impedance ("tri- state") state and arrows drawn between transitions show that one signal transition causes or affects another.

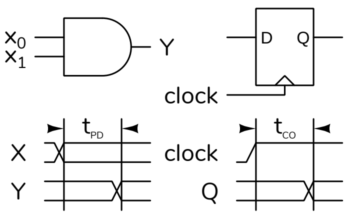


Exercise 1: Label the specifications A through C as requirements or guaranteed responses.

Propagation Delays

Combinational logic circuits and D flip-flops have one important timing specification: their propagation delay. This is the delay from a change on an input to the corresponding change on an output. The usual symbols are t_{PD} for propagation delay and t_{CO} for the clock-to-output delay for a D flip-flop.

Propagation delay, t_{PD}, is the delay from a change at an input to the corresponding change at an output. The clock-to-output delay, t_{CO}, a type of propagation delay, is the delay from the rising edge at a flip-flop clock input to the change at the Q output.

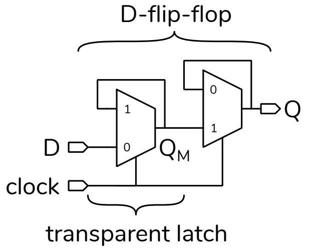


These delays are primarily caused by the time required to charge the parasitic capacitances of transistors and the metallic "interconnects" that connect them.

Exercise 2: Is t_{PD} a requirement or a guaranteed response?

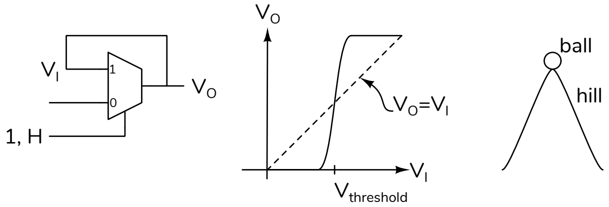
Metastability, Setup and Hold Times

Consider the following implementation of an edge-triggered D flip-flop:

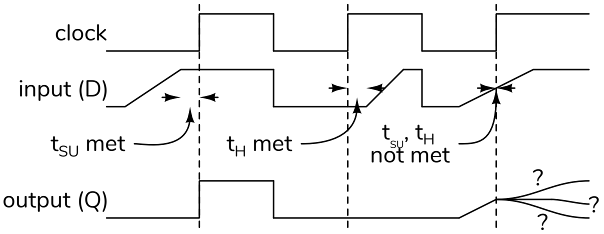


When the clock input is low, the output of the first multiplexer follows the input – the latch is "transparent". When the clock input is high, the output level is fed back to the input and held at that level[1].

If the D input is near the logic threshold voltage when the clock changes from 0 to 1 – the rising edge – the multiplexer might not be able to decide whether to output a 0 or 1.



The multiplexer output (Q_M) could remain at at the threshold level for a long time – longer than the t_{CO} specification. This behaviour is called "metastability"[2] and can result in incorrect operation of the circuit.



To avoid metastability we must ensure the voltage at the latch input is at valid level long enough to drive the latch output to a valid voltage level before the rising edge of the clock. The time required for this is called the "setup" time, t_{SU}.

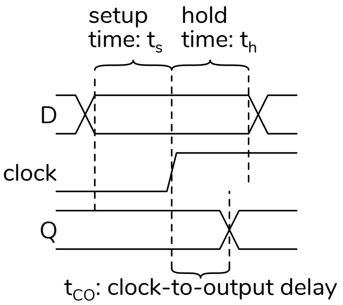
The input level must also be held at the correct level until the transparent latch has finished switching to the feedback mode. This is typically a much

[1] This is a "master-slave" flip-flop. The second, "slave," latch holds the previously latched value when the clock is 0

[2] The output is "meta" stable because it appears to be stable at a level that is not one of the true stable states (H or L).

shorter time – typically zero – and is called the "hold" time, t_{H}.

D flip-flops thus have two important timing requirements:



setup time the D input must be at a valid logic level for at least t_{S} before the rising edge of the clock

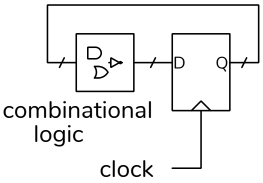
hold time the D input must remain a valid logic level for at least t_{H} after the rising edge of the clock

Exercise 3: Is t_{SU} a requirement or a guaranteed response? How about t_{H}?

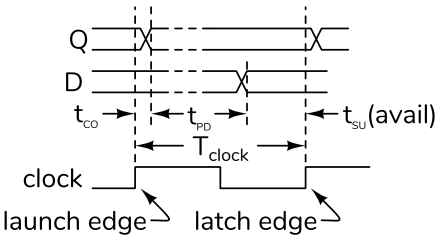
Synchronous Design and Static Timing Analysis

A circuit that uses only edge-triggered flip-flops and a single clock is called a "synchronous" circuit. These are universally used because they allow us to avoid metastable behaviour.

Any logic circuit that uses a single clock can be redrawn as one register whose inputs are driven by a combinational logic function of its outputs:



The timing diagram below shows the relationship between two adjacent clock edges, the output (Q) and input (D) of the register:



Q becomes valid t_{CO} after the first ("launching") rising clock edge. After t_{PD}, the propagation delay through the combinational logic, D will have a correct and valid logic level.

In most cases the timing requirement that is most difficult to meet is the minimum setup time.

From the timing diagram above we can write an expression for the minimum available setup time before the next ("latching") rising clock edge. :

$$t_{\text{SU}}\,(\text{avail}) = T_{\text{clock}} - t_{\text{CO}}\,(\text{max}) - t_{\text{PD}}\,(\text{max})$$

T_{clock} is a design choice. t_{CO} (max) is the maximum delay specified by the manufacturer.

Exercise 4: Which of the specifications in the formula above decrease the available setup time as they increase? Which increase it?

Each logic function (multiplexer, adder, gate, etc.) between the output and input of a flip-flop increases t_{PD} for that path. Static timing analysis computes t_{PD} for every possible path between any flip-flop output and any flip-flop input and uses the path with the longest delay to compute the maximum propagation delay, t_{PD}(max).

The amount by which the minimum available setup time exceeds the minimum required setup time is known as the "slack":

$$\text{slack} = t_{\text{SU}}\,(\text{available}) - t_{\text{SU}}\,(\text{required})$$

If the slack is positive then the available setup time exceeds the required value and the t_{SU} requirement is met, otherwise it is not and the circuit may not operate correctly due to metastable behaviour: excessively long clock-to-output delays.

Exercise 5: For a particular circuit f_{clock} is 50 MHz, t_{CO} is 2 ns (maximum), the worst-case (maximum) t_{PD} in a circuit is 15 ns and the minimum setup time requirement is 5 ns. What is the setup time slack? Will this circuit operate reliably? If not, what it the maximum clock frequency at which it will?

Exercise 6: What is the maximum clock frequency for a counter using flip-flops with 200 ps setup times, 50 ps hold times and adder logic that has a 250 ps propagation delay?

Note that the clock signal itself may have a propagation delay and may arrive at different flip-flops at different times. This is known as "clock skew" and must also be taken into account in the analysis.

SDC Timing Constraints

Timing and other design constraints are provided to the design software by a file written using the "Synopsys Design Constraint" (SDC) syntax. The only SDC command we will cover in this course is the clock frequency constraint. An example is:

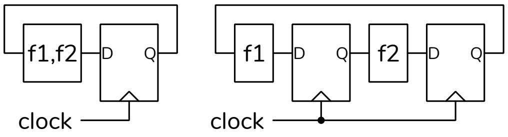
```
create_clock -name clk50 -period 20
   [get_ports {clk50}]
```

which specifies a clock named `clk50` at a port named `clk50` that has a 20 ns period[3]. The SDC format allows for many additional constraints such as external propagation delays (e.g. `set_input_delay`) and signals that should be ignored by the timing analysis (`false_path`).

Closing Timing

"Closing" timing is the process of iterating a design until all paths have positive slack. If design does not meet its timing requirements we can:

- Change the design to reduce the delay through critical timing paths. This might mean having more logic in parallel or "pipelining" – dividing up the computation across multiple clock cycles:



- Use a faster device – one with lower t_{PD} and/or lower t_{SU}.
- Relax the design constraints by reducing the required clock rate (thus increasing T_{clock}).
- Reduce the interconnect delays. We can have the EDA software spend more time ("effort") optimizing the routing between flip-flops or use a larger PLD to allow layouts with lower delays.

the choice will depend on the project requirements and available resources.

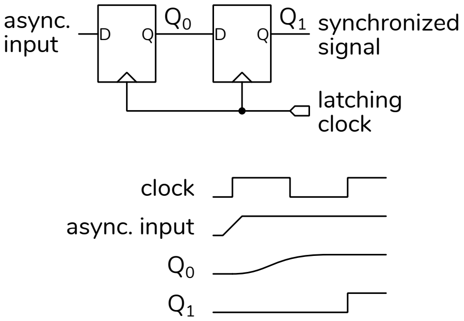
Exercise 7: Which of the above would increase design time? Which would increase the unit costs? Which would lower quality?

Asynchronous Clocks and Inputs

If all clocks are derived from the same source clock (e.g. through clock division or using a PLL) the time relationships between clocks remains constant and it's possible to verify that timing constraints will be met.

However, if two clocks are physically independent then this is not possible – the clock edges will drift relative to each other and the setup and hold timing requirements of flip-flops using different clocks (those in different "clock domains") are bound to be violated eventually. Even though it's not possible to do timing analysis for asynchronous signals, it is possible to estimate the mean time between failure (MTBF) due to metastable events when signals cross clock "domains."

The most common way to increase the MTBF is to use two flip-flops in series to build a "synchronizer":



The use of two flip-flops with no intermediate delays allows for a settling time approximately equal to one clock cycle. This is usually sufficient to ensure a sufficiently high MTBF at the output of the second flip-flop. However, if necessary, additional flip-flops can be placed in series.

An unavoidable consequence of using a synchronizer is that one (latch) clock cycle of delay is added to the signal.

The probability of a metastable event increases proportionately with the product of the clock rates. For slow events, such as button presses, this probability will be negligible.

[3] Note that this does not change the clock frequency of your circuit – that is determined by the oscillator on your board.