Static 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 understand the terms defined in this lecture and compute setup and hold slack from a delay-annotated schematic.

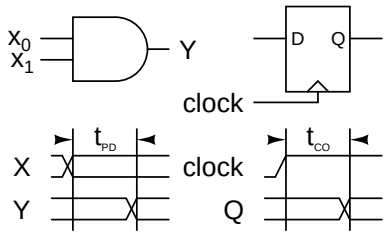
Introduction

Timing "constraints," are requirements such as the clock rate or setup and hold specifications. Meeting these constraints is often as difficult as ensuring a design is logically correct.

Reliable operation requires that the designer correctly specify the timing constraints and modify the design until they are met.

Propagation Delays

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

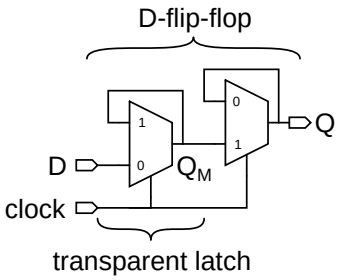


These delays are caused by the time required to charge the parasitic capacitances of transistors and interconnects.

In the timing diagrams above the parallel lines indicate times during which is signal is held at a high or low level. The crossing lines indicate the times at which the signal changes.

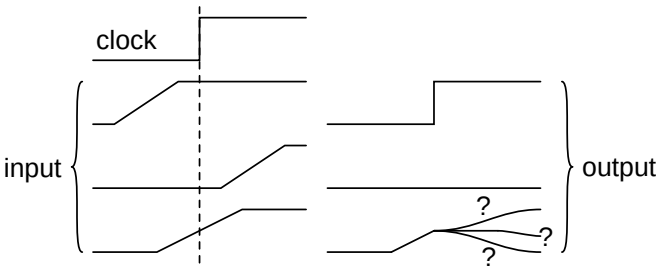
Metastability, Setup and Hold Times

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



When the clock input is 0, the output of the first multiplexer follows the input – the latch is "transparent". When the clock input is 1, the output level is fed back to the input and held at that level[1].

If the latch output is at the logic threshold voltage when the clock change from 0 to 1 then the multiplexer might not be able to decide whether to feed back a 0 or 1. The multiplexer output could remain at an invalid level for much longer than t_{CO}. This behaviour is called "metastability" and can result in unreliable circuit operation.



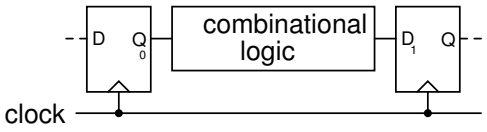
To avoid metastability we must ensure the voltage at the latch input is at valid level long enough to drive the latch output to a valid voltage level. The time required for this is called the "setup" time, t_{SU}.

The input level must also be held at the correct level until the multiplexer has switched off completely. This is typically a much shorter time – often zero – and is called the "hold" time, t_H.

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

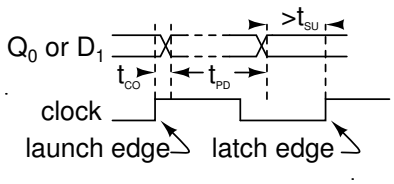
Synchronous Design

To avoid metastability almost all digital circuits are "synchronous." These circuits are composed of edge-triggered flip-flops with combinational logic between their outputs and inputs:



By ensuring the propagation delays through the combinational logic will meet the setup and hold requirements we can avoid metastable behaviour.

The timing diagram below shows the relationship between the clock edges and the valid times at the inputs and outputs of each flip-flop:



Q changes t_{CO} after the rising clock edge. t_{PD} later the input at the D input of the right flip-flop will have a valid (and correct) logic level. This must happen t_{SU} at the latest before the next rising edge of the clock. This level must also be held for at least t_H before it changes.

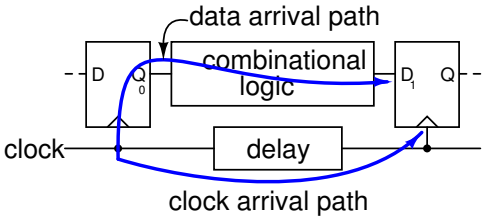
The diagram above identifies two clock edges, the "launch" and "latch" edges. In this example the edges are separated by the clock period. However, the clocks may arrive at different times due to different interconnect delays. This is known as "clock skew." It's also possible that the two clocks have different frequencies or latch on the falling edge.

This setup time is often called a "library" or "micro" setup time to distinguish it from the chip I/O setup and hold times.

Static Timing Analysis

Timing Paths

To avoid metastability we must compare the propagation delay along the data path to the propagation delay along the clock path as shown below:



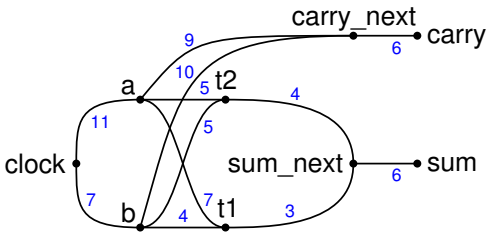
The time from the launch edge to the data arrival at the D flip-flop input is called the data available time. The time at which the latch clock edge arrives is called the clock arrival time. The delays included in calculating these times include interconnect delays, t_{CO}, and t_{PD}

Timing Netlists

It is important to note that the only timing paths that need to be analyzed are those that start at a clock input (or a chip input pin, called an "input port") and end at the D input of a flip-flop (or a chip output port).

However, there may be more than one path from a clock to a particular D input. For example, consider the half-adder shown in Figure 1. The numbers next to input pins are the delays from that input to the output, including interconnect delay to that input[2]

The data structure used for STA is a directed (acyclic) graph where each node represents a pin. Edges represent delays and are labelled with the propagation delay (including both gate and interconnect). In the following graph each node represents an output and the values on the edges represents the delays:



In this example there is only one clock, clock, and so all paths start at clock. There are four flip-flops but we will limit our analysis to carry and sum for now.

The sums of the delays along the data paths, working from output to input, are:

[2]These are not the real values, I'm using round numbers to make the arithmetic easier.

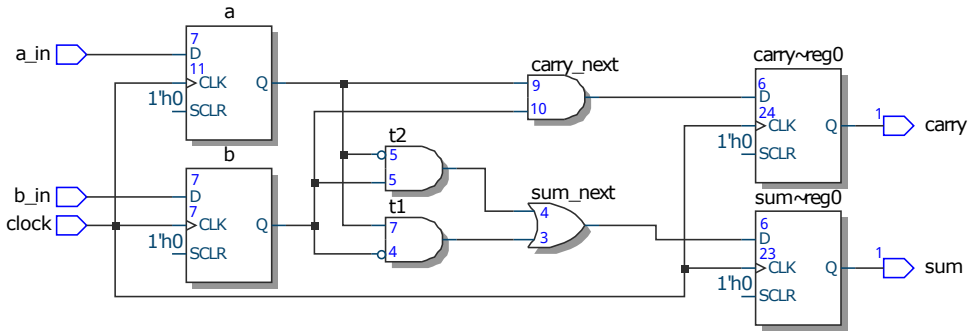
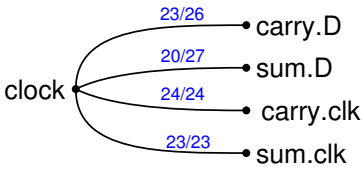


Figure 1: Delay-annotated half-adder schematic.

```
carry.D (6) + carry_next (9) + a.clk (11) = 26

carry.D (6) + carry_next (10) + b.clk (7) = 23

sum.D (6) + sum_next (4) + t2 (5) + a.clk (11) = 26

sum.D (6) + sum_next (4) + t2 (5) + b.clk (7) = 22

sum.D (6) + sum_next (3) + t1 (7) + a.clk (11) = 27
```

Exercise 1: What is the remaining path and delay? What are the clock path delays?

For the purposes of timing analysis we only need to find the path with the minimum and the path with the maximum delay between each clock to D-input path and each clock to clock input path. For the `carry` flip-flop the data path delays are 23 (min) and 26 (max). For the `sum` flip-flop these are 20 (min) and 27 (max). This reduces the graph to:



where the pairs of numbers are the minimum and maximum delays on each path.

STA Algorithm

A static timing analyzer finds the timing paths and min/max delays from a delay-annotated netlist, computes the time difference between clock and data arrival times and checks that the corresponding setup and hold requirements are met.

The minimum clock arrival time and maximum data arrival time are used when computing the setup time:

$$t_{SU} = t_{\text{clock arrival}}\ (\text{min}) - t_{\text{data arrival}}\ (\text{max})$$

and the max data arrival and minimum clock arrival times are used when computing the hold time:

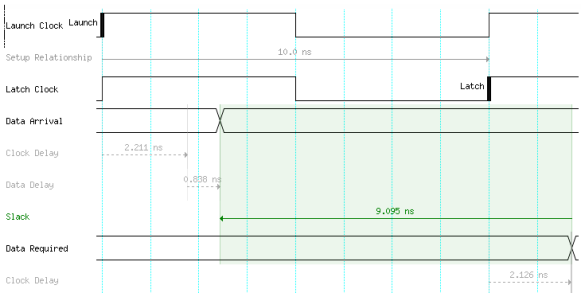
$$t_{H} = t_{\text{data arrival}}\ (\text{min}) - t_{\text{clock arrival}}\ (\text{max})$$

The setup and hold times on each timing path are then compared to the required setup and hold times. The difference is called the "slack." A positive slack means the requirement is exceeded.

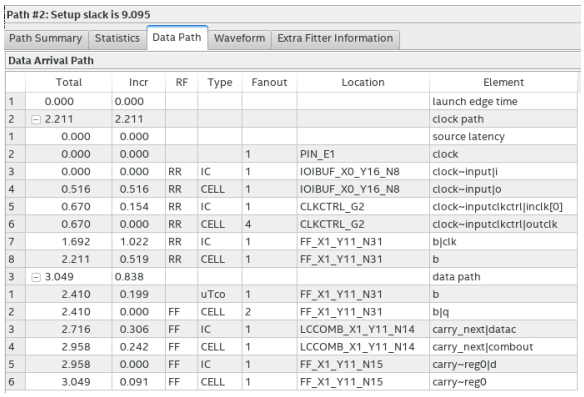
Since each clock has many launch and latch edges, the STA must pick an appropriate pair. The rule is to use the latch edge immediately following the launch edge when computing the setup time and to use the latch edge immmediately before the launch edge when computing the hold time.

Exercise 2: Use numbers in the graph above to compute the setup time slack for `carry` if the clock period is 10 ns.

The following screen capture from Time Quest, the Intel FPGA STA tool, shows the clock and data waveforms used to compute of the setup time along the path from the clock input to the `carry` flip-flop. Note the two slightly different clock delays and the term "Data Required Time" which includes the setup time.



The following screen capture shows how the data arrival time is computed by adding up the various propagation delays along the path:



Path #2: Setup slack is 9.095

Path Summary | Statistics | Data Path | Waveform | Extra Fitter Information

Data Arrival Path

	Total	Incr	RF	Type	Fanout	Location	Element
1	0.000	0.000					launch edge time
2	2.211	2.211					clock path
1	0.000	0.000					source latency
2	0.000	0.000			1	PIN_E1	clock
3	0.000	0.000	RR	IC	1	IOIBUF_X0_Y16_N8	clock~input|i
4	0.516	0.516	RR	CELL	1	IOIBUF_X0_Y16_N8	clock~input|o
5	0.670	0.154	RR	IC	1	CLKCTRL_G2	clock~inputclkctrl|inclk[0]
6	0.670	0.000	RR	CELL	4	CLKCTRL_G2	clock~inputclkctrl|outclk
7	1.692	1.022	RR	IC	1	FF_X1_Y11_N31	b|clk
8	2.211	0.519	RR	CELL	1	FF_X1_Y11_N31	b
3	3.049	0.838					data path
1	2.410	0.199		uTco	1	FF_X1_Y11_N31	b
2	2.410	0.000	FF	CELL	2	FF_X1_Y11_N31	b|q
3	2.716	0.306	FF	IC	1	LCCOMB_X1_Y11_N14	carry_next|datac
4	2.958	0.242	FF	CELL	1	LCCOMB_X1_Y11_N14	carry_next|combout
5	2.958	0.000	FF	IC	1	FF_X1_Y11_N15	carry~reg0|d
6	3.049	0.091	FF	CELL	1	FF_X1_Y11_N15	carry~reg0

In this example the clock period is 10 ns and the setup slack on this path is about 9 ns.

Closing Timning

"Closing" timing is the process of iterating a design until all paths have positive slack. There are various options when a design does not meet its timing requirements:

- ask the EDA software to spend more time (effort) optimizing the layout and routing
- use a larger or faster device or process – this makes it easier to optimize PAR
- modify the design to speed up critical timing paths. This might mean having more logic in parallel or dividing up the computation into more clock cycles.
- relax the design constraints (e.g. reduce the clock rate)

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

PVT and Corners

The propagation delays on one die will depend on the temperate and voltage. There will also be differences between die due to process differences. STA should be repeated using delays for the expected "PVT" (Process, Voltage, Temperature) extremes. The PVT combination that results in the maximum or minimum delays is called a "corner."

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 setup and hold timing requirement of flip-flops with asynchronous inputs are bound to be violated at some point. Even though it's not possible to do timing analysis on asynchronous signals, it is possible to determine how often timing violations happen when signals cross clock "domains" and the consequences. This topic will be covered in more detail later.

Timing Simulations

A timing-annotated netlist can be used by a simulator to run simulations that take into account delays. During the simulation the simulator can check that the setup and hold requirements of each flip-flop are met.

The advantage of this "dynamic" timing analysis is that the verification results are independent of, and can serve as a check on, user-provided timing constraints. The disadvantage is that the simulation may not cover all possible events. Timing simulations can be time-consuming for large designs and are primarily used for ASIC "sign-off."