



LUND
UNIVERSITY

EITF35: Introduction to Structured VLSI Design

Part 2.2.1: Sequential circuit

Liang Liu
liang.liu@eit.lth.se



Outline

- **Sequential vs. Combinational**
- **Synchronous vs. Asynchronous**
- **Basic Storage Elements**
- **Timing**
- **Folding & Pipeline**



Sequential vs. Combinational

□ A combinational circuit:



□ At any time, outputs depend only on inputs

- Changing inputs changes outputs

□ No regard for previous inputs

- No memory (history)

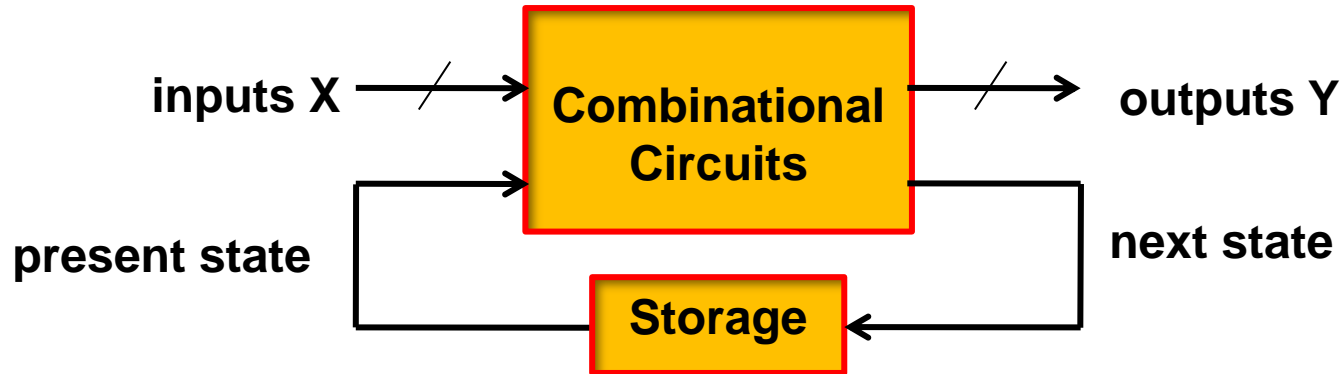
□ Time is ignored !

- Time-independent circuit



Sequential vs. Combinational

□ A sequential circuit:



□ Outputs depends on inputs and **past history** of inputs

- Previous inputs are stored as binary information into **storage elements**

□ A combinational circuit with **storage element**

- The stored information at any time defines a state

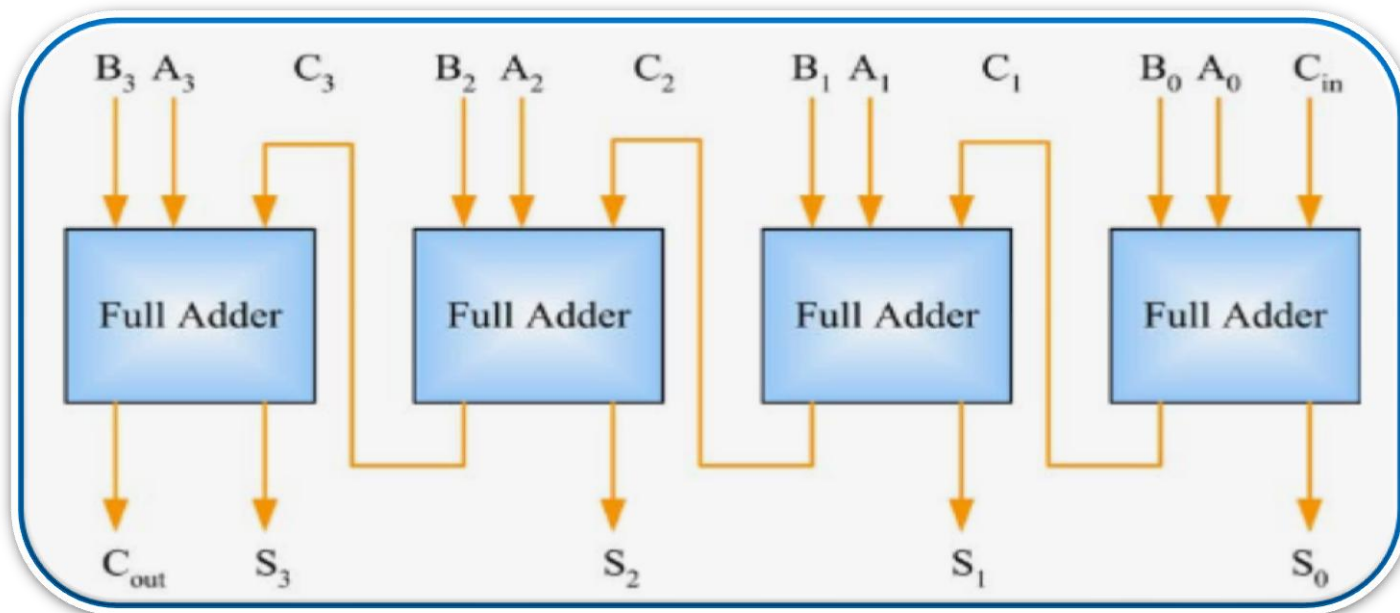


Sequential vs. Combinational: adders

□ Calculate $A_3A_2A_1A_0 + B_3B_2B_1B_0$

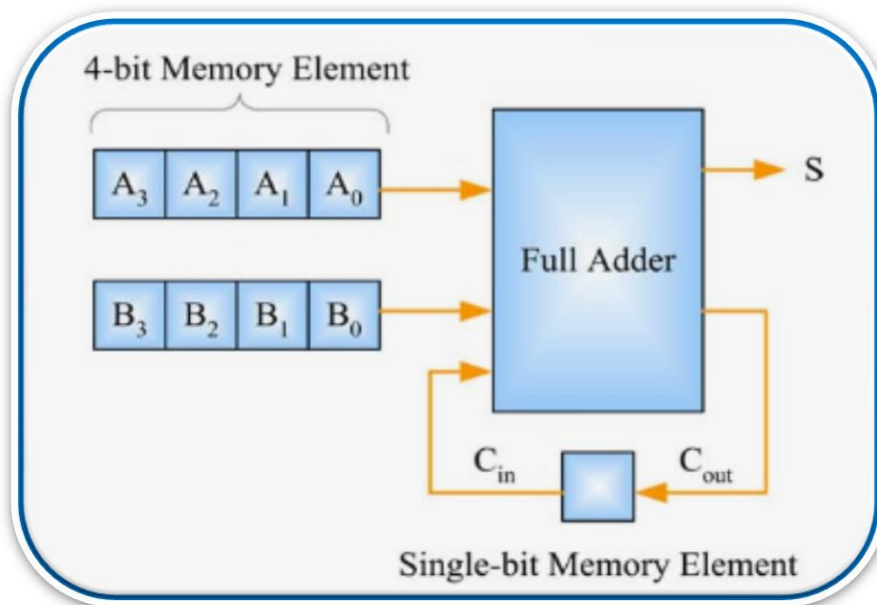
□ Combinational adder

- 4 full adders are required
- One adder is active at a time slot



What we can do with memory?

□ Sequential Adder



□ Folding!

- One full adder
- 1-bit memory for carry
- Two 4-bit memory for operators

□ 4 clocks to get the output



Outline

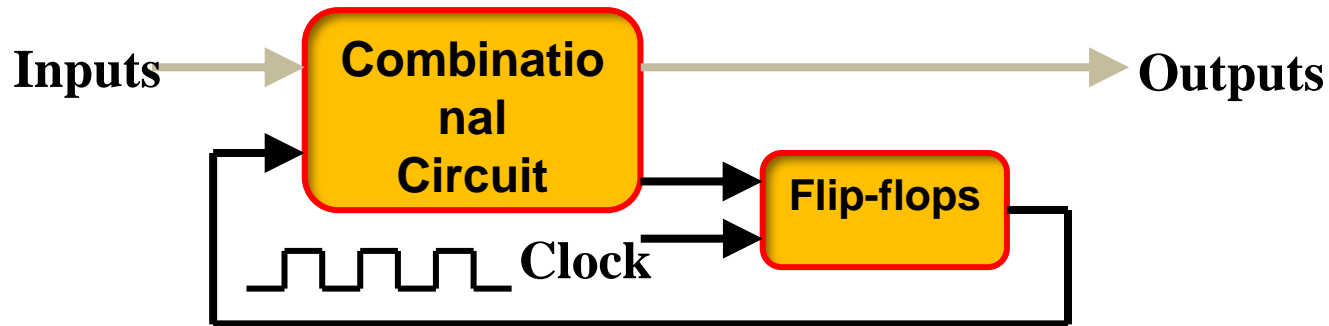
- Sequential vs. Combinational
- **Synchronous vs. Asynchronous**
- Basic Storage Elements
- Timing
- Folding & Pipeline



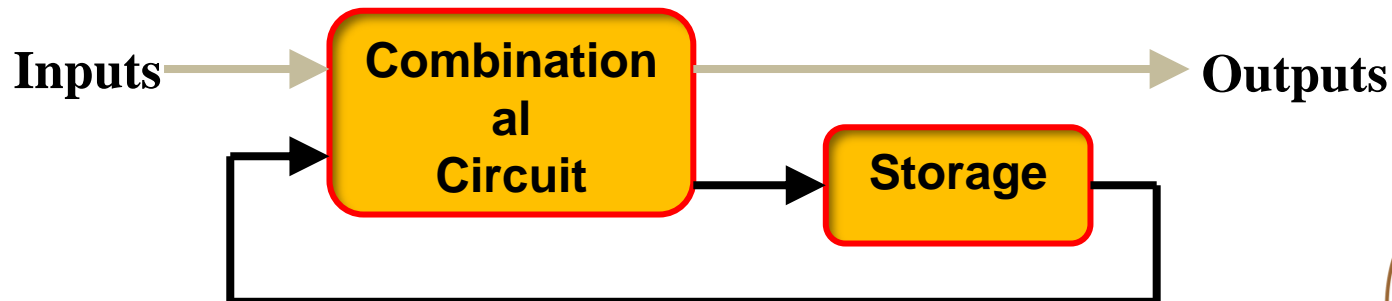
Synchronous vs. Asynchronous

Two types of sequential circuits:

- **Synchronous**: The behavior of the circuit depends on the input signal at discrete instances of time (also called **clocked**)



- **Asynchronous**: The behavior of the circuit depends on the input signals at **any instance of time**



Synchronous or Asynchronous?

□ Sync. Advantages: Simplicity to design, debug, and test

- Timing is controlled by one simple clock
- No hand-shake circuits
- Well supported by EDA tools
- Clock-gating to save power
- Recommended for **VLSI**

□ Sync. Disadvantages:

- Performance constrained by worst-case: critical path
- Overhead for clock network
- Less power efficient

We will focus on synchronous circuits in this course



Outline

- Sequential vs. Combinational
- Synchronous vs. Asynchronous
- **Basic Storage Elements**
- Timing
- Folding & Pipeline



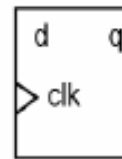
Basic storage element

- D latch: level sensitive
- D flip-flop (D-FF): edge sensitive



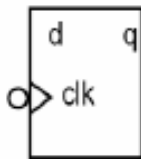
c	q*
0	q
1	d

D latch



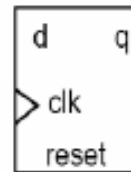
clk	q*
0	q
1	q
\uparrow	d

pos-edge triggered D-FF



clk	q*
0	q
1	q
\downarrow	d

neg-edge triggered D-FF



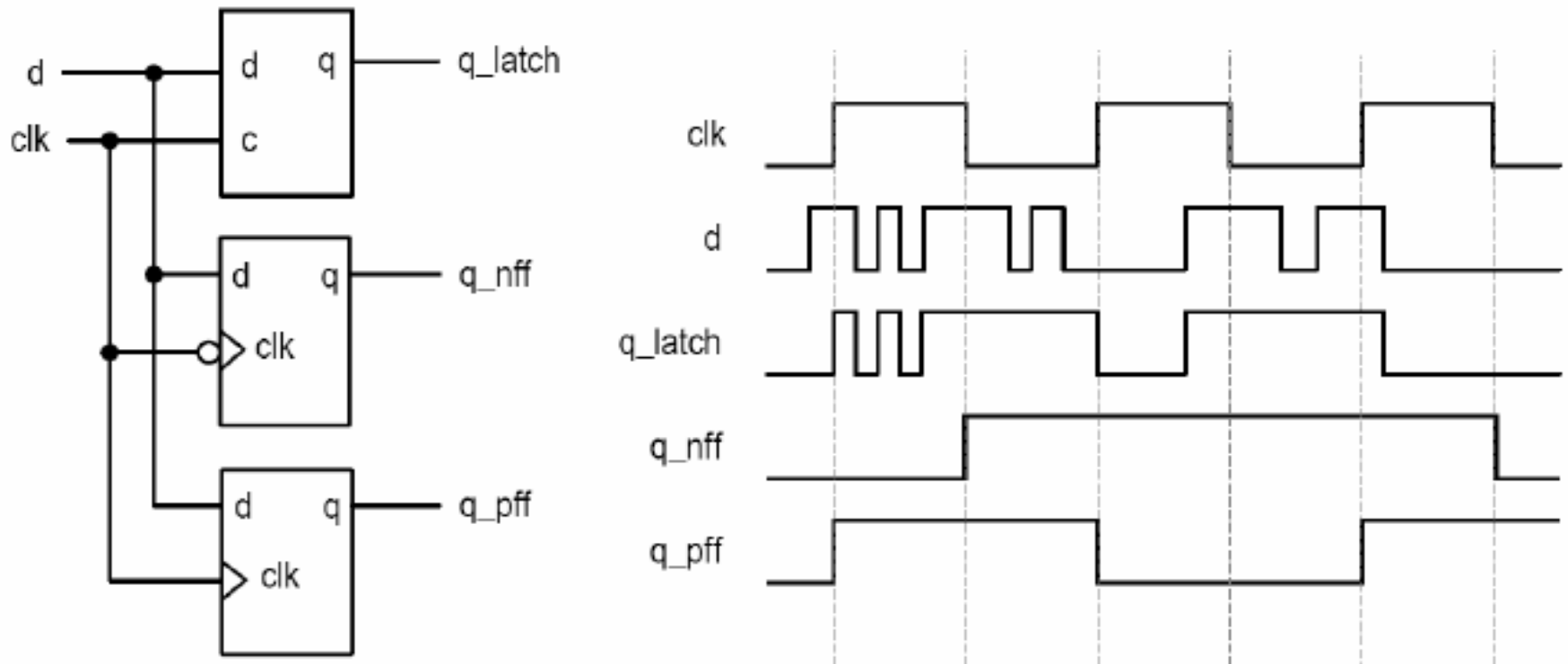
reset	clk	q*
1	-	0
0	0	q
0	1	q
0	\uparrow	d

D-FF with reset



Basic storage element

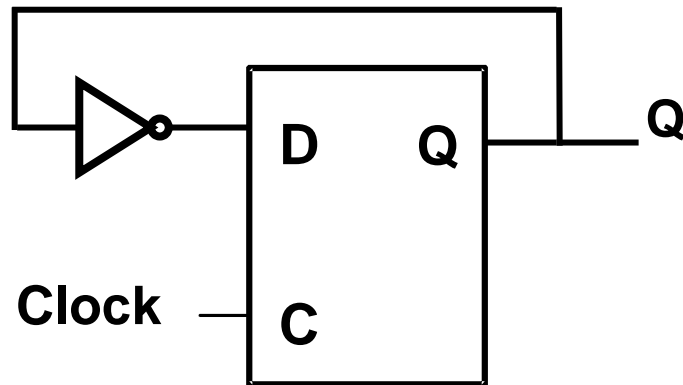
- D latch: level sensitive
- D flip-flop (D-FF): edge sensitive



Problem with Latches

- ❑ Problem: A latch is transparent; state keep changing as long as the clock remains active
- ❑ Due to this uncertainty, latches can not be reliably used as storage elements.
- ❑ What happens if Clock=1? What will be the value of Q when Clock goes to 0?

Example



Outline

- Sequential vs. Combinational
- Synchronous vs. Asynchronous
- Basic Storage Elements
- Timing**
- Folding & Pipeline

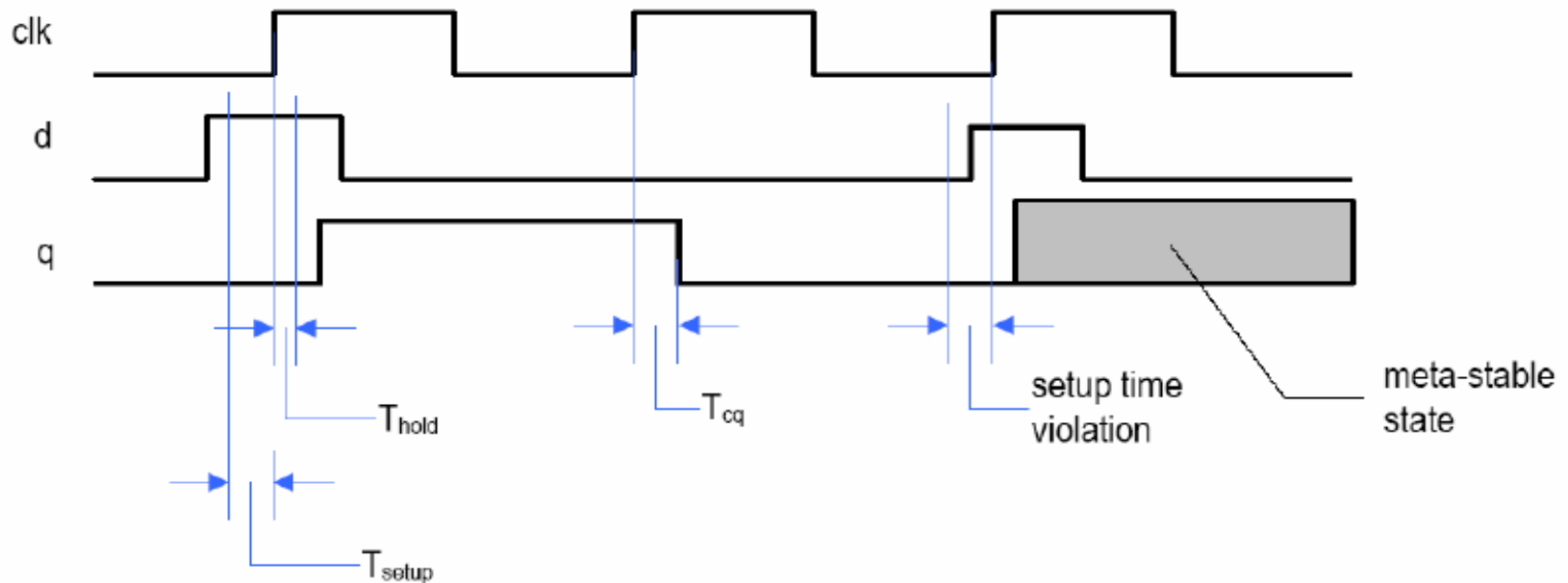


Flip Flops Timing

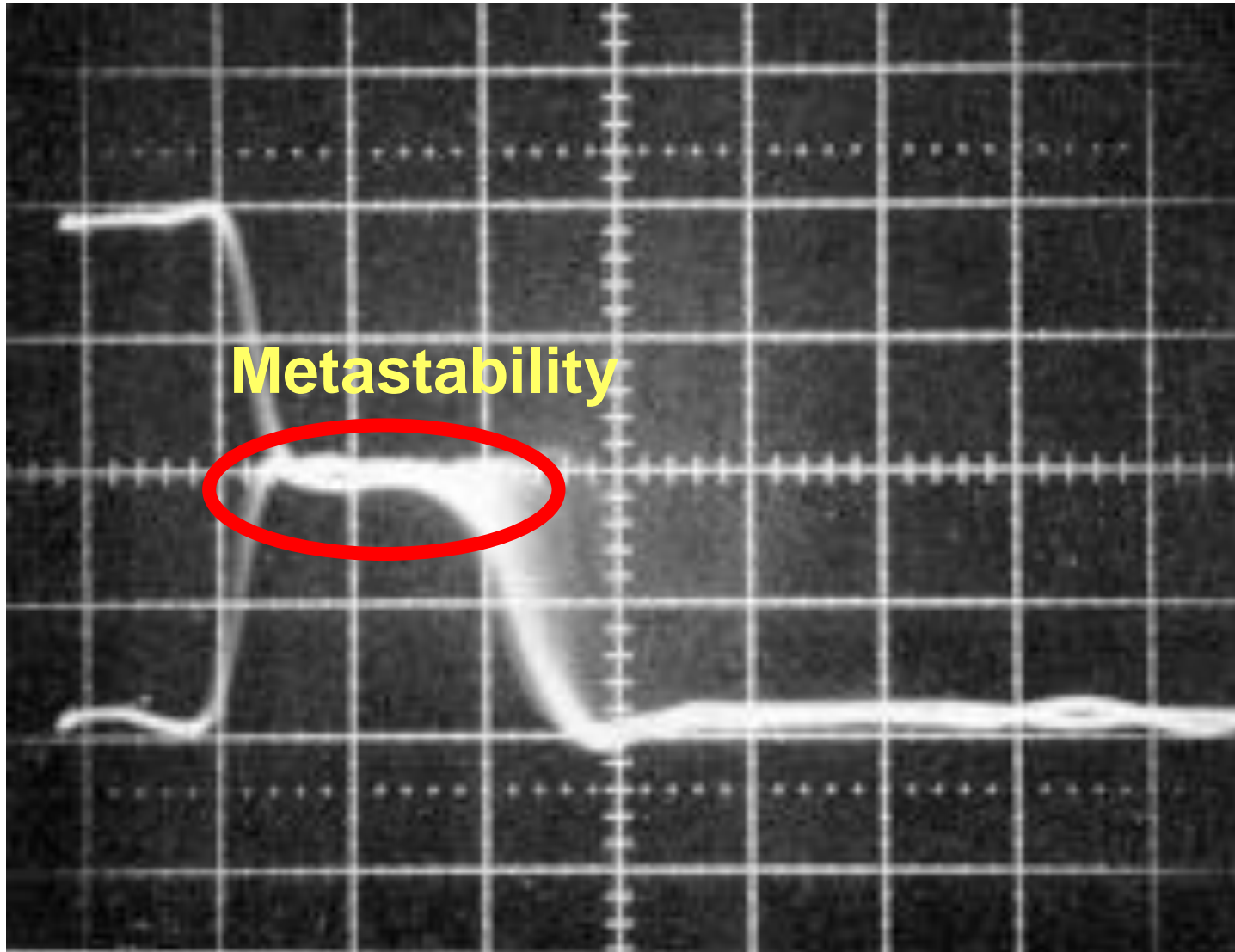
Very Important Timing Considerations!

□ **Setup Time** (T_s): The minimum time during which D input must be maintained **before** the clock transition occurs.

□ **Hold Time** (T_h): The minimum time during which D input must not be changed **after** the clock transition occurs.

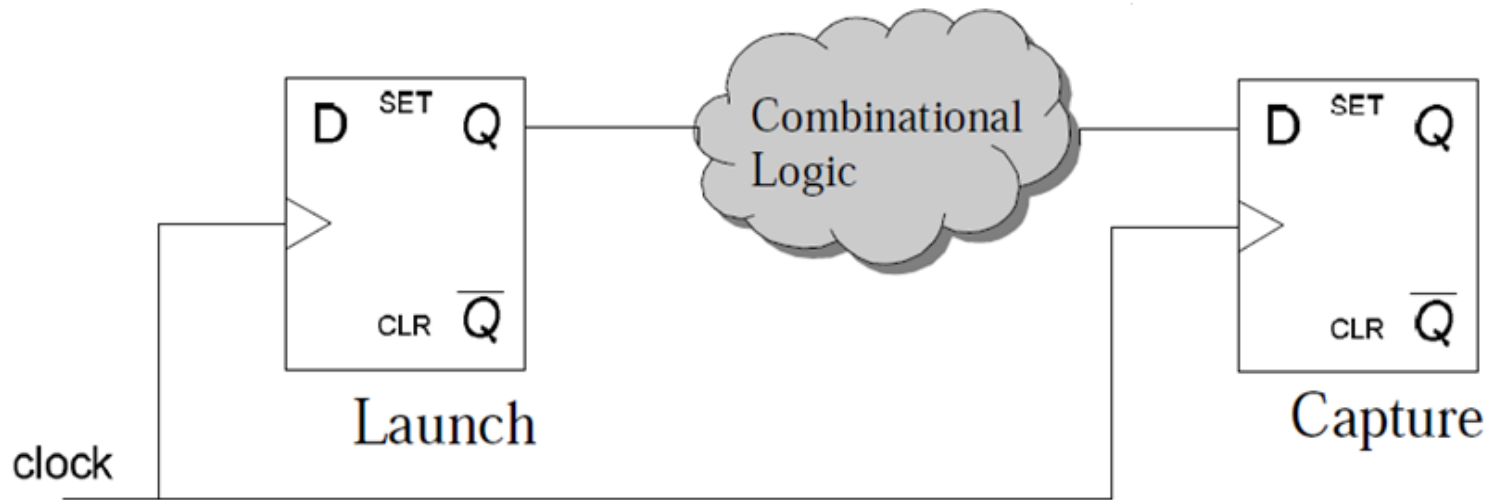


Metastability in Digital Logic



How fast can a synchronous circuit run?

□ RTL



□ Timing analysis:

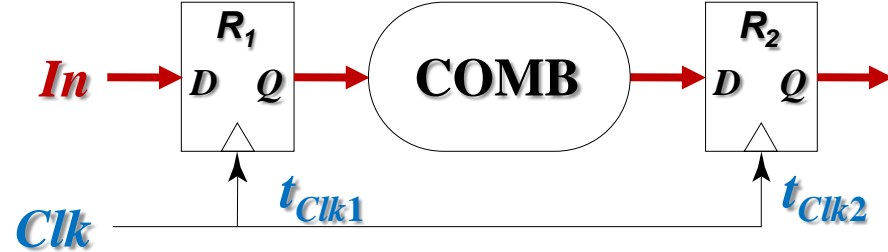
- Starting with the clock rising edge at the launch FF, end with the clock rising edge (**next period or same period**) of the capture FF



Setup Time

□ Setup Timing analysis:

- Starting with the clock rising edge at the launch FF, end with the clock rising edge (**next period**) of the capture FF

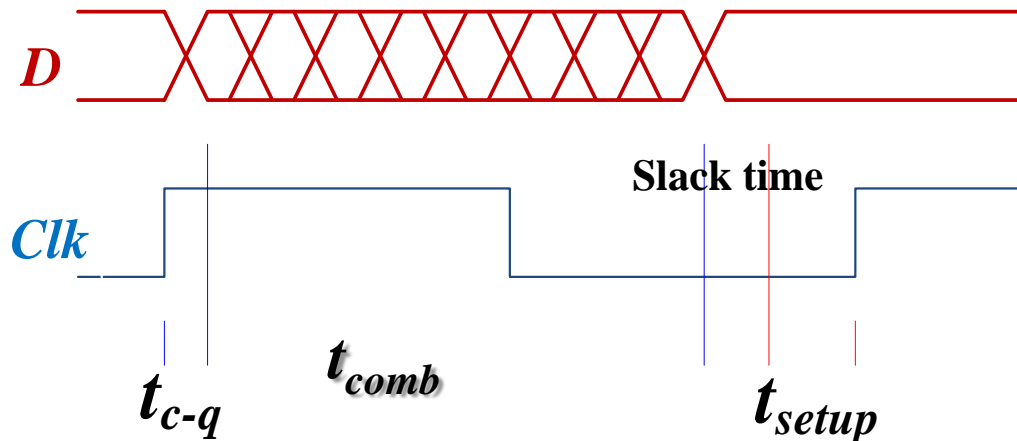


□ **Data-Path (arrive time):** $T_{\text{Combinational logic}} + FF_{\text{launch}}(\text{clk} \rightarrow Q)$

□ **Clock-Path (required time):** Clock Period - $FF_{t\text{Setup}}$

□ **Timing constraint :** $T_{\text{Combinational logic}} + FF_{\text{launch}}(\text{clk} \rightarrow Q) < \text{Clock Period} - FF_{t\text{Setup}}$

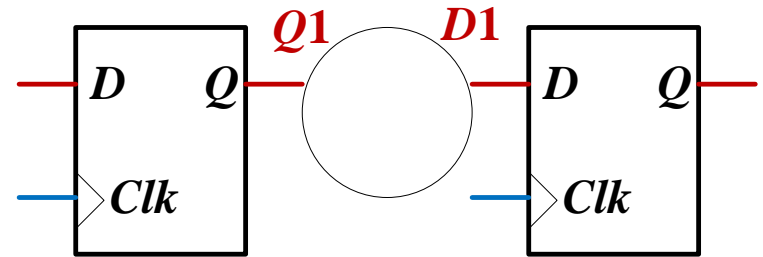
□ **Slack time:** arrive time- require time



Hold Time

Hold Timing analysis:

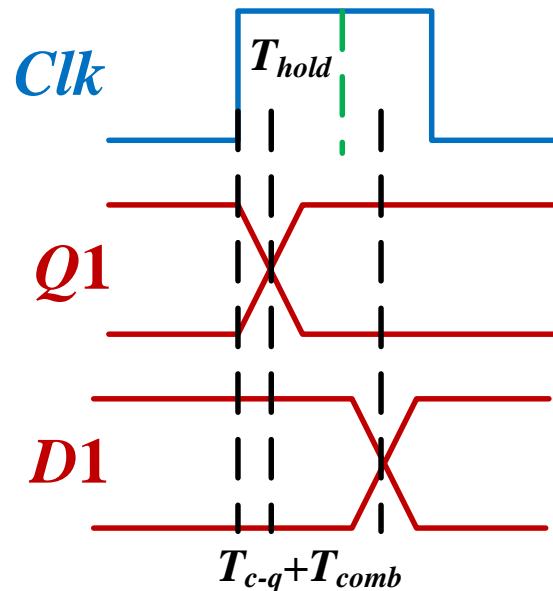
- Starting with the clock rising edge at the launch FF, end with the clock rising edge (same period) of the capture FF



Data-Path (arrive time): $T_{\text{Combinational logic}} + FF_{\text{launch}}(\text{clk} \rightarrow Q)$

Clock-Path (required time): $FF_{t\text{Hold}}$

Timing constraint : $T_{\text{Combinational logic}} + FF_{\text{launch}}(\text{clk} \rightarrow Q) > FF_{t\text{Hold}}$



Clock Uncertainty

□ **Clock uncertainty = skew ± jitter**

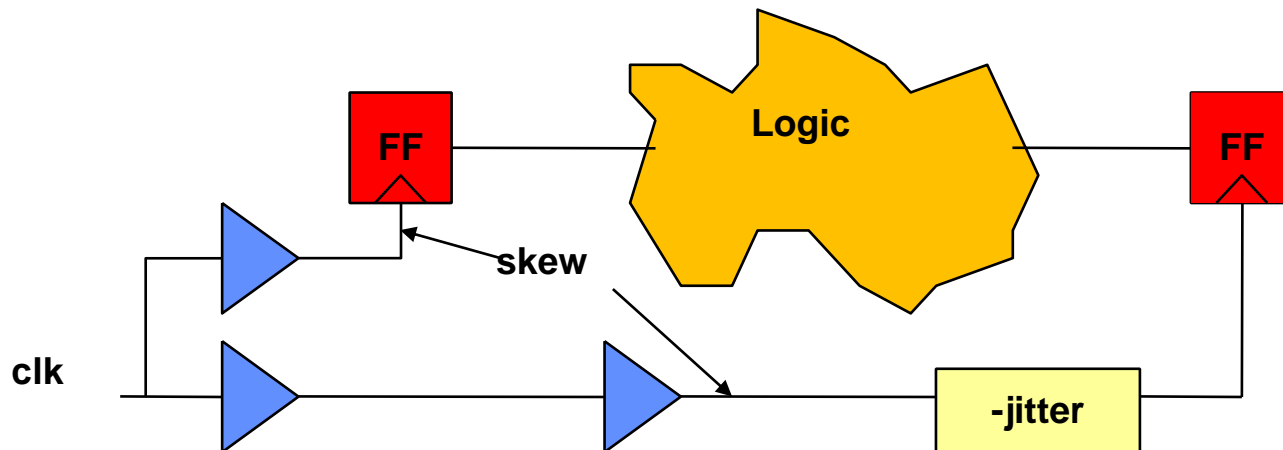
□ **Clock skew**

- The (knowable) difference in clock arrival times at each flip-flop
- Caused mainly by imperfect balancing of clock tree/mesh

□ **Clock jitter**

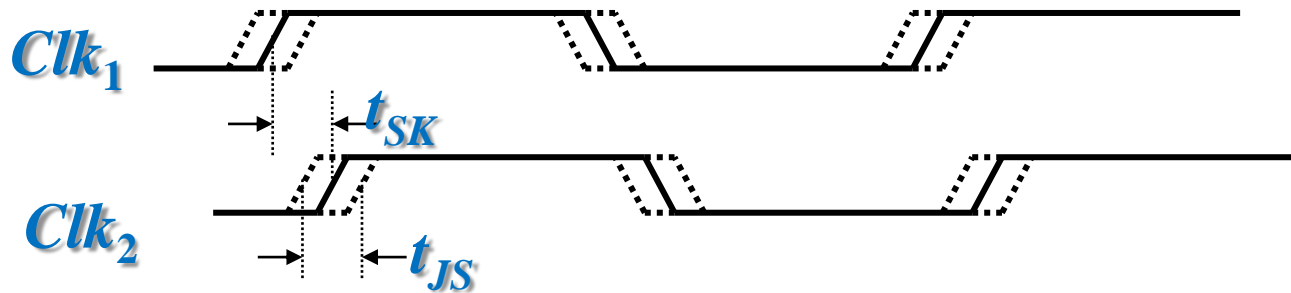
- The random (unknowable) difference in clock arrival times at each flip-flop
- Caused by on-die process, V_{dd} , temperature variation, PLL jitter, crosstalk.

□ **Clock tree to minimize clock uncertainty**

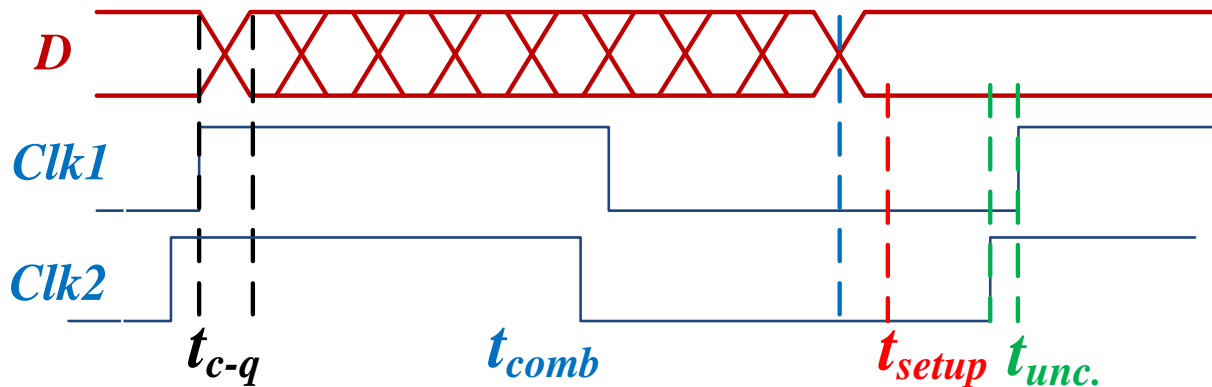


Clock Uncertainty

□ Clock uncertainty = skew \pm jitter



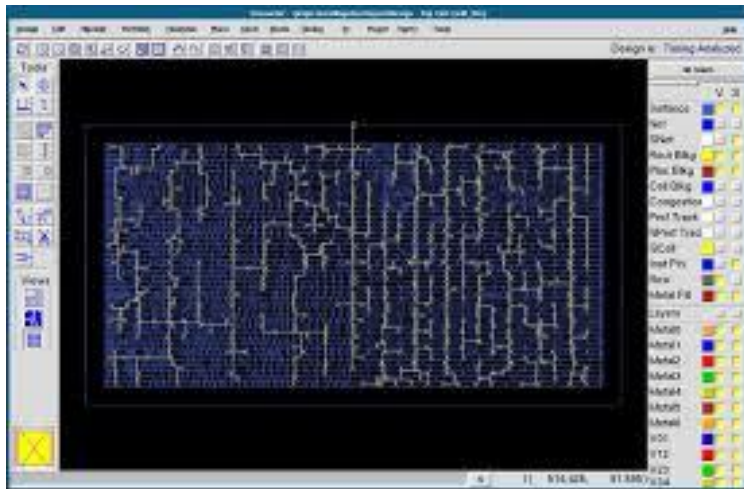
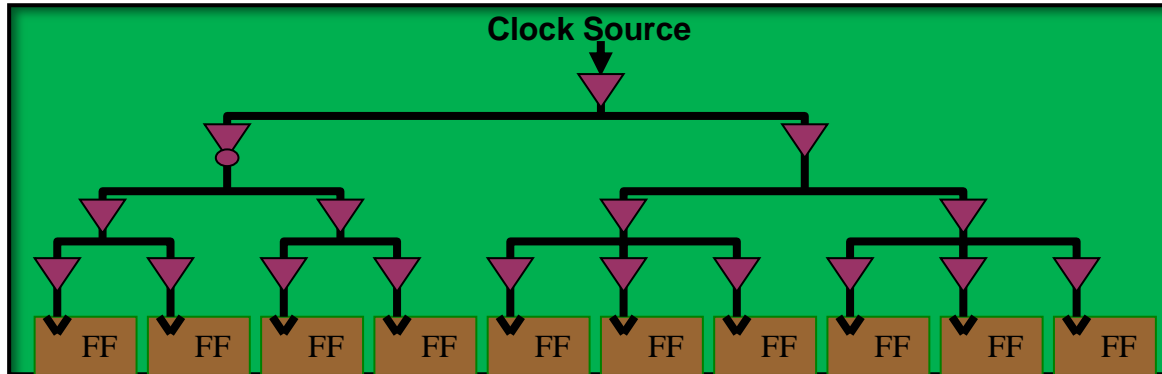
□ Timing analysis with clock uncertainty



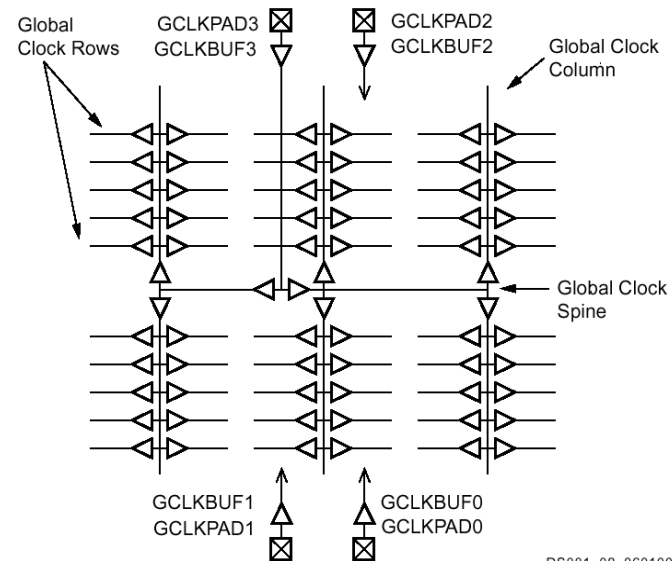
$$T_{\text{Combinational logic}} + FF_{\text{launch}}(\text{clk} \rightarrow \text{Q}) < \text{Clock Period} - FF_{t_{\text{Setup}}} - \text{Clock Uncertainty}$$



Clock Tree



Clock-tree Asic



Global clock network in Xilinx FPGA

DS001_08_060100



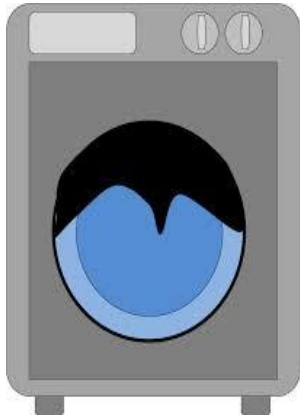
Outline

- Sequential vs. Combinational
- Synchronous vs. Asynchronous
- Basic Storage Elements
- Timing
- **Folding & Pipeline**



Pipelining

□ Start again from laundry room



□ Small laundry has one washer, one dryer and one folder, it takes 110 minutes to finish one load:

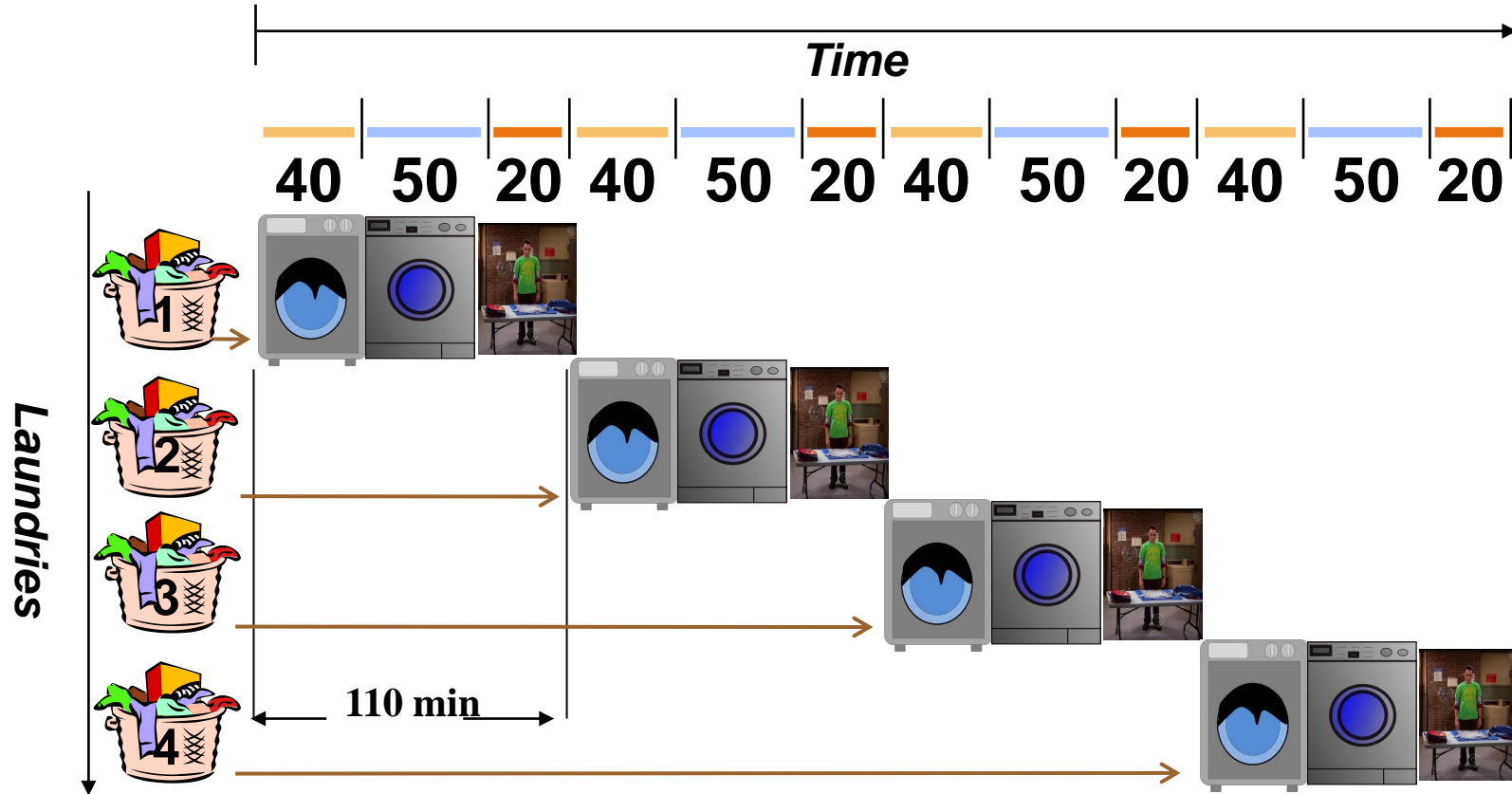
- Washer takes 40 minutes
- Dryer takes 50 minutes
- “Folding” takes 20 minutes



□ Need to do 4 laundries



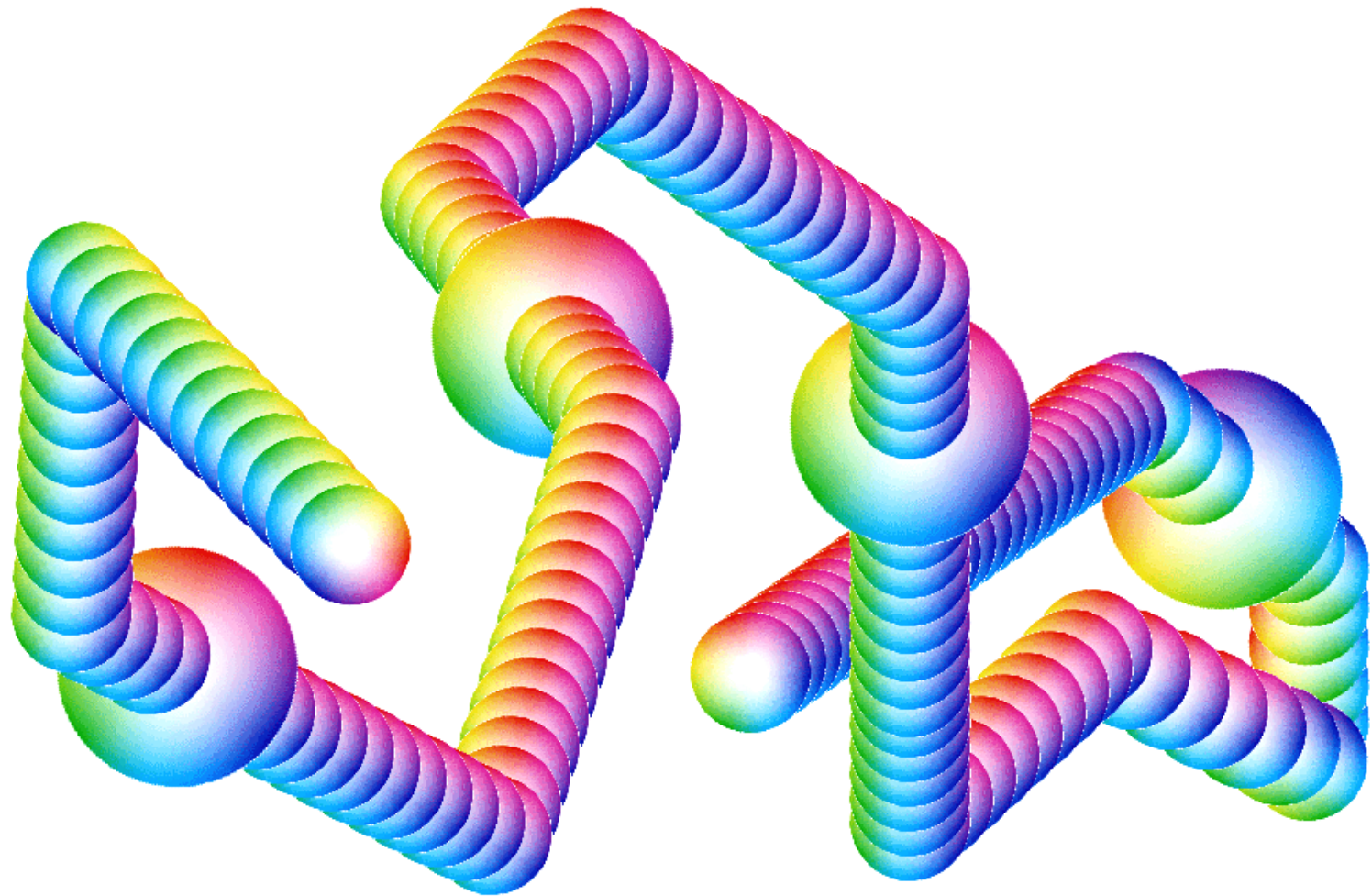
Not very smart way...



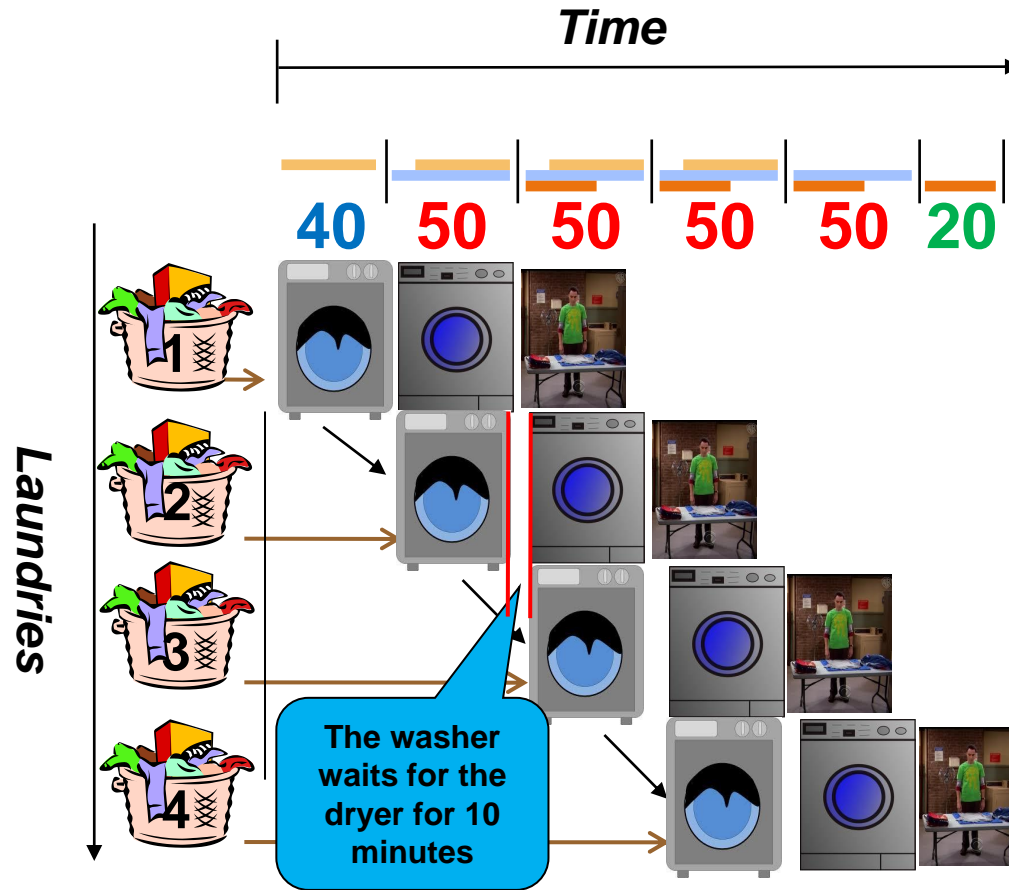
$$\begin{aligned} \text{Total} &= N * (\text{Washer} + \text{Dryer} + \text{Folder}) \\ &= \underline{\quad 440 \quad} \text{ mins} \end{aligned}$$



If we pipelining



If we pipelining

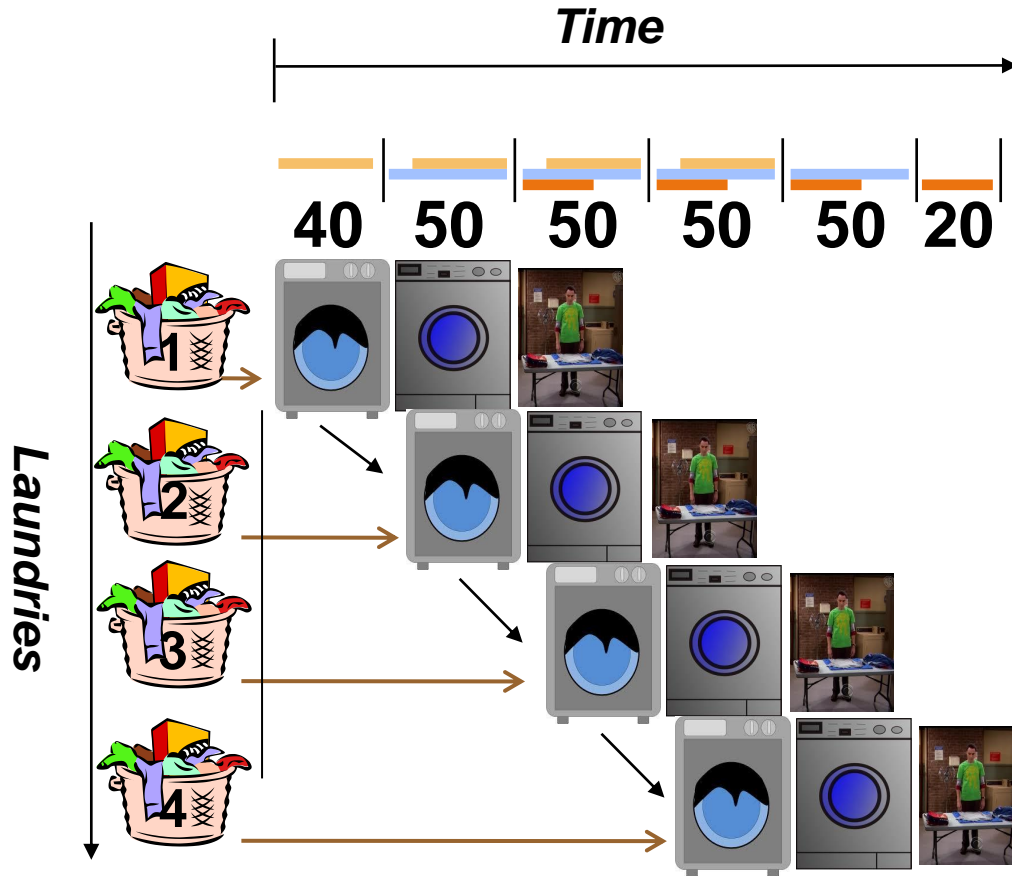


$$\text{Total} = \text{Washer} + N * \text{Max}(\text{Washer}, \text{Dryer}, \text{Folder}) + \text{Folder}$$

$$= \underline{\quad 260 \quad} \text{ mins}$$



Pipeline Facts



- Multiple tasks operating simultaneously
- Pipelining doesn't help **latency** of single task, it helps **throughput** of entire workload
- Pipeline rate limited by slowest pipeline stage**
- Unbalanced** lengths of pipe stages reduces speedup
- Time to “**fill**” pipeline and time to “**drain**” it reduces speedup
- Potential speedup \propto **Number of pipe stages**



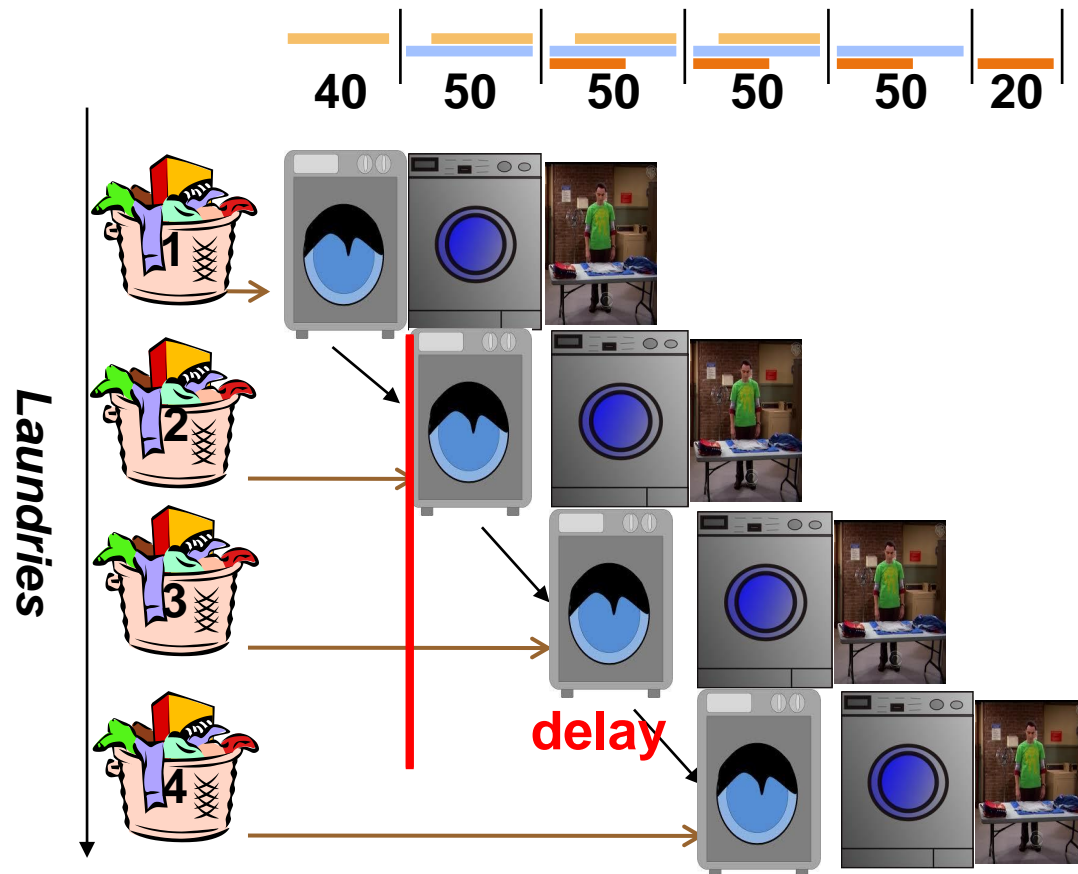
Some definitions

Very Important!

- **Latency**: The delay from when an input is established until the output associated with **that input** becomes valid.

(non-pipeline Laundry = 110 mins)

(pipeline Laundry = 120 mins)



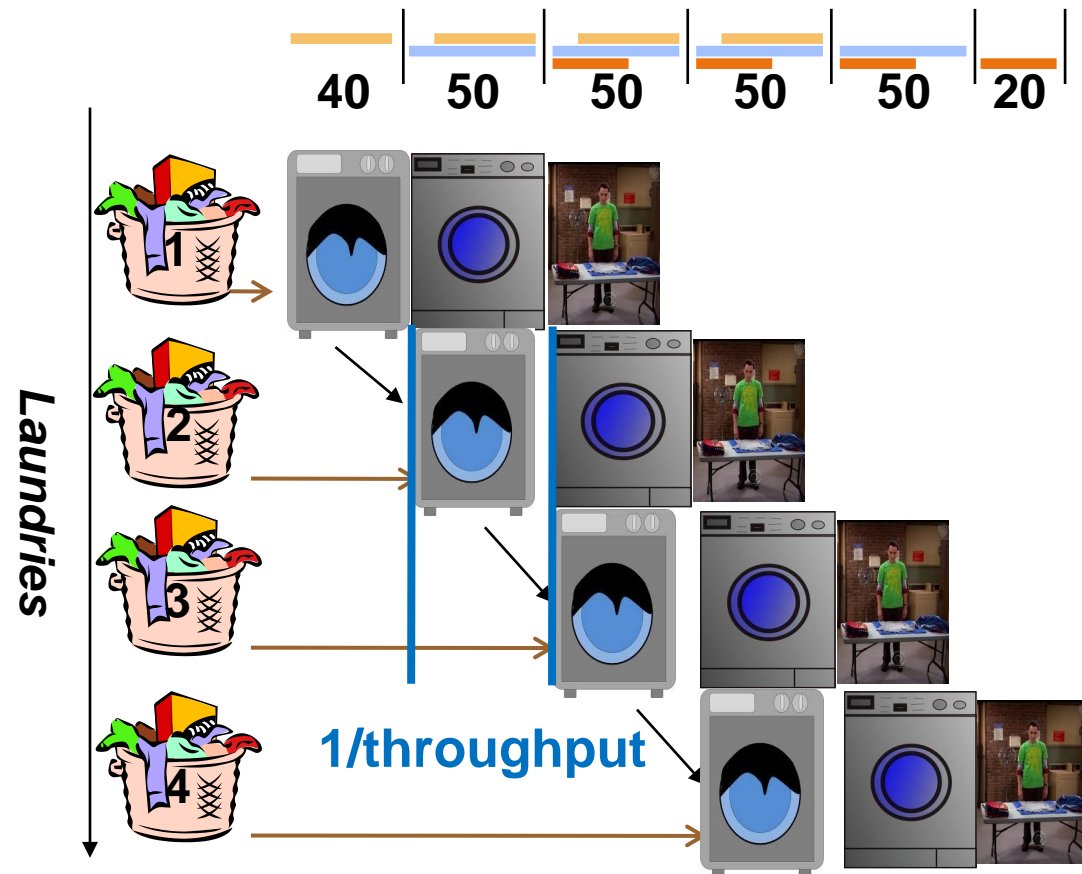
Some definitions

Very Important!

- **Throughput:** The rate of which **inputs** or **outputs** are processed or **how frequently** a laundry can be loaded

(non-pipeline Laundry = $\frac{1}{110}$ outputs/min)

(pipeline Laundry = $\frac{1}{50}$ outputs/min)



Combinational, Folding and Pipelined

□ Combinational Circuits

- **Advantage:** low latency
- **Disadvantage:** low throughput, more hardware, low utilization

□ Folding

- **Advantage:** less hardware, high utilization
- **Disadvantage:** high latency, limited application

□ Pipeline

- **Advantage:** very high throughput
- **Disadvantages:** pipeline latency, more hardware



Thanks!

