Digital Logic Systems

Recitation 10: Signed Addition, Synchronous Circuits and Flip-Flops

Guy Even Moti Medina

School of Electrical Engineering Tel-Aviv Univ.

May 20, 2019

Project Remarks

1. Read the forum, good question are being asked there.
2. Check that your file is uploaded with a proper filename.

Recall - Combinational Circuits

Combinational circuits we saw (before addition)

1. Basic combinational gates (AND2, OR2, MUX2:1,...)
2. MUX(n:1)
3. Shifters (cyclic, logical, arith.) - move input by $\langle \vec{sa} \rangle$ places
4. Decoders n bits$\rightarrow 2^n$ bits
5. Encoders (and priority encoder) 2^n bits$\rightarrow n$ bits

Design approaches

1. Divide and conquer - recursively split the input
2. Flat - Usage of black boxes and smart wiring

Tips

1. **Modularity** is the key!
2. **Evolution** - begin with a naive design, and try improving it

Recall - Binary Adders

We saw **3 different implementations** of binary adder

1. RCA(n) - cheap and intutitive design, but has a linear delay
2. CSA(n) - is slightly more expensive but as a logarithmic delay
3. COMP-ADDER(n) - outputs T and S. Cheaper than CSA, the delay remained logarithmic

Signed Addition

Signed addition warm-up:

1. We must define a representation of a **signed** integer.
2. We must be able to implement **addition/subtraction** of signed integers.
3. Can we use the **same circuitry** for adding unsigned and signed integers?

Signed number representation - Two's Complement

We denote the number represented in two's complement representation by $A[n-1:0]$ as follows:

$$[A[n-1:0]] \stackrel{\triangle}{=} -2^{n-1} \cdot A[n-1] + \langle A[n-2:0] \rangle.$$

We denote the set of signed numbers that are representable in two's complement representation using n-bit binary strings by T_n.

The Set T_n

$$T_n = \left\{-2^{n-1}, -2^{n-1}+1, \ldots, 2^{n-1}-1\right\}.$$

Example

$T_{32} = \{-2147483648, ..., 2147483647\}$ which should remind you of the limits of int variables, compiled for 32-bit machines.

Two's Complement - Sign bit, Sign extension & Negation

Claim (MSB determines the sign)

$$[A[n-1:0]] < 0 \quad \Longleftrightarrow \quad A[n-1] = 1.$$

Claim (Sign-Extension)

If $A[n] = A[n-1]$, *then*

$$[A[n:0]] = [A[n-1:0]]\,.$$

Corollary (Sign-Extension)

$$[A[n-1]^* \circ A[n-1:0]] = [A[n-1:0]]\,.$$

Claim (Negation)

$$-[A[n-1:0]] = [\textsc{inv}(A[n-1:0])] + 1$$

Comparison between representation methods

binary string $\vec{X}$	$\langle \vec{X} \rangle$	2's comp	1's comp	sign-mag
000	0	0	0	+0
001	1	1	1	1
010	2	2	2	2
011	3	3	3	3
100	4	−4	−3	−0
101	5	−3	−2	−1
110	6	−2	−1	−2
111	7	−1	0	−3

- symmetric range: one's complement and sign-magnitude.
- two representations for zero: one's complement and sign-magnitude.

Signed Arithmetic Circuits

So far, we've defined a representation of **signed** integers and had a few observations about it.
It is now time to design the two's complement arithmetic logic.
Our design will follow the next procedure:

1. Define a **reduction** of two's complement to a binary addition
2. Detect **overflow** and the **sign** of the sum.
3. Construct a **signed adder** S-ADDER(n)
4. Extend the S-ADDER(n) to **adder/subtractor** ADD-SUB(n)

Reduction of two's comp addition to binary addition

Theorem

Let $C[n-1]$ denote the carry-bit in position $[n-1]$ associated with the binary addition described in Equation 16.1 and let

$$z \stackrel{\triangle}{=} [A[n-1:0]] + [B[n-1:0]] + C[0].$$

Then,

$$C[n] - C[n-1] = 1 \implies z < -2^{n-1} \quad (1)$$

$$C[n-1] - C[n] = 1 \implies z > 2^{n-1} - 1 \quad (2)$$

$$z \in T_n \iff C[n] = C[n-1] \quad (3)$$

$$z \in T_n \implies z = [S[n-1:0]]. \quad (4)$$

Example

$[A[3:0]]$	-3	-4	-6	7
$[B[3:0]]$	-5	-5	5	1
$C[0]$	1	0	0	1
$C[n]$	1	1	1	0
$C[n-1]$	1	0	1	1
$[S[n-1:0]]$	-7	7	-1	-7
z	-7	-9	-1	9

Detecting overflow

Overflow - the sum of signed numbers is not in T_n.

Definition

Let $z \stackrel{\triangle}{=} [A[n-1:0]] + [B[n-1:0]] + C[0]$. The signal OVF is defined as follows:

$$\mathrm{OVF} \stackrel{\triangle}{=} \begin{cases} 1 & \text{if } z \notin T_n \\ 0 & \text{otherwise.} \end{cases}$$

the term "out-of-range" is more appropriate than "overflow" (which suggests that the sum is too big). Favor tradition...

Claim

$$\mathrm{OVF} = \mathrm{XOR}(C[n-1], C[n]).$$

Determining the sign of the sum

Definition

The signal NEG is defined as follows:

$$\mathrm{NEG} \triangleq \begin{cases} 1 & \text{if } z < 0 \\ 0 & \text{if } z \geq 0. \end{cases}$$

brute force method:

$$\mathrm{NEG} = \begin{cases} S[n-1] & \text{if no overflow} \\ 1 & \text{if } C[n] - C[n-1] = 1 \\ 0 & \text{if } C[n-1] - C[n] = 1. \end{cases} \tag{5}$$

Claim

The NEG *signal determines the sign of the correct sum (z) even in case of an overflow. Its function can be expressed as follows:*

$$\mathrm{NEG} = \mathrm{XOR}_3(A[n-1], B[n-1], C[n])$$

A two's-complement adder

Definition

A two's-complement adder with input length n is a combinational circuit specified as follows.

Input: $A[n-1:0], B[n-1:0] \in \{0,1\}^n$, and $C[0] \in \{0,1\}$.

Output: $S[n-1:0] \in \{0,1\}^n$ and $\mathrm{NEG}, \mathrm{OVF} \in \{0,1\}$.

Functionality: Define z as follows:

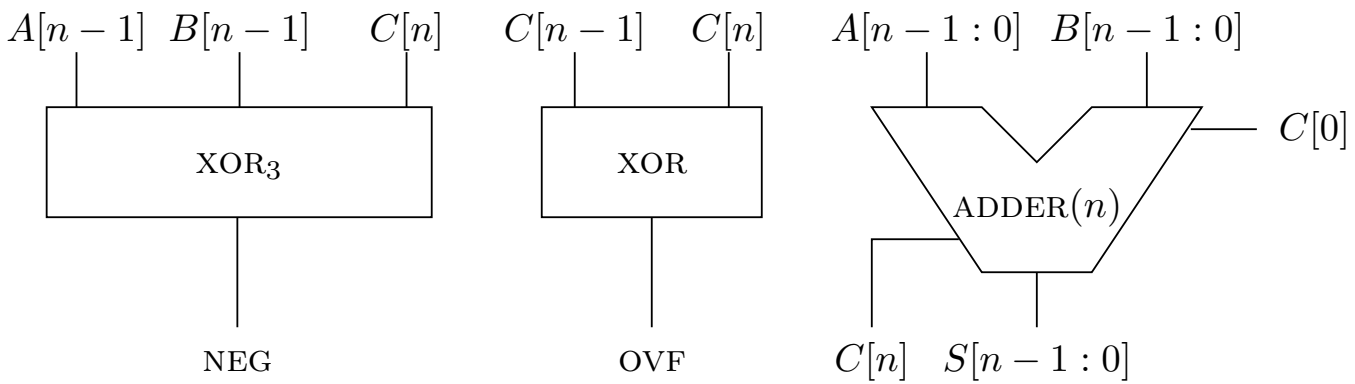
$$z \stackrel{\triangle}{=} [A[n-1:0]] + [B[n-1:0]] + C[0].$$

The functionality is defined as follows:

$$\begin{aligned} z \in T_n &\implies [S[n-1:0]] = z \\ z \in T_n &\iff \mathrm{OVF} = 0 \\ z < 0 &\iff \mathrm{NEG} = 1. \end{aligned}$$

We denote a two's-complement adder by $\text{S-ADDER}(n)$.

A two's complement adder S-ADDER(n)



In an arithmetic logic unit (ALU), one may use the same circuit for signed addition and unsigned addition.

A two's complement adder/subtractor

Definition

A two's-complement adder/subtractor with input length n is a combinational circuit specified as follows.

Input: $A[n-1:0], B[n-1:0] \in \{0,1\}^n$, and $sub \in \{0,1\}$.

Output: $S[n-1:0] \in \{0,1\}^n$ and $\mathrm{NEG}, \mathrm{OVF} \in \{0,1\}$.

Functionality: Define z as follows:

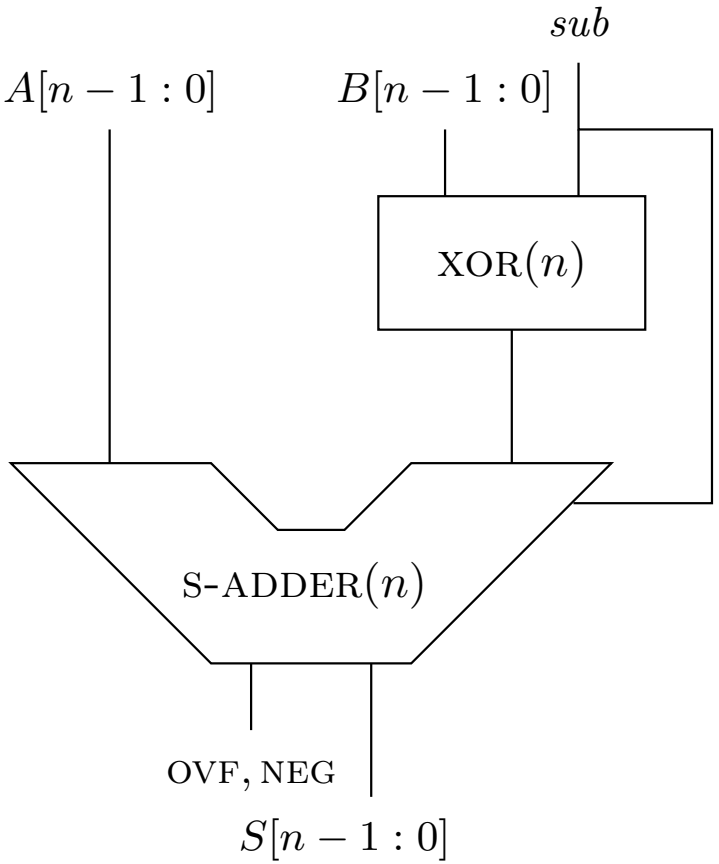
$$z \triangleq [A[n-1:0]] + (-1)^{sub} \cdot [B[n-1:0]].$$

The functionality is defined as follows:

$$\begin{aligned} z \in T_n &\implies [S[n-1:0]] = z \\ z \in T_n &\iff \mathrm{OVF} = 0 \\ z < 0 &\iff \mathrm{NEG} = 1. \end{aligned}$$

We denote a two's-complement adder/subtractor by $\mathrm{ADD\text{-}SUB}(n)$.

An implementation of an ADD-SUB(n)



Claim

The implementation of ADD-SUB(n) *is correct.*

Exam Fall-2011, Question B

ב. **(7 נק')** המספר המיוצג בשיטת *משלים ל-2* על ידי $A[n-1:0]$ הוא:

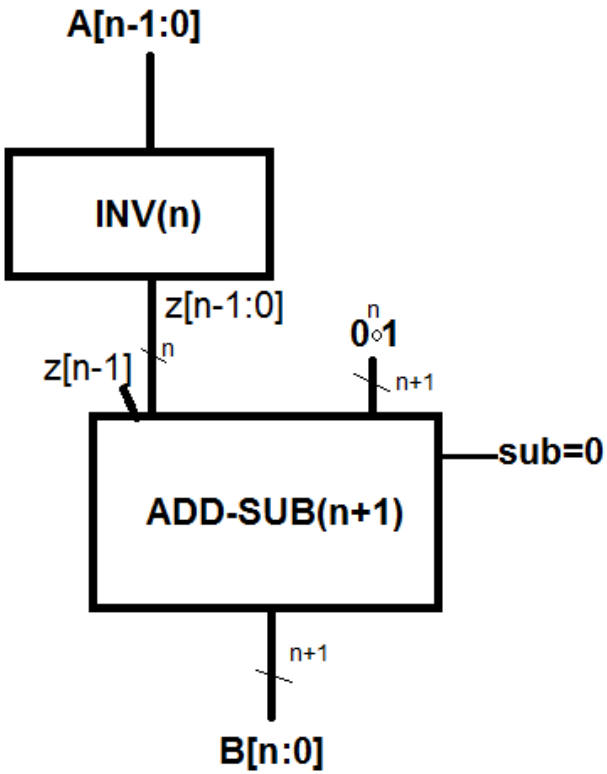
$$\left[A[n-1:0]\right] \triangleq -2^{n-1} \cdot A[n-1] + \sum_{i=0}^{n-2} A[i] \cdot 2^i .$$

ציירו סכימה של מעגל שבהינתן מחרוזת $A[n-1:0] \in \{0,1\}^n$, פולט מחרוזת $B[n:0] \in \{0,1\}^{n+1}$ כך ש:

$$[B] = -[A].$$

שימו לב שבקלט יש ***n*** ביטים ובפלט יש ***n+1*** ביטים.

Exam Fall-2011, Question B - Answer



Explanation

The duplication of $z[n-1]$ extends the sign of z for $n+1$ bits. The circuit will never output overflow and its correctness is according to the claim proved in class $-[A] = [\bar{A}] + 1$.

Exam Fall-2014 Moed B, Question 1

שאלה 1 – מעגל צירופי (30 נק')

המעגל C_n מוגדר לפי המפרט הבא (הניחו כי $n>5$):

קלט: $x[n-1:0] \in \{0,1\}^n$

פלט: $y[n-1:0] \in \{0,1\}^n, \quad ovf, neg \in \{0,1\}$

פונקציונליות: נגדיר את z באופן הבא: $z \triangleq 31 \cdot [x[n-1:0]]$ (31 כפול המספר המיוצג על x בשיטת משלים ל-2)

ההגדרה החלקית של y היא: $if\ z \in \{-2^{n-1}, \dots, 2^{n-1}-1\}\ then\ [y] = [z]$

ההגדרות של ovf, neg הן:

$$neg \triangleq \begin{cases} 1 & if\ z < 0 \\ 0 & otherwise \end{cases}$$

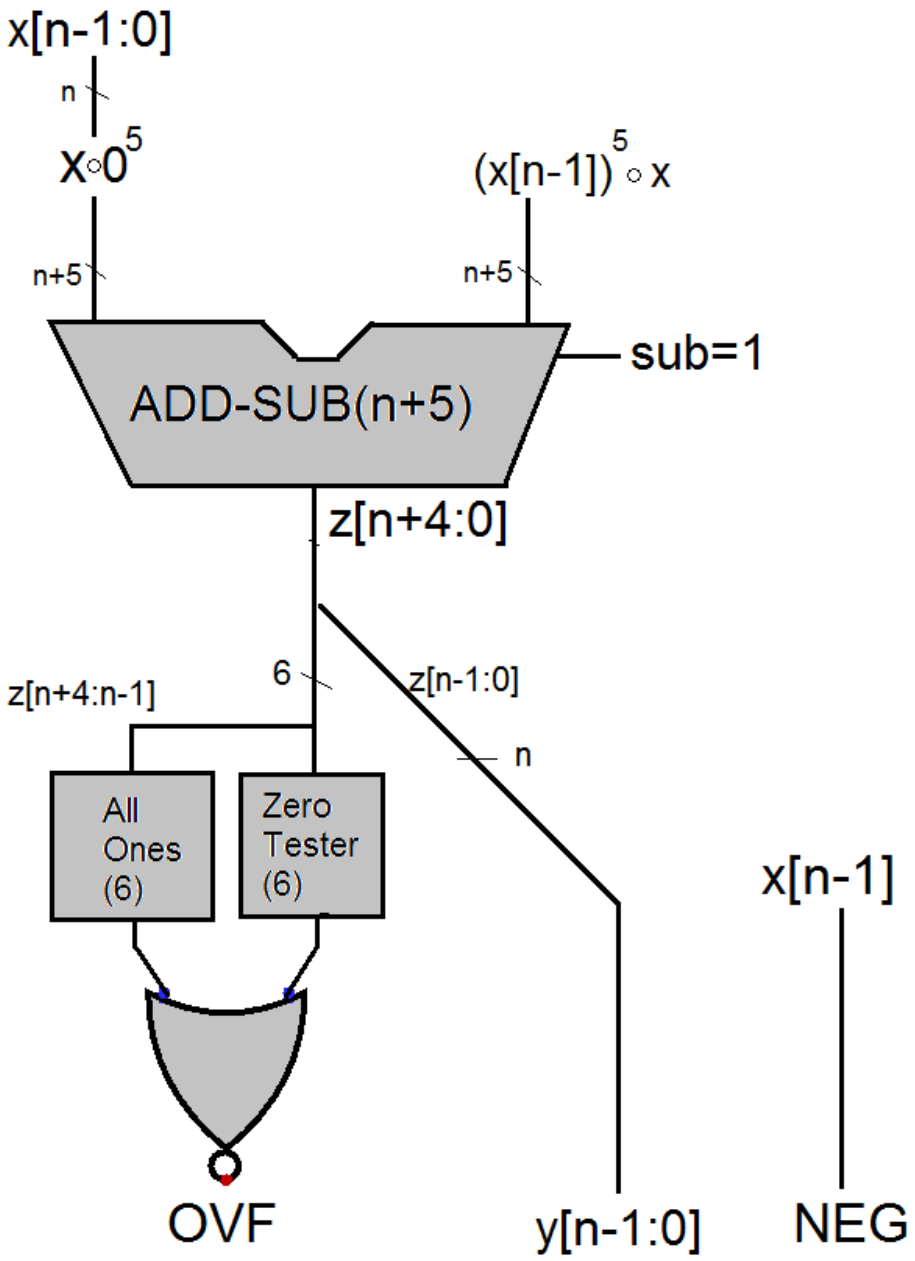
$$ovf \triangleq \begin{cases} 1 & if\ z \notin \{-2^{n-1}, \dots, 2^{n-1}-1\} \\ 0 & if\ z \in \{-2^{n-1}, \dots, 2^{n-1}-1\} \end{cases}$$

א. (60%) ציירו סכמת בלוקים של המעגל C_n. (עם עדיפות להשהיה קטנה על פני מחיר נמוך)

ג. (15%) חשבו את מחיר המעגל C_n כפי שמימשתם אותו בסעיף א.

ד. (15%) חשבו את השהיית המעגל C_n כפי שמימשתם אותו בסעיף א.

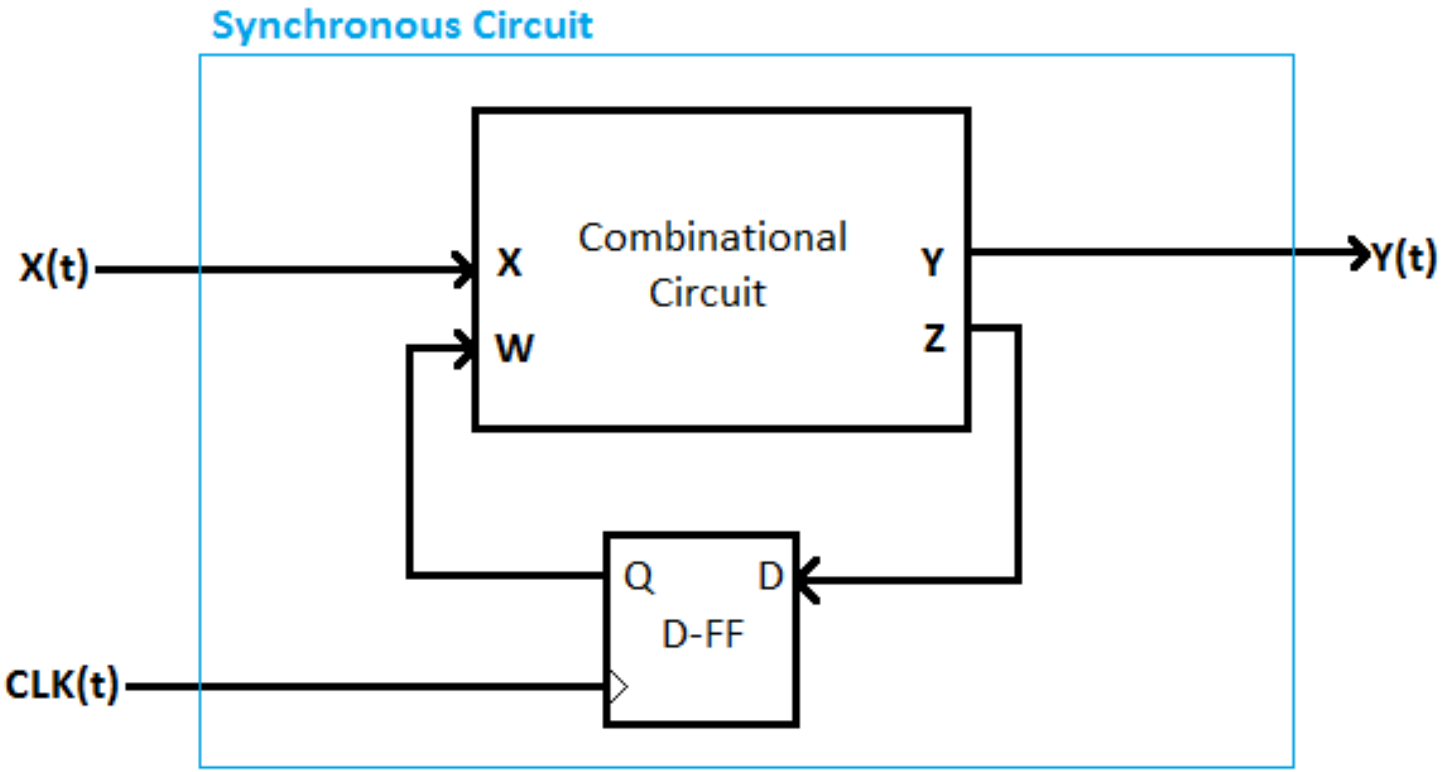
Exam Fall-2014 Moed B, Question 1 - Answer



Synchronous Circuits

Synchronous Circuits - Attributes

1. Consist of combinational circuits and flip-flops (FFs)
2. Can contain cycles, as long as the underlying combinational circuits are still acyclic.
3. Must contain a special CLK input, which should be fed to all FFs



Synchronous Circuits - Time

1. We introduce **discrete time**
2. The time is dictated by a very special signal **CLK**$\in\{0,1\}$. This signal is given automatically, you don't have to worry about generating it. Just don't forget to connect it to the required elements (FFs).
3. Each **rising-edge** of CLK advances the time to the next clock cycle.
4. The inputs are now **time-dependent**: No more X, Y, Z. In synchronous circuits we deal with $X(t), Y(t), Z(t)$. The "user" of a synchronous circuit can change these inputs every clock cycle.
5. Don't confuse between the time indices and the string indices.

Example

- $X[i](t)$ is the value of the binary string X at index i at time t.
- $X[2](3)$ is the value of the binary string X at index 2 at time 3.

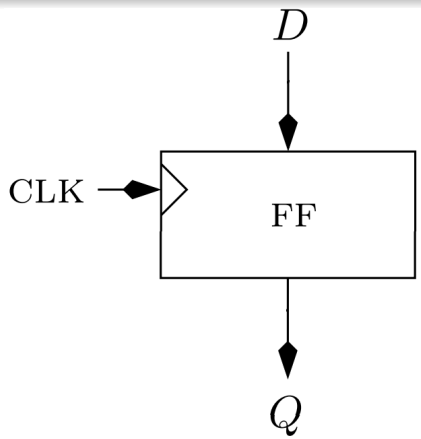
Edge-triggered flip-flop

Definition (edge-triggered flip-flop)

Inputs: $D(t)$ and a clock CLK.

Output: $Q(t)$.

Functionality: If $D(t)$ is stable during the critical segment C_i, then $Q(t) = D(t_i)$ during the interval $(t_i + t_{pd}, t_{i+1} + t_{cont})$.

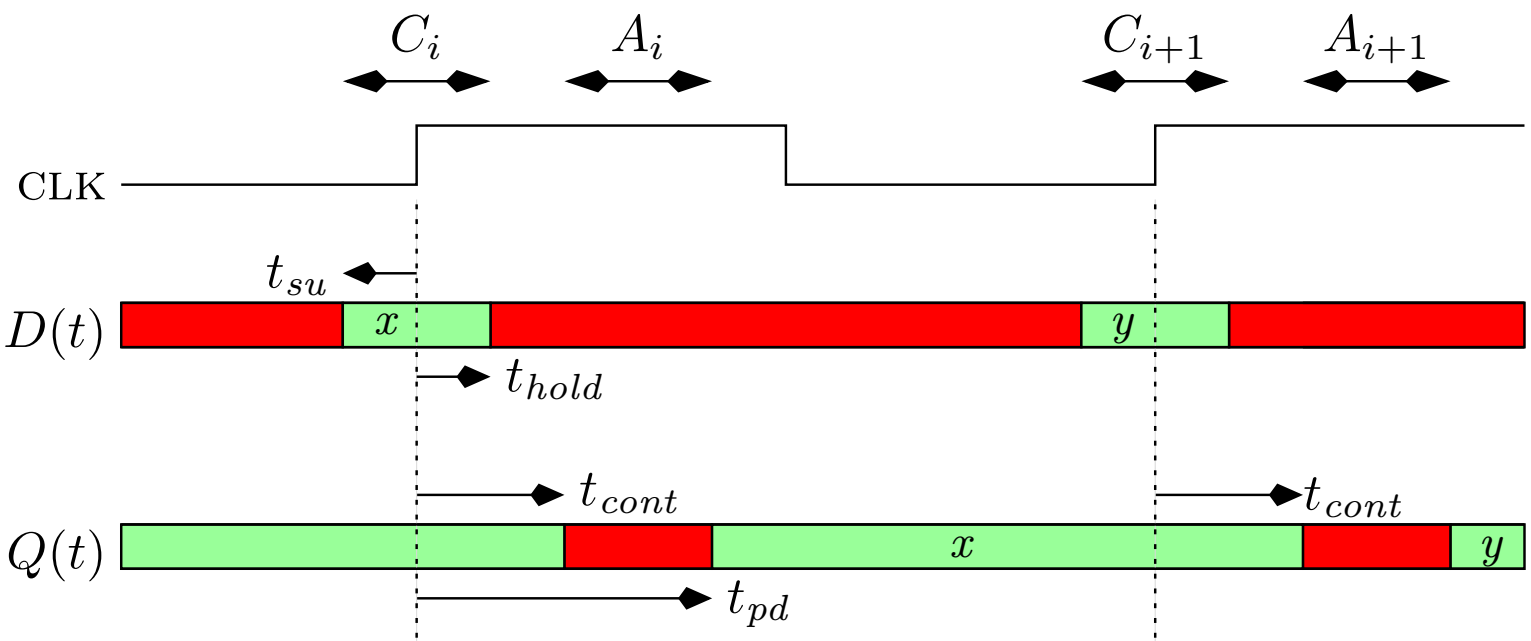


Cost of a Flip-Flop

The FFs have their own cost which is typically much more expensive than combinational gates. Hence, a cost of a synchronous circuit will consist of combinational price + FF price.

Timing diagram of a Flip Flop

- The x-axis corresponds to time.
- A green interval means that the signal is stable during this interval.
- A red interval means that the signal may be instable.
- In order to sample the $D(t)$ by the FF, $D(t)$ must be stable during the C_i, or in other words:
- In order to sample the $D(t)$ by the FF, $D(t)$ must be stable t_{su} seconds before the rising edge and until t_{hold} after it.



The Zero Delay Model

Simplified model for specifying and simulating the functionality of circuits with flip-flops.

1. Transitions of all signals are instantaneous.
2. Combinational gates: $t_{pd} = t_{cont} = 0$.
3. Flip-flops satisfy:

$$t_{su} = t_{i+1} - t_i,$$
$$t_{hold} = t_{cont} = t_{pd} = 0.$$

4. This allows us to specify the functionality of a flip-flop in the zero delay model as follows:

$$Q(t+1) = D(t).$$

Clock enabled flip-flops (zero-delay model)

Definition

A clock enabled flip-flop is defined as follows.

Inputs: $D(t), \mathrm{CE}(t) \in \{0,1\}$ and a clock CLK.

Output: $Q(t) \in \{0,1\}$.

Functionality:

$$Q(t+1) = \begin{cases} D(t) & \text{if } \mathrm{CE}(t) = 1 \\ Q(t) & \text{if } \mathrm{CE}(t) = 0. \end{cases}$$