Digital Logic Design: a rigorous approach ©

Chapter 19: Foundations of Synchronous Circuits

Guy Even Moti Medina

School of Electrical Engineering Tel-Aviv Univ.

May 30, 2019

Book Homepage:
http://www.eng.tau.ac.il/~guy/Even-Medina

Preliminary questions

1. What is a synchronous circuit?
2. How can we tell if the clock period is not too short? Is it possible to compute the minimum clock period?
3. Is it possible to separate between the timing analysis and functionality in synchronous circuits?
4. How can we initialize a synchronous circuit?

Synchronous Circuits

- building blocks: combinational gates, wires, and flip-flops.
- the graph G of a synchronous circuit is directed but may contain cycles.
- use flip-flops, hence the labeling $\pi : V \to \Gamma \cap IO \cup \{FF\}$.
- a flip-flop has two inputs D and $\textsc{clk}$ that play quite different roles. We must make sure that we know the input port of each incoming edge.
- the clock signal must be fed to the $\textsc{clk}$ input port of each and every flip-flop!
- definition based on a reduction to a combinational circuit...

Synchronous Circuits

Definition

A synchronous circuit is a circuit C composed of combinational gates, wires, and flip-flops that satisfies the following conditions:

1. There is an input gate that feeds the clock signal CLK.
2. The set of ports that are fed by the clock CLK equals the set of clock-inputs of the flip-flops.
3. Let C' denote a circuit obtained from C by stripping the flip-flops away. Then, the circuit C' is a combinational circuit.

Stripping Flip-Flops Away

Definition

1. Delete the input gate that feeds the clock CLK and all the wires carrying the clock signal.
2. Remove all the flip-flops.
3. Add an output gate for each D port.
4. Add an input gate for each Q port.

Example - stripping FFs away

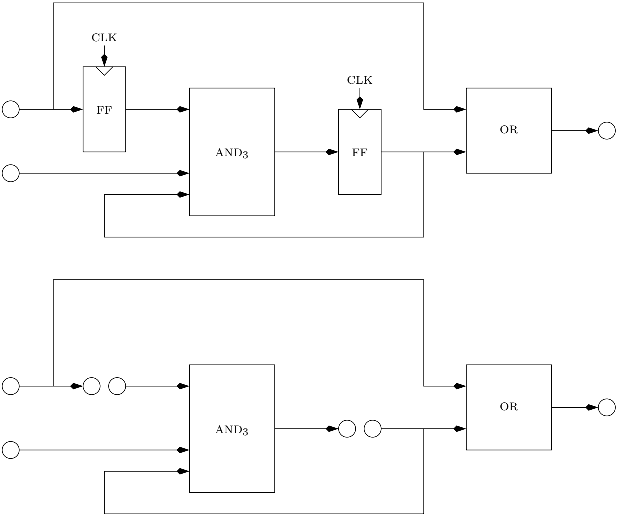


Figure: A synchronous circuit C and the combinational circuit C' obtained from C by stripping away the flip-flops.

Remarks:

It is easy to check if a given circuit C is a synchronous circuit.

- Check if there is a clock signal that is connected to all the clock terminals of the flip-flops and only to them.
- Strip the flip-flops away to obtain the circuit C'. Check if C' is a combinational circuit.

Cycles in a synchronous circuit

Claim

Every cycle in a synchronous circuit traverses at least one flip-flop.

The Canonic Form of a Synchronous Circuit

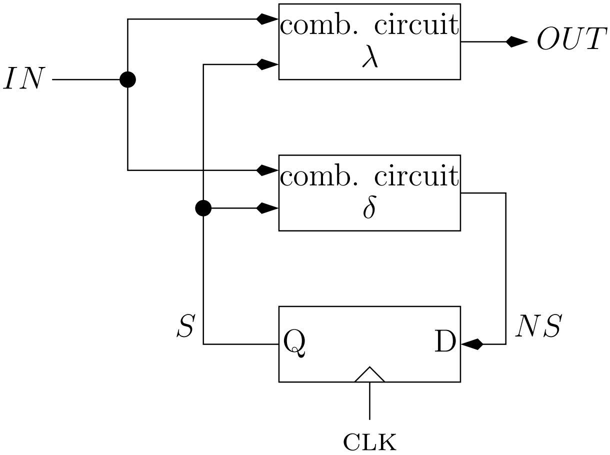


Figure: A synchronous circuit in canonic form.

Stability interval

Definition

stability interval of signal X: an interval corresponding to the ith clock cycle during which the signal X is stable.

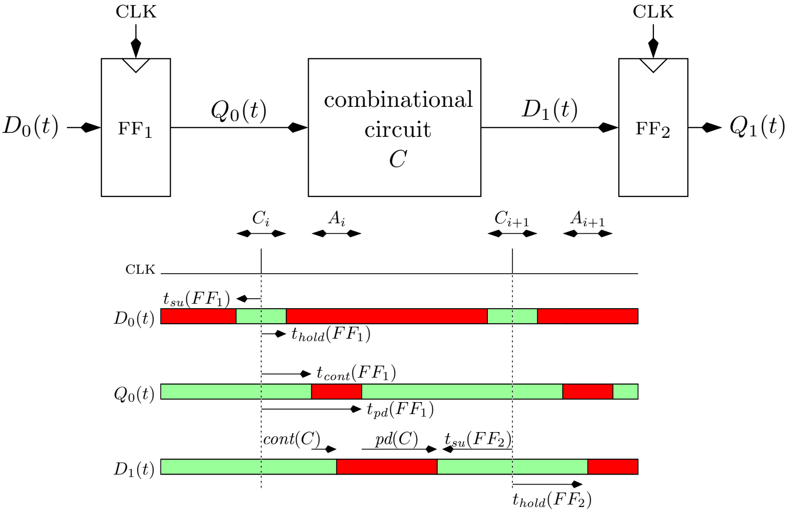
Notation:

- $stable(X)_i$ - stability interval of X during clock cycle i.
- X_i - the digital value of X during the interval $stable(X)_i$.

Stability Interval and Specification:

- If X is an input, then we are guaranteed that the input will stable during $stable(X)_i$.
- If Y is an output, then we must design the circuit so that Y will be stable during $stable(Y)_i$.

Example



CLK
CLK
$D_0(t)$
FF$_1$
$Q_0(t)$
combinational
circuit
C
$D_1(t)$
FF$_2$
$Q_1(t)$
C_i
A_i
C_{i+1}
A_{i+1}
CLK
$t_{su}(FF_1)$
$D_0(t)$
$t_{hold}(FF_1)$
$t_{cont}(FF_1)$
$Q_0(t)$
$t_{pd}(FF_1)$
$cont(C)$
$pd(C)$
$t_{su}(FF_2)$
$D_1(t)$
$t_{hold}(FF_2)$

Stability Interval of D_0

We require that the input $D_0(t)$ to flip-flop FF_1 is stable during the critical segments of FF_1, namely, for every $i \geq 0$:

$$\begin{aligned} stable(D_0)_i &\triangleq C_{i+1}(FF_1) \\ &= (t_{i+1} - t_{su}(FF_1), t_{i+1} + t_{hold}(FF_2)). \end{aligned}$$

Note, that the stability interval corresponding to the ith clock cycle of an input of a flip-flop must contain the critical segment C_{i+1}. Indeed, in the ith clock cycle, the flip-flop samples its input at the end of the cycle, at time t_{i+1}.

Stability Interval of Q_0

The stability interval of the output $Q_0(t)$ of flip-flop FF_1 is defined by

$$stable(Q_0)_i \triangleq (t_i + t_{pd}(FF_1), t_{i+1} + t_{cont}(FF_1)).$$

The rational behind this definition is that if the input $D_0(t)$ is stable during every critical segment C_i, then the output $Q_0(t)$ of the flip-flop is stable in the above interval.

Problem!?

- we have a problem with the guarantee for the stability interval of Q_0 during clock cycle zero.
- This is not a minor technical issue! How can we argue anything about the output of FF_1 during clock cycle zero?!
- To solve this problem, we need an initialization assumption...
- In the meantime, assume that

$$stable(Q_0)_i \triangleq (t_i + t_{pd}(FF_1), t_{i+1} + t_{cont}(FF_1)).$$

holds also for $i = 0$.

Stability Interval D_1

To ensure proper functionality, the input $D_1(t)$ must be stable during the critical segments of flip-flop FF_2. Therefore, we define the stability interval of $D_1(t)$ as follows:

$$\begin{aligned} stable(D_1)_i &\triangleq C_{i+1}(FF_2) \\ &= (t_{i+1} - t_{su}(FF_2), t_{i+1} + t_{hold}(FF_2)). \end{aligned}$$

Timing analysis: sufficient conditions

A sufficient condition that guarantees that $D_1(t)$ is indeed stable during the stability intervals $\{stable(D_1)_i\}_{i \geq 0}$.

Claim

The signal $D_1(t)$ is stable during the critical segments of flip-flop FF_2 if

$$\forall i \geq 0: \; t_{\text{pd}}(FF_1) + \text{pd}(C) + t_{\text{su}}(FF_2) \leq t_{i+1} - t_i, \; \textit{and}$$
$$t_{\text{hold}}(FF_2) \leq t_{\text{cont}}(FF_1) + \text{cont}(C).$$

two important lessons:

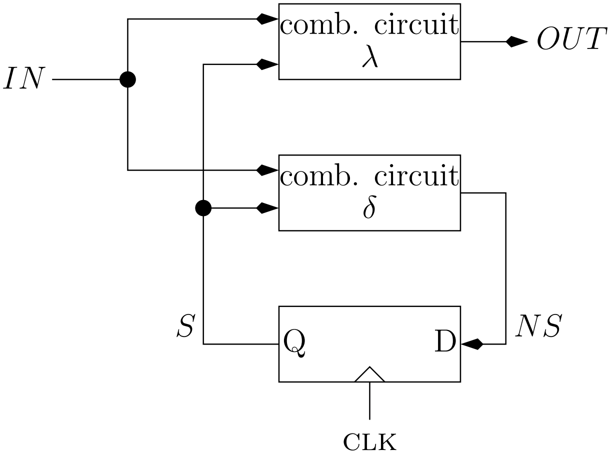
Minimum clock period: To ensure proper functionality, the clock period cannot be too short. Namely, the time $t_{i+1} - t_i$ between two consecutive rising clock edges must be longer than $t_{pd}(FF_1) + pd(C) + t_{su}(FF_2)$.

Use simple flip-flops: Inequality

$$t_{hold}(FF_2) \leq t_{cont}(FF_1) + cont(C).$$

is satisfied if $t_{cont}(FF_1) \geq t_{hold}(FF_2)$.

The Canonic Form of a Synchronous Circuit



What about input and output timing constraints ? Constraints for NS?

Input/Output Timing Constraints

Simplifying Assumption:

- Input is an output of a flip-flip.

$$stable(IN)_i \triangleq (t_i + pd(FF_{IN}), t_{i+1} + cont(FF_{IN})).$$

- Output is an input of a flip-flip.

$$stable(OUT)_i \triangleq (t_{i+1} - setup(FF_{OUT}), t_{i+1} + hold(FF_{OUT})).$$

Benefit: timing constraints same as before. Of course $pd(FF_{IN}) > cont(FF_{IN})$.

Timing constraints of internal signals.

The only constraint we have for an internal signal is that the signal *NS* that feeds a flip-flop is stable during the critical segments. Namely, for every $i \geq 0$,

$$stable(NS)_i \triangleq C_{i+1}.$$

Paths in the canonic form

When performing a timing analysis of a synchronous circuit in canonic form, we notice that there are only four maximal paths without flip-flops:

1. the path $IN \to \delta \to NS$,
2. the path $S \to \delta \to NS$,
3. the path $IN \to \lambda \to OUT$, and
4. the path $S \to \lambda \to OUT$.

We regard the signal IN to be the output of a flip-flop, and the signal OUT to be an input to a flip-flop, then we have four paths of the type studied in the simple example.

Sufficient Conditions

the timing constraints of NS are satisfied if:

$$\forall i \geq 0 : \quad \max\{pd(IN), t_{pd}(FF)\} + pd(\delta) + t_{su}(FF) \leq t_{i+1} - t_i, \text{ and}$$
$$\min\{cont(IN), t_{cont}(FF)\} + cont(\delta) \geq t_{hold}(FF).$$

the timing constraints of OUT are satisfied if:

$$\forall i \geq 0 : \quad \max\{pd(IN), t_{pd}(FF)\} + pd(\lambda) + setup(OUT) \leq t_{i+1} - t_i, \text{ and}$$
$$\min\{cont(IN), t_{cont}(FF)\} + cont(\lambda) \geq hold(OUT).$$

Claim

The timing constraints of the signals OUT and NS are satisfied if the above equations hold.

Satisfying the Timing Constrains

What do we need to do to make sure that the timing constraints of a synchronous circuit are satisfied?

- lower bounds on the clock period.
- use simple flip-flops in which $t_{cont} \geq t_{hold}$.

Initialization

- we require that the output of every flip-flop be defined and stable during the interval $(t_0 + t_{pd}(FF), t_1 + t_{cont}(FF))$.
- How is the first clock cycle $[t_0, t_1)$ defined?
- What is the state of a flip-flop after power on?
- introduce a reset signal.
- How is a reset signal generated? Why does a reset signal differ from the the output of the flip-flop? After all, the reset signal might be metastable.
- no solution to this problem within the digital abstraction. All we can try to do is reduce the probability of such an event.
- In practice, a special circuit, often called a reset controller, generates a reset signal that is stable during the first clock period with very high probability. In fact, the first clock period of the synchronous circuit is defined by the reset controller.

Specification of the reset signal

Assume that the reset signal is output by a flip-flop so that it satisfies two conditions:

$$reset(t) \triangleq \begin{cases} 1 & \text{if } t \in (t_0 + t_{pd}(FF), t_1 + t_{cont}(FF)), \\ 0 & \text{if } t > t_1 + t_{pd}(FF). \end{cases}$$

Using the reset

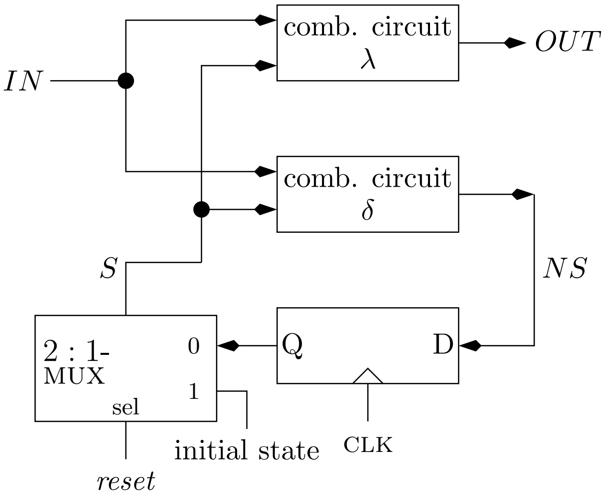
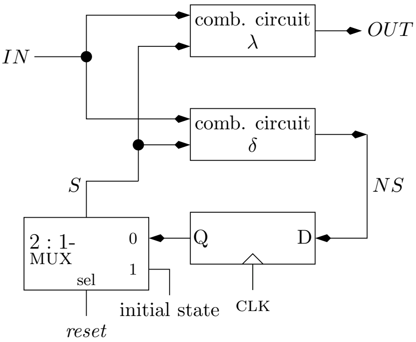


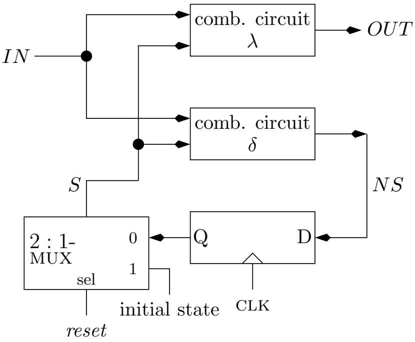
Figure: A synchronous circuit in canonic form with reset.

edge triggered flip-flop with a reset



Flip-flop with the multiplexer are encapsulated into a single module called an edge triggered flip-flop with a reset. Let FF' denote an edge triggered flip-flop with a reset, then $t_{pd}(FF') = t_{pd}(FF) + pd(\text{MUX})$ and $t_{cont}(FF') = t_{cont}(FF) + cont(\text{MUX})$. On the other hand, $t_{su}(FF') = t_{su}(FF)$ and $t_{hold}(FF') = t_{hold}(FF)$.

Using the reset



Claim

If the reset signal satisfies the specification, then $S(t)$ is stable during the interval

$$(t_0 + t_{\mathsf{pd}}(FF) + \mathsf{pd}(\textsc{mux}), t_1 + t_{\mathsf{cont}}(FF) + \mathsf{cont}(\textsc{mux})).$$

Assumptions

1. Initialization: the signal S satisfies

$$(t_0 + t_{pd}(FF), t_1 + t_{cont}(FF)) \subseteq stable(S)_0$$

2. Clock period is long enough: Let Φ denote the clock period (i.e., $\Phi = t_{i+1} - t_i$, for every $i \geq 0$). Then,

$$\max\{pd(IN), t_{pd}(FF)\} + pd(\delta) + t_{su}(FF) \leq \Phi, \text{ and}$$
$$\max\{pd(IN), t_{pd}(FF)\} + pd(\lambda) + setup(OUT) \leq \Phi.$$

3. Hold times are smaller than the contamination delays: formally, we require that:

$$\min\{cont(IN), t_{cont}(FF)\} + cont(\delta) \geq t_{hold}(FF).$$
$$\min\{cont(IN), t_{cont}(FF)\} + cont(\lambda) \geq hold(OUT).$$

Functionality: the canonic form

We denote the logical value of a signal X during the stability interval $\mathit{stable}(X)_i$ by X_i.

Claim

If the assumptions hold, then the following relations hold for every $i \geq 0$:

$$\begin{aligned} NS_i &= \delta(IN_i, S_i) \\ OUT_i &= \lambda(IN_i, S_i) \\ S_{i+1} &= NS_i. \end{aligned}$$

Finite State Machines

The functionality of a synchronous circuit in the canonic form is so important that it justifies a term called finite state machines.

Definition

A finite state machine (FSM) is a 6-tuple $\mathcal{A} = \langle Q, \Sigma, \Delta, \delta, \lambda, q_0 \rangle$, where

- Q is a set of states.
- Σ is the alphabet of the input.
- Δ is the alphabet of the output.
- $\delta : Q \times \Sigma \to Q$ is a transition function.
- $\lambda : Q \times \Sigma \to \Delta$ is an output function.
- $q_0 \in Q$ is an initial state.

What does an FSM do?

An FSM is an abstract machine that operates as follows. The input is a sequence $\{x_i\}_{i=0}^{n-1}$ of symbols over the alphabet Σ. The output is a sequence $\{y_i\}_{i=0}^{n-1}$ of symbols over the alphabet Δ. An FSM transitions through the sequence of states $\{q_i\}_{i=0}^{n}$. The state q_i is defined recursively as follows:

$$q_{i+1} \triangleq \delta(q_i, x_i)$$

The output y_i is defined as follows:

$$y_i \triangleq \lambda(q_i, x_i).$$

FSM - terminology

Other terms for a finite state machine are a finite automaton with outputs and transducer. In the literature, an FSM according to our definition is often called a Mealy Machine. Another type of machine, called Moore Machine, is an FSM in which the domain of output function λ is Q (namely, the output is only a function of the state and does not depend on the input).

State Diagrams

FSMs are often depicted using state diagrams.

Definition

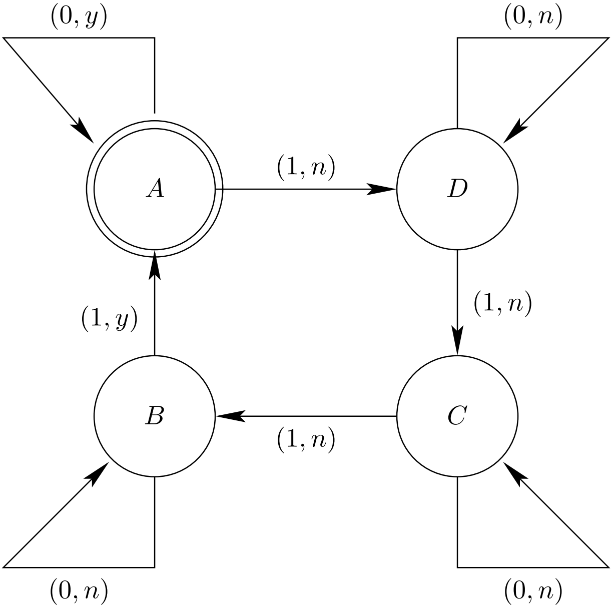
The state diagram corresponding to an FSM $\mathcal{A}$ is a directed graph $G = (Q, E)$ with edge labels $(x, y) \in \Sigma \times \Delta$. The edge set E is defined by

$$E \triangleq \{(q, \delta(q, x)) : q \in Q \text{ and } x \in \Sigma\}.$$

Each edge $(q, \delta(q, x))$ is labeled $(x, \lambda(q, x))$.

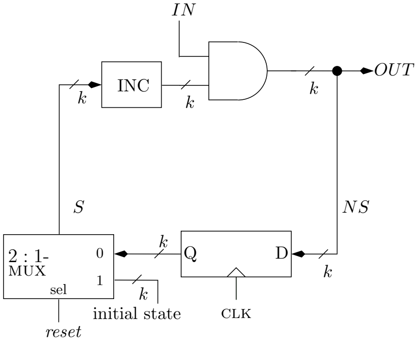
The vertex q_0 corresponding to the initial state of an FSM is usually marked in an FSM by a double circle.
We remark that a state diagram is in fact a multi-graph, namely, one allows more than one directed edge between two vertices. Such edges are often called parallel edges. Note that the out-degree of every vertex in a state diagram equals $|\Delta|$.

A state diagram of an FSM that counts (mod 4)



Timing analysis: the general case

Assume that $pd(IN) = 9$ while $t_{pd}(FF) = pd(\text{MUX}) = pd(\text{AND}) = 1$ and $t_{su}(FF) = setup(OUT) = 1$. Moreover, assume that $pd(INC) = 7$. The timing analysis in the canonic form is too pessimistic!



Timing analysis: the general case

Given a synchronous circuit C, we distinguish between four types of signals:

1. Inputs - these are signals that are fed by input gates.
2. Outputs - these are signals that are fed to output gates.
3. Inputs to the D-ports of flip-flops.
4. Outputs of flip-flops.

Timing Constraints

Input constraints: For every input signal IN, it is guaranteed that the stability intervals of IN satisfy, for every $i \geq 0$:

$$stable(IN)_i \triangleq (t_i + pd(IN), t_{i+1} + cont(IN)).$$

Output constraints: For every output signal OUT, it is required that the stability intervals of OUT satisfy:

$$stable(OUT)_i \triangleq (t_{i+1} - setup(OUT), t_{i+1} + hold(OUT)).$$

Critical segments: For every signal NS that feeds a D-port of a flip-flop, it is required that NS is stable during the critical segments, namely:

$$stable(NS)_i \triangleq C_{i+1}.$$

Feasibility of timing constraints

We say that a timing constraint of signal X is satisfied if the signal X is indeed stable during the intervals $\{stable(X)_i\}_{i \geq 0}$.

Definition

The timing constraints are feasible if there exists a clock period Φ such that all timing constraints are satisfied if $t_{i+1} - t_i = \Phi$.

Algorithms: feasibility and minimum clock period

We now present two algorithms:

1. Algorithm FEAS(C), decides whether the timing constraints of a synchronous circuit C are feasible.
2. Algorithm Min-Φ(C) computes the minimum clock period of C if the timing constraints are feasible.

For simplicity, we assume that all the flips-flops in the synchronous circuit C are identical and have the same parameters (i.e $t_{su}(FF), t_{hold}(FF), t_{cont}(FF), t_{pd}(FF)$).

Algorithm FEAS(C)

The input of algorithm FEAS(C) consists of:

1. A description of the circuit C, namely, a directed graph $G = (V, E)$ and a labeling $\pi : V \rightarrow \Gamma \cup IO \cup \{FF\}$,
2. $cont(IN)$ for every input signal IN, and
3. $hold(OUT)$ for every output signal OUT.

Algorithm 1 FEAS(C) - an algorithm that decides if the timing constraints of a synchronous circuit C are feasible.

1. Let C' denote the combinational circuit obtained from C by stripping away the flip-flops.
2. Assign weights $w(v)$ to vertices in C' as follows.

$$w(v) \triangleq \begin{cases} cont(IN) & \text{if input gate } v \text{ feeds } IN. \\ t_{cont}(FF) & \text{if } v \text{ corresponds to } Q\text{-port of a flip-flop.} \\ -hold(OUT) & \text{if output gate } v \text{ is fed by } OUT. \\ -t_{hold}(FF) & \text{if } v \text{ corresponds to } D\text{-port of a flip-flop.} \\ cont(\pi(v)) & \text{if } \pi(v) \text{ is a combinational gate.} \end{cases}$$

3. Compute

$$w^* \triangleq \min\{w(p) \mid \ p \text{ is a path from a source to a sink in } C'\}.$$

4. If $w^* \geq 0$, then return("feasible"), else return("not feasible").

Algorithm Min-$\Phi(C)$

The input of algorithm Min-$\Phi(C)$ consists of:

1. A description of the circuit C, namely, a directed graph $G = (V, E)$ and a labeling $\pi : V \to \Gamma \cup IO \cup \{FF\}$,
2. $pd(IN)$ for every input signal IN, and
3. $setup(OUT)$ for every output signal OUT.

Algorithm 2 Min-$\Phi(C)$ - an algorithm that computes the minimum clock period of a synchronous circuit C.

1. Let C' denote the combinational circuit obtained from C by stripping away the flip-flops.
2. Assign delays $d(v)$ to vertices in C' as follows.

$$d(v) \triangleq \begin{cases} pd(IN) & \text{if input gate } v \text{ feeds } IN. \\ t_{pd}(FF) & \text{if } v \text{ corresponds to } Q\text{-port of a flip-flop.} \\ setup(OUT) & \text{if output gate } v \text{ is fed by } OUT. \\ t_{su}(FF) & \text{if } v \text{ corresponds to } D\text{-port of a flip-flop.} \\ pd(\pi(v)) & \text{if } \pi(v) \text{ is a combinational gate.} \end{cases}$$

3. Compute

$$\Phi^* \triangleq \max\{d(p) \mid p \text{ is a path from a source to a sink in } C'\}.$$

4. Return(Φ^*).

Notation

Given a vertex $v \in C'$, let $c^*(v)$ denote lightest weight of a path from a source to v. Similarly, let $d^*(v)$ denote the largest delay of a path from a source to v.
Using this notation, we have a simple description of the algorithms:

- FEAS(C) decides that the timing constraints are feasible if and only if $\min_v c^*(v) \geq 0$.
- Min-$\Phi(C)$ returns $\Phi^* = \max_v d^*(v)$.

Correctness

Assume that the flip-flops are reset so that their outputs are stable during $(t_0 + t_{pd}(FF), t_1 + t_{cont}(FF))$. Assume also that the inputs satisfy the input constraints.

Claim

If $\min_v c^*(v) \geq 0$ *and* $t_{i+1} - t_i \geq \max_v d^*(v)$, *then, for every vertex* v, *every output of* v *is stable during the interval*

$$(t_i + d^*(v), t_{i+1} + c^*(v)).$$

Moreover, the inputs to flip-flops are stable during the critical segments and the output constraints are satisfied.

The proof uses double induction... (clock cycle and index of vertex in topological ordering)

The zero delay model

In the zero delay model transitions of all signals are instantaneous. This means that the propagation delay and contamination delay of combinational circuits is zero. In addition, the parameters of flip-flops satisfy:

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

We emphasize that this model is used only as a simplified model for specifying and simulating the functionality of circuits with flip-flops.

Zero delay model - cont

For simplicity, we normalize time so that the clock period is 1 time unit. That is, $t_{i+1} - t_i = 1$, for every i. This allows us to specify the functionality of a flip-flop in the zero delay model as follows:

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

The meaning of this specification is as follows. (1) The critical segment C_i equals $[t_{i-1}, t_i)$. (2) The value of $D(t)$ is stable during the critical segment $[t_{i-1}, t_i)$. This value is sampled by the flip-flop during the clock cycle $(i-1)$. In the next clock cycle $[t_i, t_{i+1})$, the flip-flop's output $Q(t)$ equals the value of the input sampled during the previous cycle.

Logical Simulation of Synchronous Circuits

Assumptions:

- Initialization: For every flip-flop FF_i, let $S_0(FF_i) \in \{0, 1\}$ denote the value output by FF_i in clock cycle $t = 0$.
- Input sequence: For every input gate X let $IN_t(X) \in \{0, 1\}$ the input fed by X in clock cycle t.

Simulation Algorithm

Algorithm 3 $\mathrm{SIM}(C, S_0, \{IN_t\}_{t=0}^{T-1})$ - An algorithm for simulating a synchronous circuit C with respect to an initialization S_0 and a sequence of inputs $\{IN_t\}_{t=0}^{T-1}$.

1. Construct the combinational circuit C' obtained from C by stripping away the flip-flops.
2. For $t = 0$ to $T - 1$ do:
   1. Simulate the combinational circuit C' with input values corresponding to S_t and IN_t. Namely, every input gate in C feeds a value according to IN_t, and every Q-port of a flip-flop feeds a value according to S_t. For every sink z in C', let z_t denote the value fed to z according to this simulation.
   2. For every Q-port S of a flip-flop, define $S_{t+1} \leftarrow NS_t$, where NS denotes the D-port of the flip-flop.

Synthesis and Analysis

Two tasks are often associated with synchronous circuits. These tasks are defined as follows.

1. Analysis: given a synchronous circuit C, describe its functionality by an FSM.
2. Synthesis: given an FSM $\mathcal{A}$, design a synchronous circuit C that implements $\mathcal{A}$.

Analysis: Sync Circuit $\mapsto$ FSM

The task of analyzing a synchronous circuit C is carried out as follows.

1. Define the FSM $\mathcal{A} = \langle Q, \Sigma, \Delta, \delta, \lambda, q_0 \rangle$ as follows.
    1. The set of states is $Q \triangleq \{0,1\}^k$, where k denotes the number of flip-flops in C.
    2. Define the initial state q_0 to be the initial outputs of the flip-flops.
    3. $\Sigma = \{0,1\}^\ell$, where ℓ denotes the number of input gates in C.
    4. $\Delta = \{0,1\}^r$, where r denotes the number of output gates in C.
    5. Define the transition function $\delta : \{0,1\}^k \times \{0,1\}^\ell \to \{0,1\}^k$ to be the function implemented by the combinational "part" of C for the inputs of the flip-flops.
    6. Define the output function $\lambda : \{0,1\}^k \times \{0,1\}^\ell \to \{0,1\}^r$ to be the function implemented by the combinational "part" of C for the output gates.

Synthesis: FSM $\mapsto$ Sync Circuit

Given an FSM $\mathcal{A} = \langle Q, \Sigma, \Delta, \delta, \lambda, q_0 \rangle$, the task of designing a synchronous circuit C that implements $\mathcal{A}$ is carried out as follows.

1. Encode Q, Σ and Δ by binary strings. Formally, let f, g, h denote one-to-one functions, where

$$\begin{aligned} f &: Q \to \{0,1\}^k \\ g &: \Sigma \to \{0,1\}^\ell \\ h &: \Delta \to \{0,1\}^r. \end{aligned}$$

2. Design a combinational circuit C_δ that implements the (partial) Boolean function $B_\delta : \{0,1\}^k \times \{0,1\}^\ell \to \{0,1\}^k$ defined by

$$B_\delta(f(x), g(y)) \triangleq f(\delta(x,y)), \text{ for every } (x,y) \in Q \times \Sigma.$$

3. Design a combinational circuit C_λ that implements the (partial) Boolean function $B_\lambda : \{0,1\}^k \times \{0,1\}^\ell \to \{0,1\}^r$

$$B_\lambda(f(x), g(z)) \triangleq h(\lambda(x,z)), \text{ for every } (x,z) \in Q \times \Delta.$$

Synthesis - cont

- How many flip-flops are required? $f : Q \to \{0,1\}^k$ is one-to-one. So

$$k \geq \log_2 |Q|$$

- It is not clear that minimizing k is a always a good idea. Certain encodings lead to more complicated Boolean functions B_δ and B_λ.
- The question of selecting a "good" encoding is a very complicated task, and there is no simple solution to this problem.

Summary

- definition of synchronous circuits.
- synchronous circuits in canonic form.
- Timing analysis.
- Initialization.
- Functionality: finite-state machines.
- Timing in the general case. Two algorithms are presented: one verifies whether the timing constraints are feasible. The second algorithm computes the minimum clock period.
- simulation algorithm.
- analysis and synthesis of synchronous circuits.