Chapter 12

Synchronous Circuits

Contents

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?

In this chapter we deal with synchronous circuits. From a functional point of view, synchronous circuits implement finite state machines. However, we use a syntactic definition and show that every circuit that obeys these syntactic rules implements a finite state machine. Correct functionality of a synchronous circuit requires satisfaction of certain timing constraints. Most importantly, all data inputs of flip-flops must be stable during the critical sections. A key advantage of the model (in which the critical section and the instability interval of each flip-flop are disjoint) is that it is possible to satisfy all the timing constraints if the clock period is sufficiently long.

To simplify the discussion, we consider a canonic form of synchronous circuits. In the canonic form, a synchronous circuit is decomposed into three parts: (i) the flip-flips store the state, (ii) a combinational circuit computes the output, and (iii) a combinational circuit computes the next state.

Finally, we deal with the issue of initialization. Loosely speaking, initialization of the circuit means that the flip-flops output correct and stable values during the first clock period.

12.1 Syntactic definition

The building blocks of a synchronous circuit are combinational gates, nets, and flip-flops. The definition of synchronous circuit considers the circuit after the flip-flops are removed. We refer to this as *stripping flip-flops away.* Formally,

Definition 12.1 *A* synchronous circuit *is a circuit C composed of combinational gates, nets, and flip-flops that satisfies the following conditions:*

1. *There is a net called* CLK *that carries a clock signal.*
2. *The* CLK *net is fed by an input gate.*
3. *The set of ports that are fed by the* CLK *net equals the set of clock-inputs of the flip-flops.*
4. *Define the circuit C' as follows: The circuit C' is obtained by (i) deleting the* CLK *net, (ii) deleting the input gate that feeds the* CLK *net, and (iii) replacing each flip-flip with an output gate (instead of the port D) and an input gate (instead of the port Q). We require that the circuit C' is combinational.*

We remark that in a synchronous circuit the clock signal is connected only to the clock port of the flip-flops; the clock may not feed other inputs (i.e. inputs of combinational gates or the D-port of flip-flops). Moreover, every clock-port of a flip-flop is fed by the clock signal.

Figure 12.1 depicts a synchronous circuit C and the corresponding combinational circuit C'.

Question 12.1 *Suggest an efficient algorithm that decides if a given circuit is synchronous.*

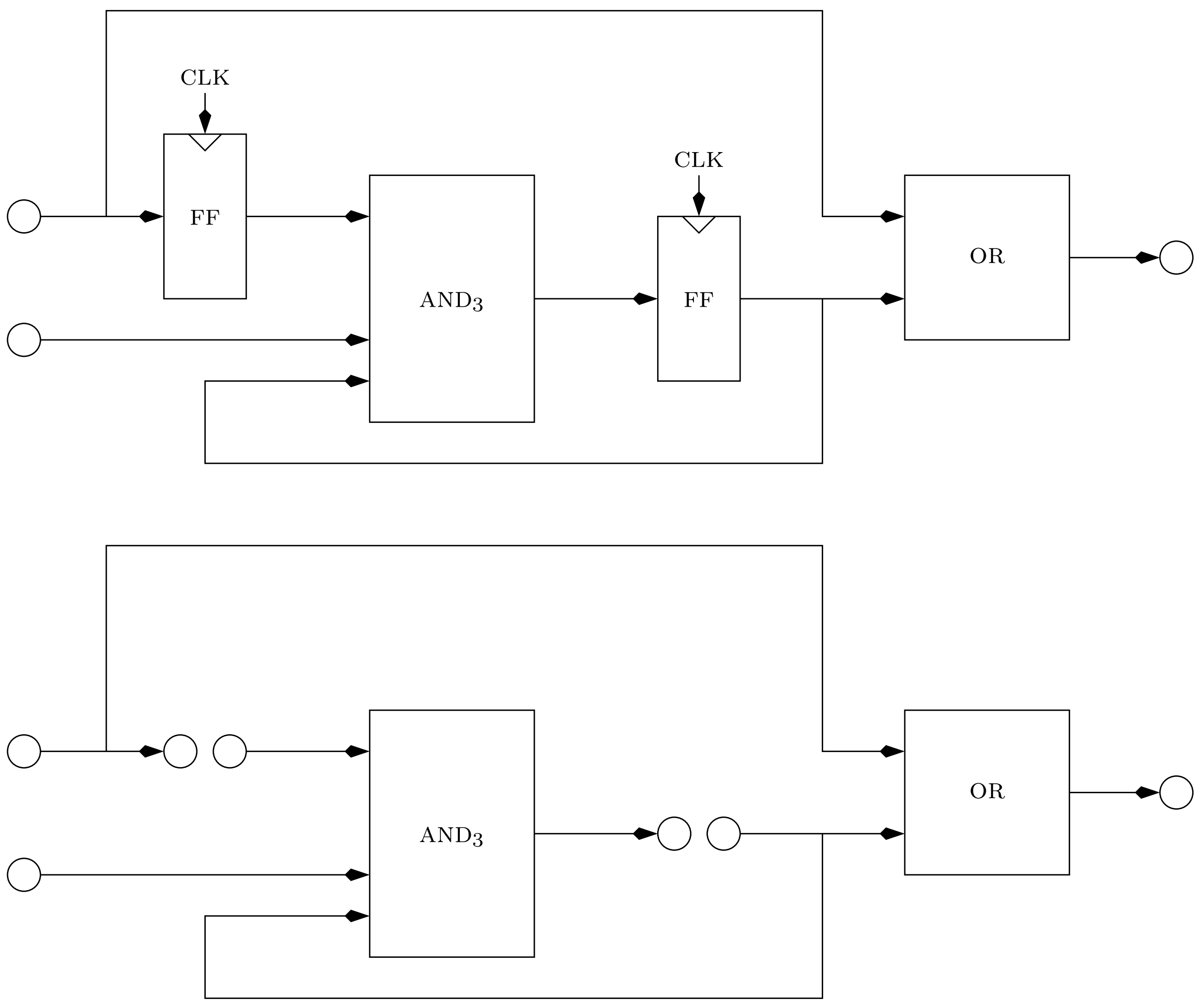

Figure 12.1: A synchronous circuit C and the corresponding combinational circuit C'.

Question 12.2 *In Definition 12.1 we required that the clock signal feed only the clock ports of flip-flops. Consider the circuit depicted in Figure 12.2 that violates this requirement. Proper functioning of the D-FF (and hence the circuit) is guaranteed only if setup and hold time requirements are satisfied. Under which conditions can the signal feeding the D port of the D-FF satisfy the setup and hold time requirements? Suppose that these conditions are met, is it valid to say that "the flip-flop samples the outcome of the combinational circuit that feeds it at the end of each cycle"?*

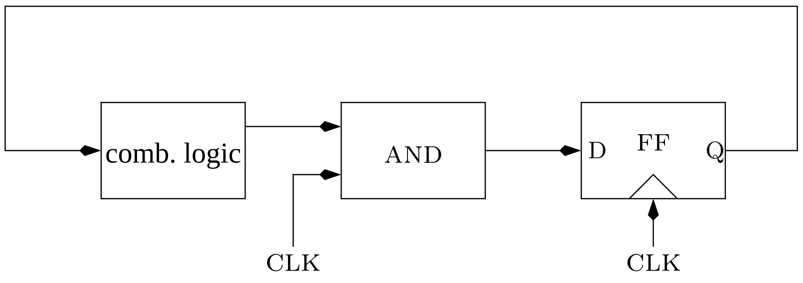

Figure 12.2: A non-synchronous circuit in which the clock feeds also a gate.

12.2 Timing analysis: the canonic form

In this section we analyze the timing constraints of a synchronous circuit that is given in canonic form.

12.2.1 Canonic form of a synchronous circuit

Consider the synchronous circuit depicted in Figure 12.3. The circuit has an input IN, and output OUT, and internal signals S (for "state") and NS (for "next state"). We abuse notation and refer to the combinational circuits λ and δ by the Boolean functions that they implement. In this example, all the signals in the circuit carry single bits (as normal signals do). However, we could easily deal with the case in which IN, OUT, S, NS are buses (i.e. multiple-bit signals).

One can transform every synchronous circuit so that it fits the description in Figure 12.3. This is achieved by: (i) gathering the flip-flops into one group and (ii) duplicating the combinational circuits (if necessary) so that we can separate between the combinational circuits that produce output signals and combinational circuits that produce signals that are fed back to the flip-flops. This is why we refer to the circuit depicted in Figure 12.3 as a *canonic form* of a synchronous circuit.

12.2.2 Timing constraints

Stability interval. We associate with each signal an interval corresponding to the ith clock cycle during which the signal is supposed to be stable. We refer to this interval as

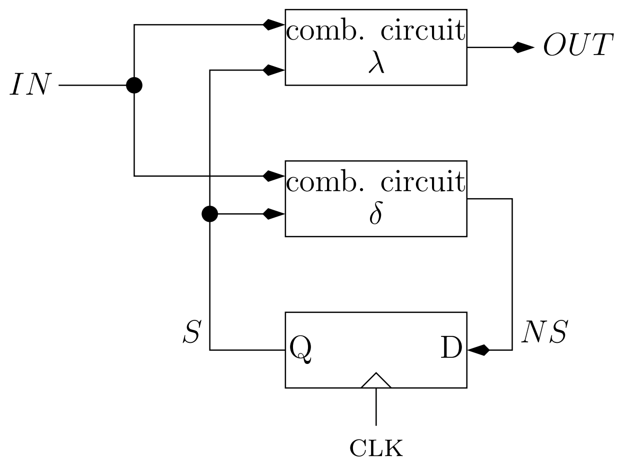

Figure 12.3: A synchronous circuit in canonic form.

the *stability interval*. We denote the stability interval corresponding to the ith interval of signal X by $stable(X)_i$. We denote the value of $dig(X)$ during the interval $stable(X)_i$ by X_i.

Input/Output timing constraints. The input/output timing constraints formulate the timing interface between the circuit and the "external world". The constraint corresponding to the input tells us when the input is guaranteed to be stable, and the constraint corresponding to the output tells us when the circuit's output is required to be stable. Usually the external world is also a synchronous circuit. This means that the signal IN is an output of another synchronous circuit. Similarly, the signal OUT is an input of another synchronous circuit. It is helpful to think of IN as the output of a flip-flop and of OUT as the input of a flip-flop.

1. The timing constraint corresponding to IN is defined by two parameters: $pd(IN) > cont(IN)$ as follows. The stability intervals of signal IN are guaranteed to satisfy

$$\forall i \geq 0 : \quad [t_i + pd(IN), t_{i+1} + cont(IN)] \subseteq stable(IN)_i. \tag{12.1}$$

   Recall that t_i denotes the starting time of the ith clock period. Note that if $pd(IN) \leq cont(IN)$, then the stability intervals $stable(IN)_i$ and $stable(IN)_{i+1}$ overlap. This means that IN is always stable, which is obviously not an interesting case.

2. The timing constraint corresponding to OUT is defined by two parameters: $setup(OUT)$ and $hold(OUT)$ as follows. The stability intervals of signal OUT must satisfy the following condition:

$$\forall i \geq 0 : \quad [t_{i+1} - setup(OUT), t_{i+1} + hold(OUT)] \subseteq stable(OUT)_i. \tag{12.2}$$

   Note that that timing constraint of OUT is given relative to the end of the ith cycle (i.e. t_{i+1}) .

Note that there is an asymmetry in the terminology regarding IN and OUT. The parameters associated with IN are $pd(IN)$ and $cont(IN)$, whereas the parameters associated with OUT are $setup(OUT)$ and $hold(OUT)$. This is not very aesthetic if OUT is itself an input to another synchronous circuit. The reason for this asymmetric choice is that it is useful to regard IN as an output of a flip-flip and OUT as an input of a flip-flop (even if they are not).

Timing constraints of internal signals. The only constraint we have for an internal signal is that NS is stable during the critical segments. Namely,

$$\forall i \geq 0 : \quad C_{i+1} \subseteq \mathit{stable}(NS)_i.$$

Note that, as in the case of the output signal, the timing constraint of NS corresponding to clock cycle i is relative to the end of the ith clock cycle (i.e. the critical segment C_{i+1}).

12.2.3 Sufficient conditions

We associate a contamination delay $cont(x)$ and a propagation delay $pd(x)$ with each combinational circuit x. The following claim follows directly from the propagation delay and contamination delay of the combinational circuits λ and δ.

Claim 12.1 *If*

$$[t_i + t_{\text{pd}}, t_{i+1} + t_{\text{cont}}] \subseteq \text{stable}(S)_i, \tag{12.3}$$

then the stability intervals of the signals OUT and NS satisfy:

$$[t_i + \max\{t_{\text{pd}}, \text{pd}(IN)\} + pd(\lambda), t_{i+1} + \min\{t_{\text{cont}}, \text{cont}(IN)\} + \text{cont}(\lambda)] \subseteq \text{stable}(OUT)_i \tag{12.4}$$

$$[t_i + \max\{t_{\text{pd}}, \text{pd}(IN)\} + pd(\delta), t_{i+1} + \min\{t_{\text{cont}}, \text{cont}(IN)\} + \text{cont}(\delta)] \subseteq \text{stable}(NS)_i. \tag{12.5}$$

The following claim provides a sufficient condition so that the timing constraint of OUT holds.

Claim 12.2 *If*

$$\begin{aligned} [t_i + t_{\text{pd}}, t_{i+1} + t_{\text{cont}}] &\subseteq \text{stable}(S)_i & (12.6)\\ \max\{t_{\text{pd}}, \text{pd}(IN)\} + pd(\lambda) + \text{setup}(OUT) &\leq t_{i+1} - t_i & (12.7)\\ \min\{t_{\text{cont}}, \text{cont}(IN)\} + \text{cont}(\lambda) &\geq \text{hold}(OUT), & (12.8)\end{aligned}$$

then the timing constraint of OUT corresponding to cycle i holds, namely,

$$[t_{i+1} - \text{setup}(OUT), t_{i+1} + \text{hold}(OUT)] \subseteq \text{stable}(OUT)_i.$$

Proof: If Equation 12.7 holds, then

$$t_i + \max\{t_{pd}, pd(IN)\} + pd(\lambda) \leq t_{i+1} - setup(OUT).$$

If Equation 12.8 holds, then

$$t_{i+1} + hold(OUT) \leq t_{i+1} + \min\{t_{cont}, cont(IN)\} + cont(\lambda).$$

By Equation 12.4 it follows that

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

□

The following claim provides a sufficient condition so that the flip-flop's input is stable during the critical segments.

Claim 12.3 *If*

$$[t_i + t_{\text{pd}}, t_{i+1} + t_{\text{cont}}] \subseteq \text{stable}(S)_i \tag{12.9}$$
$$\max\{t_{\text{pd}}, \text{pd}(IN)\} + \text{pd}(\delta) + t_{\text{su}} \leq t_{i+1} - t_i \tag{12.10}$$
$$t_{\text{hold}} \leq \min\{t_{\text{cont}}, \text{cont}(IN)\} + \text{cont}(\delta), \tag{12.11}$$

then the signal NS is stable during the critical segment C_{i+1}.

Proof: The conditions together with Equation 12.5 imply that the critical segment $C_{i+1} \subseteq stable(NS)_i$. □
Note that, in the proof of Claim 12.3, we showed that the critical segment corresponding to clock cycle $i+1$ (i.e. C_{i+1}) is contained in the stability interval of NS corresponding to cycle i (i.e. $stable(NS)_i$).

Corollary 12.4 *Assume that Equation 12.3 holds with respect to $i = 0$. Assume that Equations 12.7,12.8,12.10, and 12.11 hold. Then (i) the timing constraints of NS and OUT hold with respect to every clock cycle $i \geq 0$, and (ii) Equation 12.3 holds for every $i \geq 0$.*

Proof: The proof is by induction on i. The induction basis for $i = 0$ follows from Claims 12.2 and 12.3. The induction step for $i+1$ is proved is follows. Since NS is stable during C_{i+1}, it follows that Equation 12.3 hold for $i + 1$. We then apply Claims 12.2 and 12.3 to show that NS and OUT satisfy the timing constraints with respect to clock cycle $i + 1$. □

We point out that we are left with the assumption that the flip-flop is properly initialized so that S satisfies Equation 12.3 with respect to $i = 0$. We deal with issue of initialization in Section 12.2.6.

12.2.4 Satisfying the timing constraints

Our goal is to simplify the conditions in Claims 12.2 and 12.3 so that they are reduced to lower bounds on the clock period. This will guarantee well defined functionality provided that the clock period is large enough.

Notice first that Equations 12.6 and 12.9 are identical. This constraints hold trivially if the inputs of the flip-flips are stable during the critical segment C_i. We now focus on the other constraints.

We first address the conditions expressed in Claim 12.2. Equation 12.7 is a lower bound on the clock period. However, Equation 12.8 may not hold. This is a serious problem that can lead to failure to meet the timing constraint of OUT.

We would like to argue that, under reasonable circumstances, Equation 12.8 does hold (without having to take special care). In a typical situation the signal IN is the output of a combinational circuit, all the inputs of which are outputs of flip-flops. Assume, for simplicity, that all the flip-flops are identical. It follows that $cont(IN) \geq t_{cont}$. By the definition of the contamination delay of a combinational circuit it follows that $cont(\lambda) \geq 0$. Hence the right hand side of Equation 12.8 is at least t_{cont}. Similarly, the signal OUT feeds a combinational circuit that feeds a flip-flop. Hence $hold(OUT) \leq t_{hold}$. Since $t_{hold} < t_{cont}$, it follows that, under these assumptions, Equation 12.8 holds.

We now address the conditions expressed in Claim 12.3. Equation 12.10 sets a lower bound on $\varphi(\text{CLK})$. Equation 12.11, like Equation 12.8, might not hold. However, under the same assumptions used for Equation 12.8, the right hand side of Equation 12.11 is at least t_{cont}. Since $t_{hold} < t_{cont}$, it follows that, under these assumptions, Equation 12.11 holds.

We summarize this discussion with the following claim.

Claim 12.5 *Assume that* $\text{cont}(IN) \geq t_{\text{cont}}$ *and* $\text{hold}(OUT) \leq t_{\text{hold}}$. *If*

$$\varphi(\text{CLK}) \geq \max\{t_{\text{pd}}, \text{pd}(IN)\} + \max\{\text{pd}(\lambda) + \text{setup}(OUT), \text{pd}(\delta) + t_{\text{su}}\}$$

and Equation 12.3 holds with respect to $i = 0$, *then the timing constraints of* NS *and* OUT *hold with respect to every clock cycle* $i \geq 0$, *and Equation 12.3 holds for every* $i \geq 0$.

12.2.5 Minimum clock period

We now define the minimum clock period.

Definition 12.2 *The* minimum clock period *of a synchronous circuit C is the shortest clock period for which the timing constraints of the output signals and signals that feed the flip-flops are satisfied.*

We denote the minimum clock period of a synchronous circuit by $\varphi^*(C)$.

Claim 12.5 deals with a synchronous circuit in canonic form, and states that, under reasonable conditions,

$$\varphi^*(C) = \max\{t_{pd}, pd(IN)\} + \max\{pd(\lambda) + setup(OUT), pd(\delta) + t_{su}\}. \tag{12.12}$$

The timing analysis of synchronous circuits in canonic form is overly pessimistic. The problem is that each of the combinational circuits λ and δ is regarded as a "gate" with a propagation delay. In practice it may be the case, for example, that the accumulated delay from the input IN to the output OUT is significantly different than the accumulated delay from S to the output OUT. The situation is even somewhat more complicated in the case of multi-bit signals. In the next section we deal with the general case.

12.2.6 Initialization

Meeting the timing constraints relies on the circuit being properly initialized. Specifically, we require that

$$[t_0 + t_{pd}, t_1 + t_{cont}] \subseteq \mathit{stable}(S)_0. \tag{12.13}$$

If we consider the state of a circuit (shortly) after power is turned on, then the definition of a flip-flop does not guarantee anything about the values of S (the flip-flop's output).

The natural solution to the problem of initialization is to introduce a reset signal. There are other situations where resetting the circuit is desirable. For example, a human user presses a reset button or the operating system decides to reset the system. However, the situation after power-up is more complicated.

Here we are confronted with a boot-strapping problem: How is a reset signal generated? Why does a reset signal differ from the signal S? After all, the reset signal might be stuck in a non-logical value. How do we guarantee that the reset signal is stable for a long enough period so that it can be used to initialize the circuit?

Not surprisingly, there is no solution to this problem within the digital abstraction. The reason is that a circuit attempting to generate a reset signal (or any digital signal) may be in a meta-stable state. All we can try to do is reduce the probability of such an event.

We have already discussed two methods to reduce the probability of meta-stability: (i) allow slow decisions and (ii) increase the "slope" (i.e. the derivative of the energy). Slowing down the decision is achieved by using a slow clock (e.g. clock period of 10^{-3} seconds) in the circuit that generates the reset signal. Increasing the slope is achieved by cascading (i.e connecting in series) edge-triggered flip-flops. In practice, a special circuit, often called a reset controller, generates a reset signal that is guaranteed to be stable during the critical segment of the flip-flop. We then use a flip-flop with a reset signal as depicted in Figure 12.4.[1]

[1]We must take into account the possibility that the signal NS is not logical or stable during reset. The implementation of the MUX that selects between the initial state (a constant string) and NS should be such that if $\mathit{reset} = 1$, then the MUX outputs the initial state even if the input NS is not logical. Implementation based on drivers has this property, while implementation based on combinational gates may not have this property.

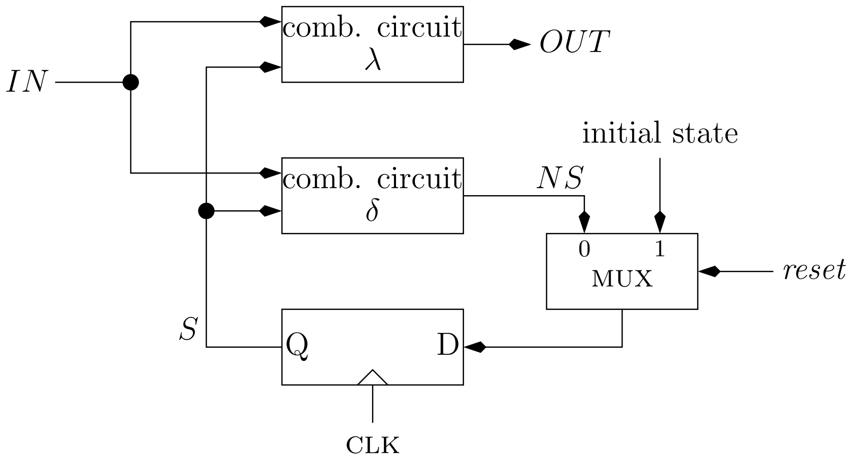

Figure 12.4: A synchronous circuit in canonic form with reset.

12.2.7 Functionality

We denote the logical value of a signal X during the stability interval *stable*$(X)_i$ by X_i. The following corollary implies that if the clock period is sufficiently large, then the functionality of the circuit is well defined. As in the case of combinational circuits (i.e. the Simulation Theorem of Combinational Circuits), we are able to derive well-defined functionality from syntax.

Corollary 12.6 *Under the premises of Claim 12.5, the following relations hold for every* $i \geq 0$*:*

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

Question 12.3 *Prove Corollary 12.6.*

Finite State Machines. We now show that Corollary 12.6 states that synchronous circuits implement *finite state machines.*

Definition 12.3 *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.
- $\lambda : Q \times \Sigma \to Q$ *is an* output function.

- $q_0 \in Q$ *is an* initial state.

Other terms for a finite state machine are a *finite automaton with outputs* and *transducer*. In the literature, an FSM according to Definition 12.3 is often called a *Mealy Machine*. Another type of machine, called *Moore Machine*, is an FSM in which the output function λ is only a function of the state and does not depend on the input.

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:

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

State Diagrams. FSMs are often depicted using state diagrams.

Definition 12.4 *The* state diagram *corresponding to an FSM $\mathcal{A}$ is a directed graph $G = (V, E)$ with edge labels $(x, y) \in \Sigma \times \Delta$. The vertex set V equals the state set S. The edge set E is defined by*

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

An 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. Figure 12.5 depicts a state diagram of an FSM that outputs y if the weight of the input so far is divisible by 4, and n otherwise.

12.3 Timing analysis: the general case

In this section we present the timing constraints of a synchronous circuit that is not in canonic form. We present an algorithm that, given a synchronous circuit C, computes the minimum clock period $\varphi^*(C)$. We also present an algorithm that decides whether the timing constraints are feasible (i.e. whether there exists a minimum clock period); the conditions used by this algorithm are less restrictive than the conditions used in Claim 12.5.

12.3.1 Timing constraints

The timing constraints of general synchronous circuits are identical to those of the canonic form. For completeness, we list them below:

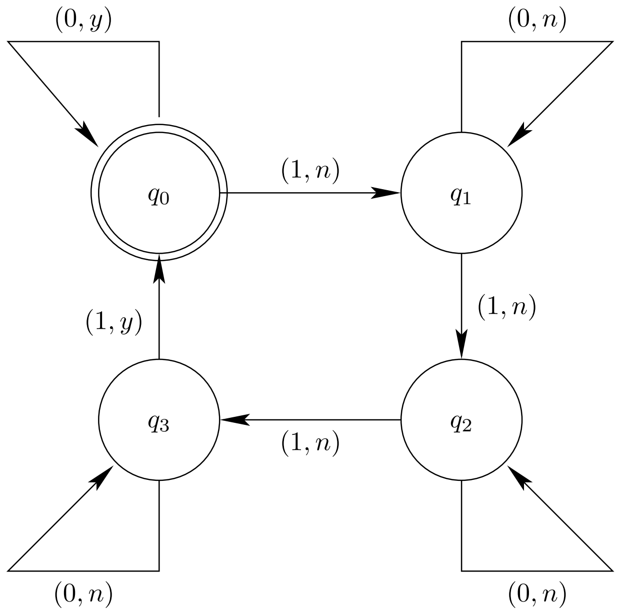

Figure 12.5: A state diagram of an FSM that counts mod$(\cdot, 4)$.

Input constraints: For every input signal IN, it is guaranteed that the stability intervals of IN satisfy:

$$\forall i \geq 0 : \quad [t_i + pd(IN), t_{i+1} + cont(IN)] \subseteq stable(IN)_i. \tag{12.14}$$

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

$$\forall i \geq 0 : \quad [t_{i+1} - setup(OUT), t_{i+1} + hold(OUT)] \subseteq stable(OUT)_i. \tag{12.15}$$

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:

$$C_{i+1} \subseteq stable(NS)_i. \tag{12.16}$$

12.3.2 Algorithm: minimum clock period

We now present an algorithm that computes the minimum clock period of a synchronous circuit. The correctness of the algorithm is based on the same assumptions used in Claim 12.5 to insure that the timing constraints are feasible.

The input of the algorithm consists of (i) a description of the circuit C, (ii) $pd(IN)$ for every input signal IN, (iii) $setup(OUT)$ for every output signal OUT, and (iv) a propagation delay $pd(v)$ for every combinational gate v. For simplicity, we assume that all the flips-flops are identical and have the same parameters (i.e $t_{su}, t_{hold}, t_{cont}, t_{pd}$).

The algorithm proceeds as follows:

1. Let C' denote the combinational circuit $C' = \langle \mathcal{G}, \mathcal{N} \rangle$ obtained from C by deleting the clock net and replacing each flip-flop with a pair of input/output gates. (See Definition 12.1.)

2. Attach a delay $d(v)$ to every input gate v of C' as follows:
$$d(v) \triangleq \begin{cases} pd(IN) & \text{if } v \text{ is an input gate of } C \text{ and } v \text{ feeds the input signal } IN \\ t_{pd} & \text{if } v \text{ corresponds to a } Q\text{-port of a flip-flop.} \end{cases}$$

3. Attach a delay $d(v)$ to every output gate v of C as follows:
$$d(v) \triangleq \begin{cases} setup(OUT) & \text{if } v \text{ is an output gate of } C \text{ and } v \text{ is fed by the output signal } OUT \\ t_{su} & \text{if } v \text{ corresponds to a } D\text{-port of a flip-flop.} \end{cases}$$

4. Attach a delay $d(v) = pd(v)$ to every combinational gate v of C.

5. Let $DG(C')$ denote the directed acyclic graph (DAG) that corresponds to C'. Let p' denote the longest path in $DG(C')$ with respect to the delays $d(v)$. Return $\varphi^*(C) = d(p')$.

The algorithm reduces the problem of computing the minimum clock period to the problem of computing a longest path in a DAG. Since a longest path in a DAG is computable in linear time, the algorithm runs in linear time as well.

12.3.3 Algorithm: correctness

Our goal is to prove that the algorithm computes the minimum clock period. Since the minimum clock period does not always exist, we need to add some conditions to guarantee its existence.

We first define delays $c(v)$ to non-sink vertices in $DG(C')$ as follows.
$$c(v) \triangleq \begin{cases} cont(IN) & \text{if } v \text{ is an input gate of } C \text{ and } v \text{ feeds the input signal } IN. \\ t_{cont} & \text{if } v \text{ corresponds to a } Q\text{-port of a flip-flop.} \\ cont(v) & \text{if } v \text{ is a combinational gate in } C. \end{cases}$$

We do not assign delays $c(v)$ to sinks because we do not need them in the following lemma that proves when signals are stable in C.

Lemma 12.7 *Consider a combinational gate, an input gate, or a flip-flop v in the synchronous circuit C. Let $\mathcal{P}_v$ denote the set of all directed paths in the directed acyclic graph $DG(C')$ that begin at a source and end in v. If the output of every flip-flop is stable in the interval $[t_i + t_{\text{pd}}, t_{i+1} + t_{\text{cont}}]$, then every output N of v satisfies*
$$[t_i + \max_{p \in \mathcal{P}_v} d(p), t_{i+1} + \min_{p \in \mathcal{P}_v} c(p)] \subseteq \text{stable}(N)_i. \tag{12.17}$$

Proof: Let $\{v_0, \ldots, v_{n-1}\}$ denote an ordering of the vertices of $DG(C')$ in topological order. Let $v = v_j$. We prove Equation 12.17 by induction on j. The induction basis, for $j = 0$, has two cases: (i) If v is an input gate, then Equation 12.17 is simply the input constraint corresponding to v. (ii) If v is a flip-flop, then the induction basis is exactly the assumption on the output of flip-flops.

The induction step is proved as follows. If v is an input gate or a flip-flop, then the proof is identical to the proof of the induction basis. We are left to deal with the case that v is a combinational gate. If every input N' of v_{j+1} satisfies Equation 12.17, then every output N of v_{j+1} satisfies: (i) N becomes stable at most $d(v_{j+1})$ time units after its last input becomes stable, and (ii) N remains stable at least $c(v_{j+1})$ time units after its first input becomes instable. The lemma follows. □

The following claim shows that if the outputs of the flip-flops are stable during the ith clock cycle, then clock period computed by the algorithm is the smallest clock period that insures that the timing constraints of the ith clock cycle are satisfied.

Claim 12.8 *Suppose that: (i) for every signal fed by a Q-port of a flip-flop,* $[t_i + t_{\text{pd}}, t_{i+1} + t_{\text{cont}}] \subseteq \text{stable}(S)_i$*, (ii) for every input* IN*,* $\text{cont}(IN) \geq t_{\text{cont}}$*, and (iii) for every output* OUT*,* $\text{hold}(OUT) \leq t_{\text{hold}}$*. Then,*

1. *For every clock period* $\varphi(\text{CLK}) \geq \varphi^*(\text{CLK})$*, the signals feeding D-ports of flip-flops are are stable during the critical segment* C_{i+1}*.*

2. *For every clock period* $\varphi(\text{CLK}) \geq \varphi^*(\text{CLK})$*, the output timing constraints corresponding to cycle* i *are satisfied.*

3. *For every clock period* $\varphi(\text{CLK}) < \varphi^*(\text{CLK})$*, a violation of the timing constraints is possible.*

Proof: Assume that $\varphi(\text{CLK}) \geq \varphi^*(\text{CLK})$. We first prove that the signals feeding the D-ports are stable during the critical segments. Consider a D-port of a flip-flop v that is fed by u. By Lemma 12.7, the output of u (i.e. the signal feeding the D-port of v) is stable during the interval

$$[t_i + \max_{p \in \mathcal{P}_u} d(p), t_{i+1} + \min_{p \in \mathcal{P}_u} c(p)].$$

Since

$$\begin{aligned}\varphi(\text{CLK}) &\geq \max_{p \in \mathcal{P}_v} d(p) \\ &= d(v) + \max_{p \in \mathcal{P}_u} d(p)\end{aligned}$$

and $d(v) = t_{su}$, we conclude that

$$t_{i+1} - t_i = \varphi(\text{CLK}) \geq t_{su} + \max_{p \in \mathcal{P}_u} d(p).$$

This implies that the signal feeding the D-port of v is stable starting at

$$t_i + \max_{p \in \mathcal{P}_u} d(p) \leq t_{i+1} - t_{su}. \tag{12.18}$$

Hence, the setup-time constraint is satisfied.

The signal feeding the D-port of v is stable until $t_{i+1} + \min_{p \in \mathcal{P}_u} c(p)$. However, every path $p \in \mathcal{P}_u$ begins at a source. A source may correspond to an input gate in C or a Q-port of a flip-flop. Since $cont(IN) \geq t_{cont}$, we conclude that $c(s) \geq t_{cont}$, for every source s. It follows that

$$\min_{p \in \mathcal{P}_u} c(p) \geq t_{cont} > t_{hold}. \tag{12.19}$$

Hence the signal feeding the D-port of v is stable during the critical segment C_i, as required.

We now prove that the output constraints are satisfied. Let OUT denote an output gate. The same argumentation as in Equation 12.18 gives

$$t_i + \max_{p \in \mathcal{P}_u} d(p) \leq t_{i+1} - setup(OUT).$$

The same argumentation as in Equation 12.18 gives $\min_{p \in \mathcal{P}_u} c(p) > t_{hold}$. Since $hold(OUT) \leq t_{hold}$, it follows that the output constraints are satisfied as well.

To prove the second part, assume that $\varphi(\text{CLK}) < \varphi^*(\text{CLK})$. Let p denote a longest path in $DG(C')$ with respect to lengths $d(v)$. Without loss of generality, p begins at a source and ends in a sink v. Let p' denote the path obtained from p by omitting the sink v. It follows that

$$t_i + d(p') > t_{i+1} - d(v).$$

If the actual propagation delays along p are maximal, then the signal feeding v is not stable at time $t_{i+1} - d(v)$. If v is a flip-flop, then its input is not stable during the critical segment. If v is an output gate, then its input does not meet the output constraint. The claim follows. □

The following corollary shows that if the circuit is properly initialized, then the clock period computed by the algorithm is the shortest clock period that satisfies all the timing constraints for all clock cycles i, for $i \geq 0$.

Corollary 12.9 *Suppose that: (i) for every signal S fed by a Q-port of a flip-flop, $[t_0 + t_{\text{pd}}, t_1 + t_{\text{cont}}] \subseteq \text{stable}(S)_0$, (ii) for every input IN, $\text{cont}(IN) \geq t_{\text{cont}}$, and (iii) for every output OUT, $\text{hold}(OUT) \leq t_{\text{hold}}$. Then,*

1. *For every clock period $\varphi(\text{CLK}) \geq \varphi^*(\text{CLK})$, the signals feeding D-ports of flip-flops are are stable during every critical segment C_{i+1}, for $i \geq 0$.*

2. *For every clock period $\varphi(\text{CLK}) \geq \varphi^*(\text{CLK})$, the output timing constraints corresponding to cycle i are satisfied, for every $i \geq 0$.*

3. *For every clock period $\varphi(\text{CLK}) < \varphi^*(\text{CLK})$, a violation of the timing constraints is possible.*

Proof: Proof is by induction on the clock cycle i. $\square$

12.3.4 Algorithm: feasibility of timing constraints

We showed in Claim 12.8 that, under reasonable assumptions, $\phi^*(\text{CLK})$ is indeed the minimum clock period. What do we do if these assumptions do not hold? For example, what should we do if $cont(IN) < t_{cont}$? Obviously, we need to rely on the contamination delays of the combinational gates along paths that lead to the flip-flop. In this section we present an algorithm that verifies whether the timing constraints are feasible.

Lemma 12.7 gives a recipe for checking the feasibility of the timing constraints. For every non-sink v in C', the guaranteed stability interval of the signals that are output by v is:

$$[t_i + \max_{p \in \mathcal{P}_v} d(p), t_{i+1} + \min_{p \in \mathcal{P}_v} c(p)].$$

The algorithm for computing $\varphi^*(C)$ deals with making sure that each such interval does not start too late (e.g. if the signal feeds a flip-flop, then it is stable not later than $t_{i+1} - t_{su}$). We need to show that each such interval does not end too early. Namely, we need to show that:

1. For every u that feeds a D-port of a flip-flop, require
$$\min_{p \in \mathcal{P}_u} c(p) \geq t_{hold}. \tag{12.20}$$

2. For every u that generates an output signal OUT, require
$$\min_{p \in \mathcal{P}_u} c(p) \geq hold(OUT). \tag{12.21}$$

If either Equation 12.20 or Equation 12.21 does not hold, then the timing constraints are infeasible, and a minimum clock period does not exist. If these equations hold, then we may omit the assumptions used in Claim 12.8.

Lemma 12.7 also suggests an algorithm to check whether Equations 12.20 and 12.21 hold. The algorithm simply computes for every non-sink $v \in DG(C')$ the value $\min_{p \in \mathcal{P}_v} c(p)$. This is simply computing a shortest path in a DAG and can be done in linear time (e.g. using depth first search). After these values are computed for all the non-sinks, the algorithm simply checks Equation 12.20 for every D-port and Equation 12.21 for every output. If a violation is found, then the timing constraints are infeasible.

12.4 Functionality

We started with a syntactic definition of a synchronous circuit. We then attached timing constraints to the inputs and outputs of synchronous circuit. For a given synchronous circuit C with input/output timing constraints, we differentiate between two cases:

- The timing constraints are infeasible. If this happens, then one cannot guarantee well defined functionality of C. For example, if the timing constraints are not met, then inputs of flip-flops might not be stable during the critical segments, and then the flip-flop output is not guaranteed to be even logical.
- The timing constraints are feasible. If this happens, then we know that the functionality is well defined provided that the clock period satisfies $\varphi(\text{CLK}) \geq \varphi^*(\text{CLK})$.

In this section we deal with the functionality of synchronous circuits when the timing constraints are feasible. We present a trivial timing model called the *zero delay model*. In this model, time is discrete and in each clock cycle, the circuit is reduced to a combinational circuit. The advantage of this model is that it decouples timing issues from functionality and enables simple logical simulations.

12.4.1 The zero delay model

In the zero delay model we assume that all the parameters of all the components are zero (i.e. $t_{su} = t_{hold} = t_{cont} = t_{pd} = 0$, $pd(IN) = cont(IN) = setup(OUT) = hold(OUT) = 0$, and $d(G) = 0$, for every combinational gate G). Under this unrealistic assumption, the timing constraints are feasible.

By Lemma 12.7, it follows that, in the zero delay model, the stability interval of every signal is $[t_i, t_{i+1})$. Following Corollary 12.6, we conclude that, for every signal X, X_i is well defined (recall, that X_i equals $dig(X)$ during the interval $stable(X)_i$).

12.4.2 Simulation

Simulation of synchronous circuit during cycles $i = 0, \ldots, n-1$ in the zero propagation model proceeds as follows:

We assume that the flip-flops are initialized. Let S_0 denote the ordered set of values initially stored by the flip-flops.

1. Construct the combinational circuit C' that corresponds to C.
2. For $i = 0$ to $n-1$ do:
    (a) Simulate the combinational circuit C' with input values corresponding to S_i and IN_i. Namely, every input gate in C feeds a value according to IN_i, and every Q-port of a flip-flop feeds a value according to S_i.
    (b) For every output OUT, let y denote the value that is fed to y. We set $OUT_i = y$.
    (c) For every D-port NS of a flip-flop, let y denote the value that is fed to the flip-flop. We set $NS_i = y$.
    (d) For every Q-port S of a flip-flop, define $S_{i+1} \leftarrow NS_i$, where NS denotes the D-port of the flip-flop.

Question 12.4 *Prove that if the premises of Coro. 12.9 are satisfied and $\phi(\text{CLK}) \geq \phi^*(\text{CLK})$, then the simulation algorithm for the zero delay model is correct.*

12.5 Summary

In this chapter we started with a syntactic definition of synchronous circuits. We then turned to analyze synchronous circuits in canonic form. Every synchronous circuits can be transformed to the canonic form. However, from a timing point of view, the canonic form is not general and may lead to overly pessimistic estimates of the minimum clock period.

Our analysis of the canonic form included the definition of timing constraints and the formulation of sufficient conditions for satisfying them. These conditions are simplified by relying on the assumption that the input originates from a flip-flop and the output is eventually fed to a flip-flop.

We defined the minimum clock period of a synchronous circuit. The minimum clock period exists only if the timing constraints are feasible. We then turned to the issue of initializing the values stored in the flip-flops so that the computation of the synchronous circuit could properly begin. We then showed that a synchronous circuit in canonic form with feasible time constraints implements a finite state machine.

In Section 12.3 we turned to the more general case of synchronous circuit that are not in canonic form. Two algorithms are presented. The first algorithm works under the assumption that the timing constraints are feasible. The algorithm computes the minimum clock period. The second algorithm verifies whether the timing constraints are feasible. Both algorithms are very simple and run in linear time.

Finally, in Section 12.4, we present the zero delay model. In this simplified delay model, time is discrete and functionality is decoupled from timing issues. We conclude with a simulation algorithm that works under the zero delay model.