Digital Logic Systems

Recitation 12: Foundations of Synchronous Circuits & Synchronous Modules

Guy Even   Moti Medina

School of Electrical Engineering Tel-Aviv Univ.

January 17, 2012

The Canonic Form of a Synchronous Circuit

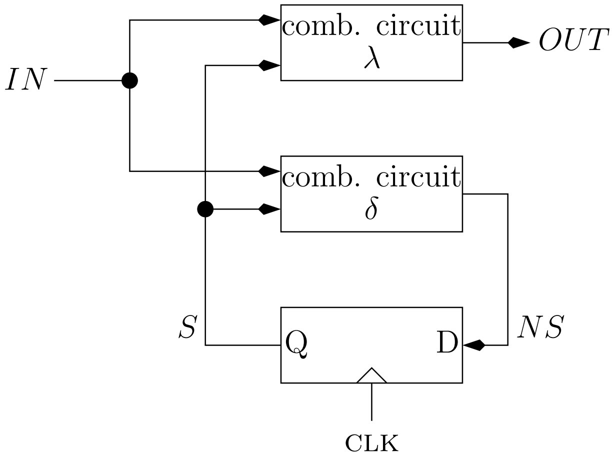

Figure: A synchronous circuit in canonic form.

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*.

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 1 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($\Phi^*$).
```

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.

Simulation

We assume that the flip-flops are initialized. Formally, let F denote the set of flip-flops in the synchronous circuit. Let $S_0 : F \rightarrow \{0, 1\}$ denote a function that specifies the initial values of each flip-flop. Let I denote the set of input gates in the synchronous circuit. Let $IN_i : I \rightarrow \{0, 1\}$ denote a function that specifies the input fed by each input gate in clock cycle i.

Simulation Algorithm

Algorithm 2 SIM$(C, S_0, \{IN_i\}_{i=0}^{n-1})$ - An algorithm for simulating a synchronous circuit C with respect to an initialization S_0 and a sequence of inputs $\{IN_i\}_{i=0}^{n-1}$.

```
1. Construct the combinational circuit $C'$ obtained from $C$ by
   stripping away the flip-flops.
2. For $i = 0$ to $n - 1$ do:
    1. 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 $S_i$. For every sink $z$ in $C'$, let $z_i$
       denote the value fed to $z$ according to this simulation.
    2. 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.
```

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

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. Transform C to a functionally equivalent synchronous circuit $\tilde{C}$ in canonic form. Compute the truth tables of the combinational circuits λ and δ. Define the Boolean functions according to these truth tables.

Synthesis

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
$$\begin{aligned} f :& Q \rightarrow \{0,1\}^k \\ g :& \Sigma \rightarrow \{0,1\}^\ell \\ h :& \Delta \rightarrow \{0,1\}^r. \end{aligned}$$
2. Design a combinational circuit C_δ that implements the (partial) Boolean function $B_\delta : \{0,1\}^k \times \{0,1\}^\ell \rightarrow \{0,1\}^k$ defined by
$$B_\delta(f(x), g(y)) \triangleq f(\delta(x,y)), \text{ for every } (x,y) \in Q \times \Sigma.$$

Synthesis (cont.)

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

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

4. Let C denote the synchronous circuit in canonic form constructed from k flip-flops and the combinational circuits C_δ for the next state and C_λ for the output.

Synthesis - cont

Let C denote the synchronous circuit in canonic form constructed from k flip-flops and the combinational circuits C_δ for the next state and C_λ for the output.
The description of the encoding step leaves a great deal of freedom. Since $|\{0,1\}^k| \geq |Q|$, it follows that $k \geq \log_2 |Q|$, and similar bounds apply to ℓ and r. However, it is not clear that using the smallest lengths is the best idea. Certain encodings lead to more complicated Boolean functions B_δ and B_λ. Thus, the question of selecting a "good" encoding is a very complicated task, and there is no simple solution to this problem.

Example: A two-state FSM

Consider the FSM $\mathcal{A} = \langle Q, \Sigma, \Delta, \delta, \lambda, q_0 \rangle$ depicted in the next figure, where

$$Q = \{q_0, q_1\},$$
$$\Sigma = \Delta = \{0, 1\}.$$

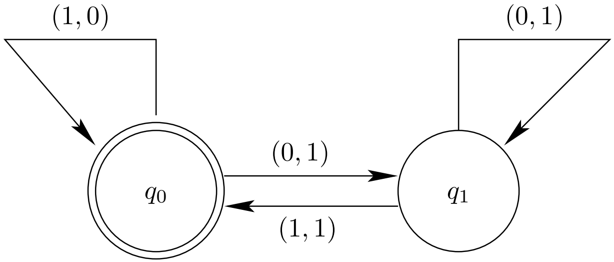

Figure: A two-state FSM.

Two-State FSMs: Synthesis

Given an FSM $\mathcal{A} = \langle Q, \Sigma, \Delta, \delta, \lambda, q_0 \rangle$, the synchronous circuit C that is obtained by executing the synthesis procedure is as follows. We encode Q, Σ and Δ by binary strings Formally, let f, g, h denote one-to-one functions, where

$$\begin{aligned} f &: Q \to \{0,1\} \\ g &: \Sigma \to \Sigma \\ h &: \Delta \to \Delta, \end{aligned}$$

where

$$f(q_0) = 0, f(q_1) = 1,$$

and

$$\forall x \in \{0,1\} : g(x) = h(x) = x.$$

Two-State FSMs: Synthesis - C_δ

We design a combinational circuit C_δ that implements the Boolean function $B_\delta : \{0,1\}^2 \to \{0,1\}$ defined by

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

$f(x)$	$g(y)$	$f(\delta(x,y))$
0	0	1
1	0	1
0	1	0
1	1	0

Table: The truth table of B_δ.

It follows that $B_\delta(f(x), g(y)) = \text{NOT}(g(y))$.

Two-State FSMs: Synthesis - C_λ

We design a combinational circuit C_λ that implements the Boolean function $B_\lambda : \{0,1\}^2 \to \{0,1\}$ defined by

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

$f(x)$	$g(y)$	$h(\lambda(x,y))$
0	0	1
1	0	1
0	1	0
1	1	1

Table: The truth table of B_λ.

It follows that $B_\lambda(f(x), g(y)) = f(x) \vee \overline{g(y)}$.

Two-State FSMs: Synthesis - the Synch. circuit C

The synchronous circuit in canonic form constructed from a flip-flops and three combinational circuits is depicted in Figure 3.

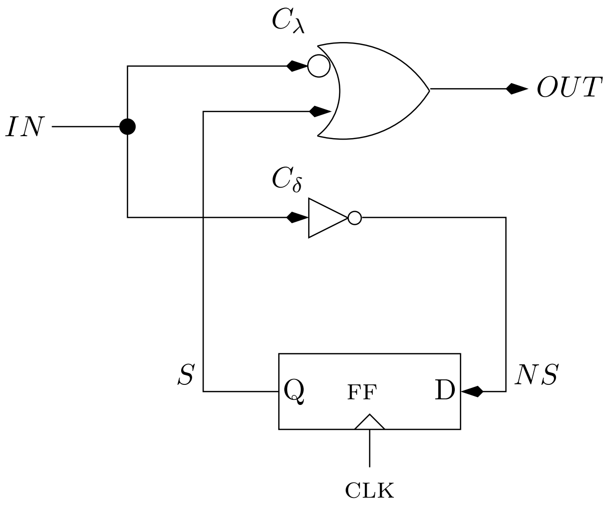

Figure: The output of the synthesis procedure on $\mathcal{A}$.

Sequential Adder

Definition

A *sequential adder* is defined as follows.

Inputs: A, B, *reset* and a clock signal CLK, where $A_i, B_i, reset_i \in \{0,1\}$.

Output: S, where $S_i \in \{0,1\}$.

Functionality: The *reset* signal is an initialization signal that satisfies:

$$reset_i = \begin{cases} 1 & \text{if } i = 0, \\ 0 & \text{if } i > 0. \end{cases}$$

Then, for every $i \geq 0$,
$\langle A[i:0] \rangle + \langle B[i:0] \rangle = \langle S[i:0] \rangle \pmod{2^{i+1}}$.

Sequential Adder (cont.)

What happens if the value of the input *reset* equals 1 in more than once cycle? The above definition means that if $reset_i = 1$, then we forget about the past, we treat clock cycle (t_i, t_{i+1}) as the first clock cycle.
Formally, we define the last initialization $r(i)$ as follows:

$$r(i) \stackrel{\triangle}{=} \max\{j \leq i : reset_j = 1\}.$$

Namely, $r(i)$ specifies the last time $reset_j = 1$ not after cycle i. If $reset_j = 0$, for every $j \leq i$, then $r(i)$ is not defined, and functionality is unspecified. If $r(i)$ is well defined, then the specification is that, for every $i \geq 0$,

$$\langle A[i : r(i)] \rangle + \langle B[i : r(i)] \rangle = \langle S[i : r(i)] \rangle \pmod{2^{i+1}}.$$

Sequential Adder: Implementation

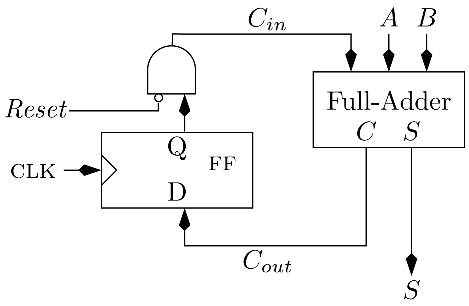

Figure: A synchronous circuit that implements a sequential adder.

Sequential Adder: Implementation - correctness

Theorem

$$\sum_{j=0}^{i} A_j \cdot 2^j + \sum_{j=0}^{i} B_j \cdot 2^j = \sum_{j=0}^{i} S_j \cdot 2^j + c_{out}(i) \cdot 2^{i+1} .$$

Proof.

The proof is by induction on i. The induction basis for $i = 0$ follows from the functionality of the full-adder:

$$A_0 + B_0 + C_{in}(0) = 2 \cdot C_{out}(0) + S_0 .$$

□

Sequential Adder: Implementation - correctness (cont.)

Proof.

We now prove the induction step for $i > 0$.

$$\begin{aligned}
\sum_{j=0}^{i} A_j \cdot 2^j + \sum_{j=0}^{i} B_j \cdot 2^j &= (A_i + B_i) \cdot 2^i + \sum_{j=0}^{i-1} A_j \cdot 2^j + \sum_{j=0}^{i-1} B_j \cdot 2^j \\
&= (A_i + B_i) \cdot 2^i + \sum_{j=0}^{i-1} S_j \cdot 2^j + C_{out}(i-1) \cdot 2^i \\
&= (C_{in}(i) + A_i + B_i) \cdot 2^i + \sum_{j=0}^{i-1} S_j \cdot 2^j \\
&= (S_i + 2 \cdot C_{out}(i)) \cdot 2^i + \sum_{j=0}^{i-1} S_j \cdot 2^j \\
&= \sum_{j=0}^{i} S_j \cdot 2^j + C_{out}(i) \cdot 2^{i+1}.
\end{aligned}$$

Sequential Adder: Analysis

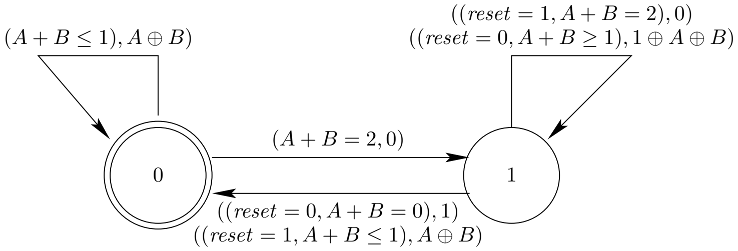
$(A+B \leq 1), A \oplus B)$
$((reset = 1, A+B = 2), 0)$
$((reset = 0, A+B \geq 1), 1 \oplus A \oplus B)$
$(A+B = 2, 0)$
0
1
$((reset = 0, A+B = 0), 1)$
$((reset = 1, A+B \leq 1), A \oplus B)$

Figure: an FSM of a sequential adder (each transition is labeled by a pair: the condition that the input satisfies and the value of the output).

Sequential Adder: Executing Min-$\Phi(C)$.

Let C denote the Sequential Adder that we have just implemented.

Assume that all the parameters equal to '1'. Assume that a full adder is implemented by a single gate.

Execute the Min-$\Phi(C)$ algorithm. Note that the *reset* signal is also an input signal, hence we should assign a weight to the input gate that feeds it as well.

The "heaviest" path is of weight 5 (*reset* input gate $\rightarrow$ NOT gate $\rightarrow$ AND gate $\rightarrow$ Full adder gate $\rightarrow$ the output gate the corresponds to the D port), hence $\varphi^* = 5$.