Chapter 12: Synchronous Circuits

Computer Structure - Spring 2008

©Dr. Guy Even

Tel-Aviv Univ.

Preliminary Questions

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

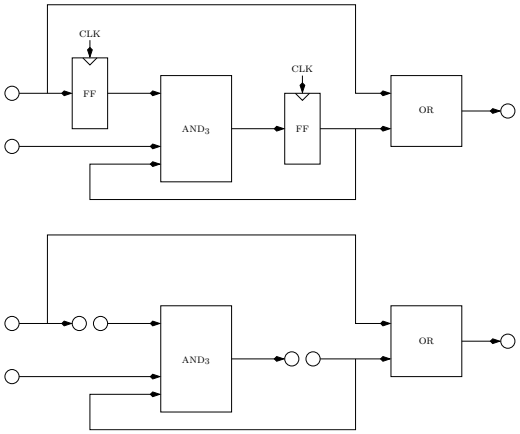
Goals

- define synchronous circuits.
- analyze timing (start with simple case...).
- define: timing constraints.
- find out if timing constraints are feasible.
- define: minimum clock period.
- algorithm: check if timing constraints are feasible.
- algorithm: compute minimum clock period.

Striping flip-flops away

- C - a circuit composed of combinational gates, nets, and flip-flops with a clock net called CLK.
- C' - a circuit obtained from C by:
  1. deleting the CLK net,
  2. deleting the input gate that feeds the CLK net, and
  3. replacing each flip-flip with an output gate (instead of the port D) and an input gate (instead of the port Q).

Striping flip-flops away - example



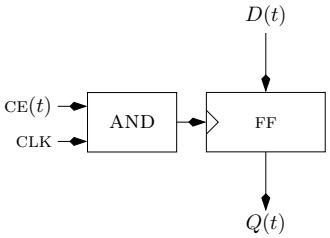
Definition: Synchronous Circuit

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. The circuit C' obtained from C by stripping away flip-flops is combinational.

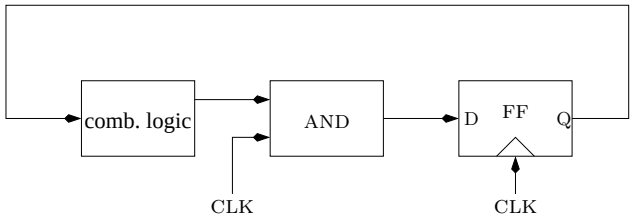
remarks on the definition of synchronous circuits

- CLK connected to all the clock-ports of flip-flops and only to them.
- We already saw that a “bad example” in which CLK feeds a gate:

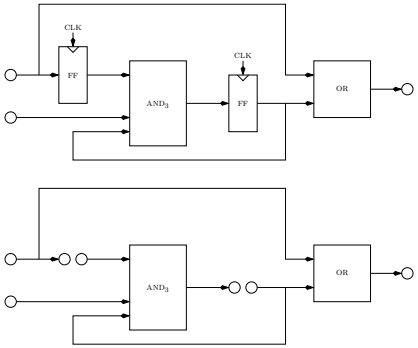


remarks on the definition of synchronous circuits

Question: What is required so that the D-port is stable during the critical segment in this “bad example”:



back to the first example



Question: Is this a synchronous circuit?

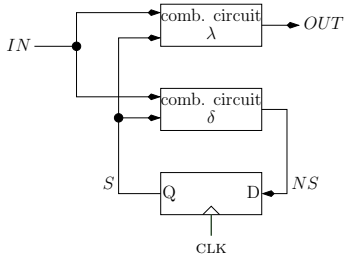
Recognizing a synchronous circuit

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

Recall the definition:
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. The circuit C' obtained from C by stripping away flip-flops is combinational.

Synchronous Circuits: canonic form

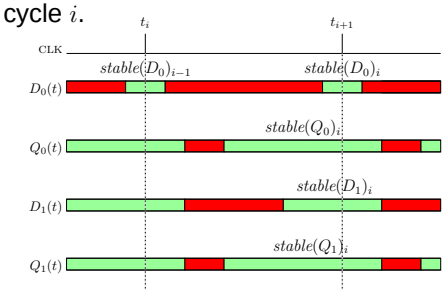


Transform a synchronous to canonic form:

- gather the flip-flops into one group.
- duplicate the combinational circuits to separate between output and next-state.

Stability Interval

- stability interval of signal X - interval during which X is stable.
- $stable(X)_i$ - stability interval of X corresponding to clock cycle i.



Timing analysis: the canonic form

Plan:

- Define timing constraints for IN and OUT.
- Define timing constraints for S and NS.
- Find sufficient conditions so that timing constraints are feasible.
- Define minimum clock period.
- Infer functionality from syntax.

Input/output timing constraints

- The input/output timing constraints formulate the timing interface between the the circuit and the “external world”.
- Input timing constraint - tells us when the input is guaranteed to be stable.
- Output timing constraint - tells us when the circuit’s output is required to be stable.
- Usually the external world is also a synchronous circuit. $\Rightarrow IN$ is an output of another synchronous circuit, and OUT is an input of another synchronous circuit.

Input timing constraint

The timing constraint corresponding to IN is defined by two parameters: *pd*$(IN) >$ *cont*(IN) as follows.

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

Remarks:

- t_i - denotes the starting time of the ith clock period.
- Why do we require that *pd*$(IN) >$ *cont*(IN)?
  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.

Output timing constraint

The timing constraint corresponding to OUT is defined by two parameters: *setup*(OUT) and *hold*(OUT) as follows.

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

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

Remarks

- 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.
- 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 constraint of NS

NS is stable during the critical segments. Namely,

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

Remark: 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}).

Remark: If NS satisfies its timing constraint for i , then S satisfies:

$$[t_{i+1} + t_{\textit{pd}}, t_{i+2} + t_{\textit{cont}}] \subseteq \textit{stable}(S)_{i+1}.$$

Stability Intervals of OUT & NS

- We associate a contamination delay *cont*(x) and a propagation delay *pd*(x) with each combinational circuit x.
- If $[t_i + t_{\textit{pd}}, t_{i+1} + t_{\textit{cont}}] \subseteq \textit{stable}(S)_i$, then the stability intervals of the signals OUT and NS satisfy:

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

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

Sufficient conditions: OUT

Claim: If

$$[t_i + t_{\textit{pd}}, t_{i+1} + t_{\textit{cont}}] \subseteq \textit{stable}(S)_i$$
$$\max\{t_{\textit{pd}}, \textit{pd}(IN)\} + pd(\lambda) + \textit{setup}(OUT) \leq t_{i+1} - t_i$$
$$\min\{t_{\textit{cont}}, \textit{cont}(IN)\} + \textit{cont}(\lambda) \geq \textit{hold}(OUT),$$

then

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

Proof: stability interval of OUT satisfies:

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

□

Sufficient conditions: NS

Claim: If

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

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

Proof: stability interval of NS satisfies:

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

□

Timing constraints for $i \geq 0$

CORO: If 4 conditions hold and

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

then

1. timing constraints of NS and OUT hold wrt every $i \geq 0$,
2. $\forall i \geq 0 \ : \ [t_i + t_{pd}, t_{i+1} + t_{cont}] \subseteq \mathit{stable}(S)_i$.

Proof: Induction on i.

- Basis: part (1) follows from sufficient conditions for OUT and NS.
- Step: NS is stable during $C_{i+1} \Rightarrow$ part (2).
- $\Rightarrow$ part(1).

□

Simplifying the conditions

- Our goal is to simplify the conditions in the 2 Claims.
- Prefer: lower bounds on the clock period.
- $\Rightarrow$ well defined functionality provided that the clock period is large enough.
- We discuss each of the 4 conditions (2 per claim).

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

- condition is a lower bound on $\varphi(\text{CLK})$. Great.

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

- condition may not hold $\Rightarrow$ serious problem that can lead to failure to meet the timing constraint of OUT...
- Hope: under reasonable circumstances, condition does hold. Why?
  - Suppose IN is the output of a combinational circuit, all the inputs of which are outputs of flip-flops.
  - Assume that all the flip-flops are identical.
  - It follows that $cont(IN) \geq t_{cont}$.
  - By definition: $cont(\lambda) \geq 0$.
  - $\Rightarrow \min\{t_{cont}, cont(IN)\} + cont(\lambda) \geq t_{cont}$.
  - Suppose OUT feeds a combinational circuit that feeds a flip-flop.
  - Hence $hold(OUT) \leq t_{hold}$.
  - $t_{hold} < t_{cont} \Rightarrow$ condition holds.

$$\max\{t_{pd}, \mathbf{pd}(IN)\} + \mathbf{pd}(\delta) + t_{su} \leq t_{i+1} - t_i$$

- condition is a lower bound on $\varphi(\text{CLK})$. Great.

$$t_{\mathbf{hold}} \leq \min\{t_{cont}, \mathbf{cont}(IN)\} + \mathbf{cont}(\delta)$$

- As before, if $cont(IN) \geq t_{cont}$, the condition holds!

Conclusion

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

$$[t_0 + t_{pd}, t_1 + t_{cont}] \subseteq stable(S)_0,$$
$$\varphi(\text{CLK}) \geq \max\{t_{pd}, pd(IN)\} + \max\{pd(\lambda) + setup(OUT), pd(\delta) + t_{su}\},$$

then

1. timing constraints of NS and OUT hold wrt every $i \geq 0$,
2. $\forall i \geq 0 : [t_i + t_{pd}, t_{i+1} + t_{cont}] \subseteq stable(S)_i$.

Under reasonable assumptions, all we need is initialization and a sufficiently long clock period.

Minimum clock period

DEF: 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 $\varphi^*(C)$.

- Minimum clock period does not exist if timing constraints are infeasible.
- "timing constraints are satisfied" - for every value of the delays provided that they are in their range. (i.e. actual propagation delay of λ is in $[0, \mathit{pd}(\lambda)]$.)
- if assumptions hold, then in canonic form

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

Discussion

- 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.

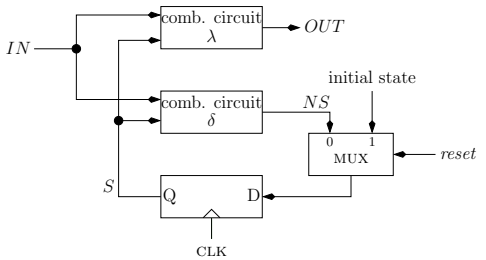
Initialization

We require that

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

- after power-up, flip-flop output may be non-logical (and even meta-stable).
- solution: introduce a reset signal.
- boot-strapping problem: How is a reset signal generated?
- no solution to this problem within the digital abstraction (meta-stability). All we can try to do is reduce the probability of such an event.
- reset controller - a special circuit that generates a reset signal.

Synchronous Circuit: canonic form with reset



Remark: NS may not be logical during reset.
Implementation of MUX must output initial-state if *reset* $= 1$.
Implementation based on drivers has this property, while implementation based on combinational gates may not have this property.

Functionality of Synchronous Circuits: canonic form

- X_i - *dig*(X) during *stable*$(X)_i$.
- Assumptions:

$$\begin{aligned} \textit{cont}(IN) &\geq t_{\textit{cont}} \\ \textit{hold}(OUT) &\leq t_{\textit{hold}} \\ [t_0 + t_{\textit{pd}}, t_1 + t_{\textit{cont}}] &\subseteq \textit{stable}(S)_0, \\ \varphi(\textsc{clk}) &\geq \max\{t_{\textit{pd}}, \textit{pd}(IN)\} \\ &\quad + \max\{\textit{pd}(\lambda) + \textit{setup}(OUT), \textit{pd}(\delta) + t_{\textit{su}}\}. \end{aligned}$$

CORO: Assumptions $\Rightarrow$ $\forall i \geq 0$:

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

Finite State Machines

Corollary states that synchronous circuits implement finite state machines.

DEF: 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.

Definition of FSM: remarks

- Other terms for a finite state machine are a finite automaton with outputs, transducer, and Mealy Machine.
- Moore Machine - an FSM in which the output function $\lambda : Q \to \Delta$.

What does an FSM do?

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

$$q_{i+1} \stackrel{\triangle}{=} \delta(q_i, x_i)$$

- The output y_i is defined as follows:

$$y_i \stackrel{\triangle}{=} \lambda(q_i, x_i).$$

State Diagrams

FSMs are often depicted using state diagrams.

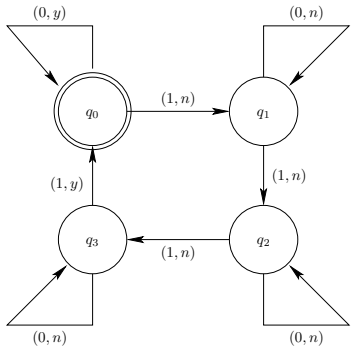
DEF: The state diagram corresponding to an FSM $\mathcal{A}$ is a directed graph $G = (V, E)$ with edge labels $(x, y) \in \Sigma \times \Delta$. The vertex set V equals the state set S. The edge set E is defined by

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

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

State Diagram: example

A state diagram of an FSM that outputs y if the weight of the input so far is divisible by 4, and n otherwise.



Timing analysis: the general case

- Deal with a synchronous circuit that is not in canonic form.
- Algorithm that computes the minimum clock period $\varphi^*(C)$. (if timing constraints are feasible.)
- Algorithm that decides whether the timing constraints are feasible (i.e. conditions used by this algorithm are less restrictive than the conditions used in previous claims).

Recap

- 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:
  - timing constraints are infeasible $\Rightarrow$ 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.
  - timing constraints are feasible $\Rightarrow$ functionality is well defined provided that the clock period satisfies $\varphi(\mathsf{CLK}) \geq \varphi^*(\mathsf{CLK})$.

Functionality

- Assume that the timing constraints are feasible.
- Introduce 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.
- Advantage: decouple timing issues from functionality and enables simple logical simulations.

The zero delay model

- In the zero delay model we assume that all the parameters of all the components are zero or infinitesimal (i.e. $\forall \varepsilon > 0$: $t_{su} = -\varepsilon$, $t_{hold} = t_{cont} = t_{pd} = \varepsilon$, $pd(IN) = cont(IN) = hold(OUT) = \varepsilon$, $setup(OUT) = -\varepsilon$ and $d(G) = t_{cont}(G) = 0$, for every combinational gate G). Under this unrealistic assumption, the timing constraints are feasible.
- Must pay attention to endpoints of intervals of stability: Output of flip-flip satisfies:

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

Hence,

$$(t_i, t_{i+1}] \subseteq stable(Q)_i.$$

The zero delay model - cont

- Similarly,

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

Hence,

$$(t_i, t_{i+1}] \subseteq stable(IN)_i.$$

- Following Corollary (synchronous circuit implements an FSM), we conclude that, for every signal X, X_i is well defined.

Simulation of a synchronous circuit

Simulation during cycles $i = 0, \ldots, n-1$ in the zero propagation model proceeds as follows:
assume: flip-flops are initialized ($\vec{S}_0$ - initial values of FFs).

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

Summary

- define synchronous circuits.
- canonic form of synchronous circuits:
  - definition of timing constraints.
  - formulation of sufficient conditions for satisfying the timing constraints.
  - simplify sufficient conditions by relying on the assumption that the input originates from a flip-flop and the output is eventually fed to a flip-flop.
  - define the minimum clock period.
  - initialization.
  - synchronous circuits implement FSMs.

Summary -cont.

- functionality:
  - zero delay model.
  - simulation.