Hardware Modeling [VU] (191.011)
– WS25 –
Finite-State Machine Modeling

Florian Huemer & Sebastian Wiedemann & Dylan Baumann

WS 2025/26

Modified: 2025-12-16, 16:04 (f8a58e9)

Motivation

- Why not simply draw circuits?

Motivation

- Why not simply draw circuits?
  - Error-prone and scaling poorly for complex FSMs

Motivation

- Why not simply draw circuits?
  - Error-prone and scaling poorly for complex FSMs
  - Hard to understand and maintain

Motivation

- Why not simply draw circuits?
  - Error-prone and scaling poorly for complex FSMs
  - Hard to understand and maintain

$\Rightarrow$ Increase level of abstraction of description

Motivation

- Why not simply draw circuits?
  - Error-prone and scaling poorly for complex FSMs
  - Hard to understand and maintain

$\Rightarrow$ Increase level of abstraction of description
  - Draw FSMs as graphs (nodes for states, edges for transitions)

Motivation

- Why not simply draw circuits?
  - Error-prone and scaling poorly for complex FSMs
  - Hard to understand and maintain

$\Rightarrow$ Increase level of abstraction of description
  - Draw FSMs as graphs (nodes for states, edges for transitions)
  - Implementation derived by tools

Common FSM Model

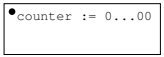
Behavior Description

The `simple_timer` module keeps an internal `N`-bit counter that is (synchronously) incremented as long as `en` is asserted. When the counter reaches its maximum value ($2^N - 1$) it overflows back to zero. If the counter hits its maximum and `en` is asserted the `tick` output is asserted.

Common FSM Model

Behavior Description

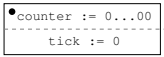
The `simple_timer` module keeps an internal `N`-bit counter that is (synchronously) incremented as long as `en` is asserted. When the counter reaches its maximum value ($2^N - 1$) it overflows back to zero. If the counter hits its maximum and `en` is asserted the `tick` output is asserted.



Common FSM Model

Behavior Description

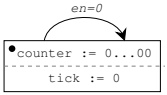
The `simple_timer` module keeps an internal `N`-bit counter that is (synchronously) incremented as long as `en` is asserted. When the counter reaches its maximum value ($2^N - 1$) it overflows back to zero. If the counter hits its maximum and `en` is asserted the `tick` output is asserted.



Common FSM Model

Behavior Description

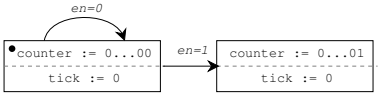
The `simple_timer` module keeps an internal `N`-bit counter that is (synchronously) incremented as long as `en` is asserted. When the counter reaches its maximum value ($2^N - 1$) it overflows back to zero. If the counter hits its maximum and `en` is asserted the `tick` output is asserted.



Common FSM Model

Behavior Description

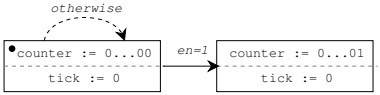
The `simple_timer` module keeps an internal `N`-bit counter that is (synchronously) incremented as long as `en` is asserted. When the counter reaches its maximum value ($2^N - 1$) it overflows back to zero. If the counter hits its maximum and `en` is asserted the `tick` output is asserted.



Common FSM Model

Behavior Description

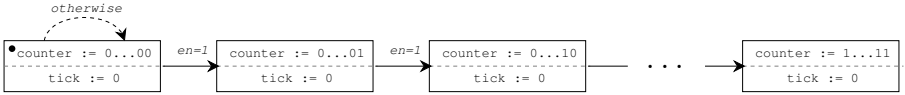
The `simple_timer` module keeps an internal `N`-bit counter that is (synchronously) incremented as long as `en` is asserted. When the counter reaches its maximum value ($2^N - 1$) it overflows back to zero. If the counter hits its maximum and `en` is asserted the `tick` output is asserted.



Common FSM Model

Behavior Description

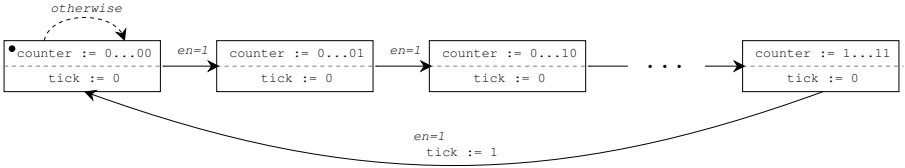
The `simple_timer` module keeps an internal `N`-bit counter that is (synchronously) incremented as long as `en` is asserted. When the counter reaches its maximum value ($2^N - 1$) it overflows back to zero. If the counter hits its maximum and `en` is asserted the `tick` output is asserted.



Common FSM Model

Behavior Description

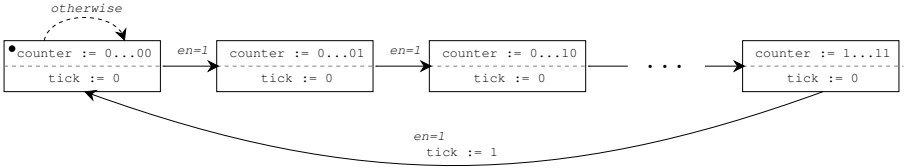
The `simple_timer` module keeps an internal `N`-bit counter that is (synchronously) incremented as long as `en` is asserted. When the counter reaches its maximum value ($2^N - 1$) it overflows back to zero. If the counter hits its maximum and `en` is asserted the `tick` output is asserted.



Common FSM Model

Behavior Description

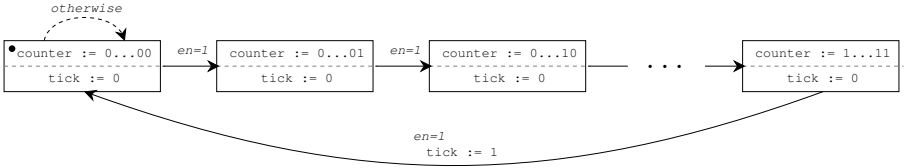
The `simple_timer` module keeps an internal `N`-bit counter that is (synchronously) incremented as long as `en` is asserted. When the counter reaches its maximum value ($2^N - 1$) it overflows back to zero. If the counter hits its maximum and `en` is asserted the `tick` output is asserted.



Common FSM Model

Behavior Description

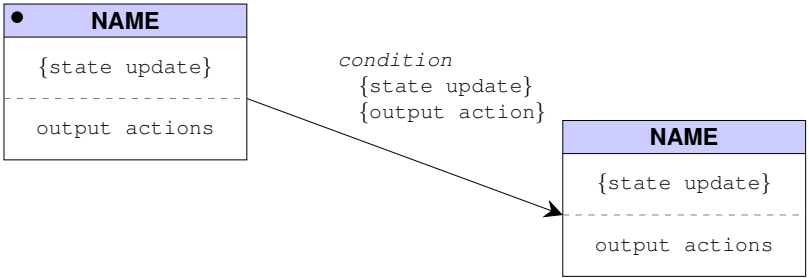
The `simple_timer` module keeps an internal `N`-bit counter that is (synchronously) incremented as long as `en` is asserted. When the counter reaches its maximum value ($2^N - 1$) it overflows back to zero. If the counter hits its maximum and `en` is asserted the `tick` output is asserted.



2^N states$\Rightarrow$ Scaling poorly! (“state explosion”)

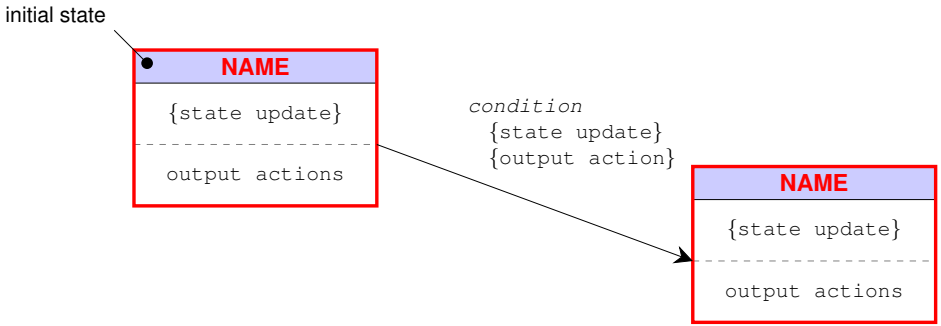
HWMod FSM Model

- Need more abstract FSM model



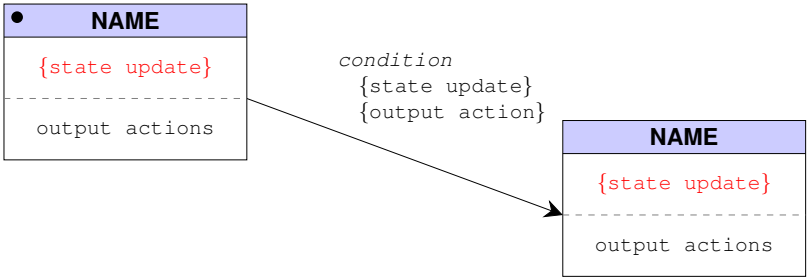
HWMod FSM Model

- Need more abstract FSM model
  - “Abstract states” gather multiple states with similar behavior



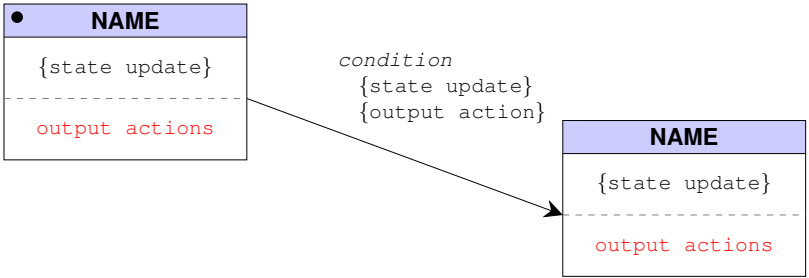
HWMod FSM Model

- Need more abstract FSM model
  - Optional assignment to state register; per default value unchanged



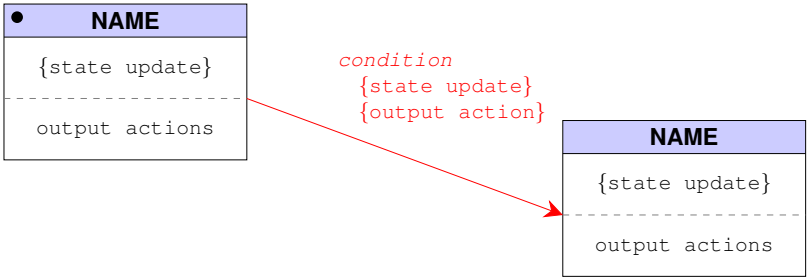
HWMod FSM Model

- Need more abstract FSM model
  - Mapping of each state to an output



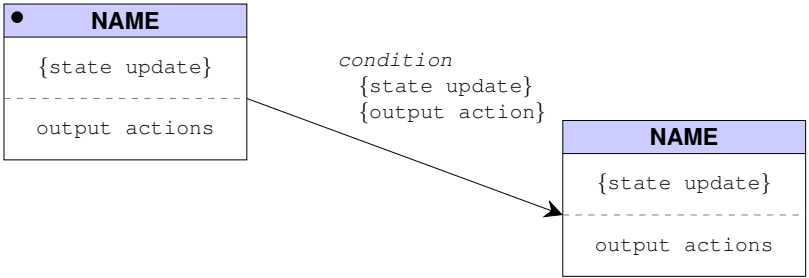
HWMod FSM Model

- Need more abstract FSM model
  - Conditional state transitions; implicit `otherwise` self-loop if not exhaustive
  - Optional assignments to state register and outputs; overriding node's actions



HWMod FSM Model

- Need more abstract FSM model
  - Conditional state transitions; implicit `otherwise` self-loop if not exhaustive
  - Optional assignments to state register and outputs; overriding node's actions
- Mealy when transition output action depends on input



Example I: simple_timer

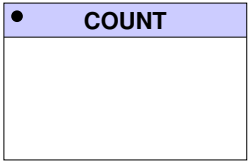
Behavior Description

The simple_timer module keeps an internal N-bit counter that is (synchronously) incremented as long as en is asserted. When the counter reaches its maximum value ($2^N - 1$) it overflows back to zero. If counter hits its maximum and en is asserted the tick output is asserted.

Example I: `simple_timer`

Behavior Description

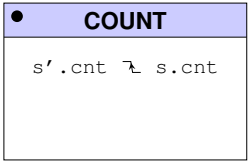
The `simple_timer` module keeps an internal `N`-bit counter that is (synchronously) incremented as long as `en` is asserted. When the counter reaches its maximum value ($2^N - 1$) it overflows back to zero. If counter hits its maximum and `en` is asserted the `tick` output is asserted.



Example I: `simple_timer`

Behavior Description

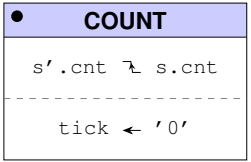
The `simple_timer` module keeps an internal `N`-bit counter that is (synchronously) incremented as long as `en` is asserted. When the counter reaches its maximum value ($2^N - 1$) it overflows back to zero. If counter hits its maximum and `en` is asserted the `tick` output is asserted.



Example I: simple_timer

Behavior Description

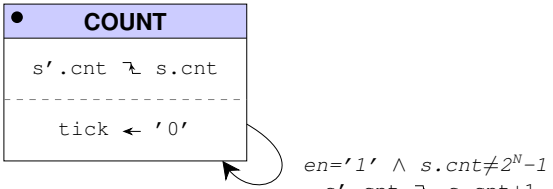
The simple_timer module keeps an internal N-bit counter that is (synchronously) incremented as long as en is asserted. When the counter reaches its maximum value ($2^N - 1$) it overflows back to zero. If counter hits its maximum and en is asserted the tick output is asserted.



Example I: simple_timer

Behavior Description

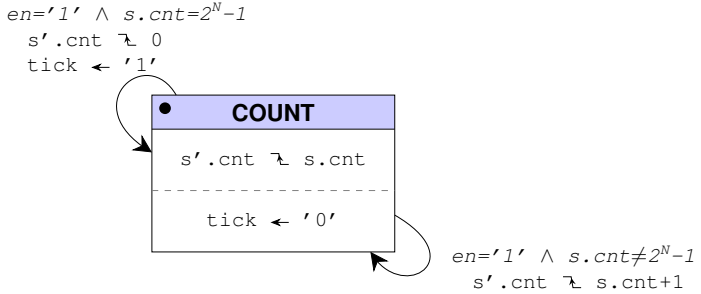
The simple_timer module keeps an internal N-bit counter that is (synchronously) incremented as long as en is asserted. When the counter reaches its maximum value ($2^N - 1$) it overflows back to zero. If counter hits its maximum and en is asserted the tick output is asserted.



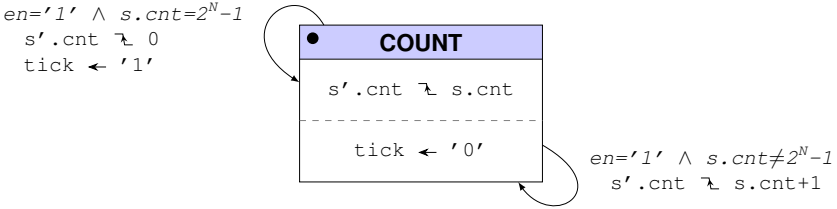
Example I: simple_timer

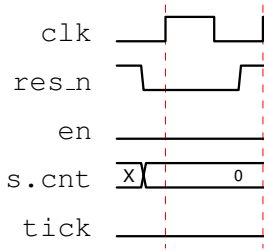
Behavior Description

The simple_timer module keeps an internal N-bit counter that is (synchronously) incremented as long as en is asserted. When the counter reaches its maximum value ($2^N - 1$) it overflows back to zero. If counter hits its maximum and en is asserted the tick output is asserted.

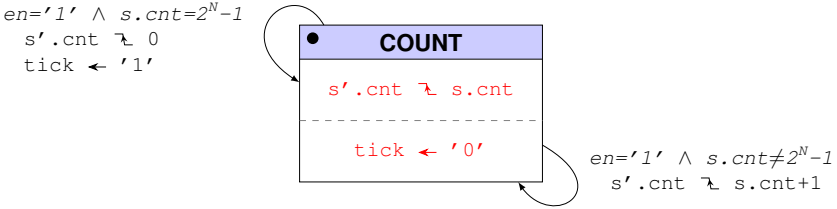


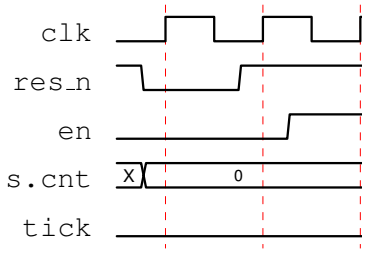
Synchronous FSM Behavior



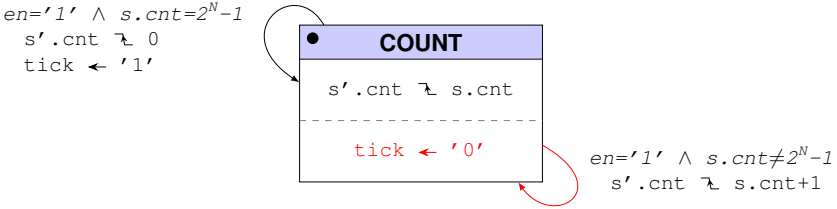


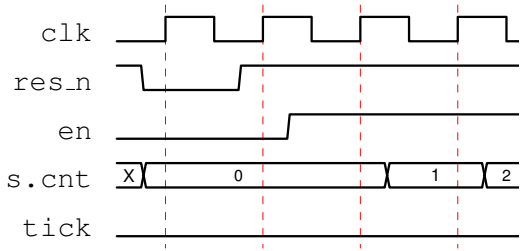
Synchronous FSM Behavior



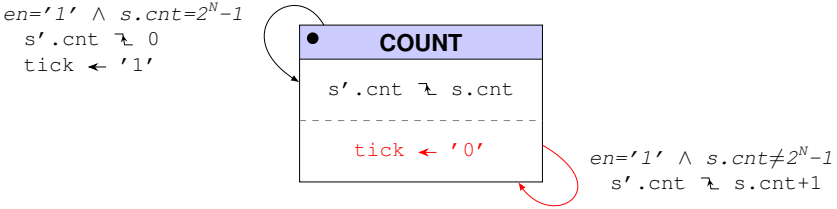


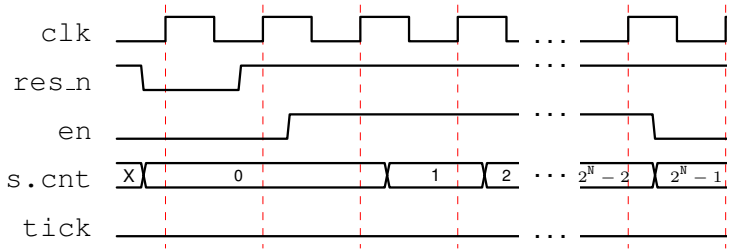
Synchronous FSM Behavior



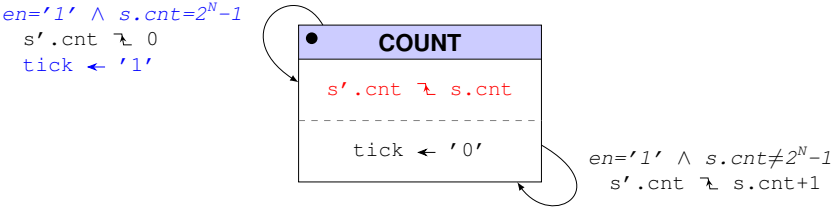


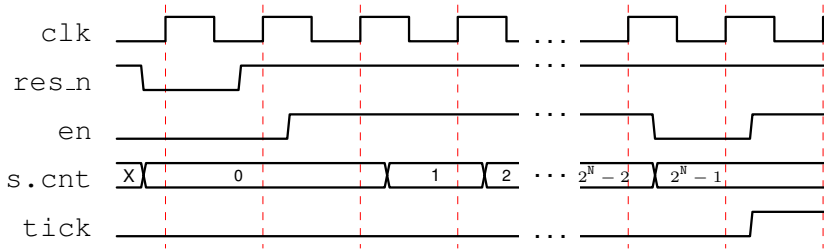
Synchronous FSM Behavior



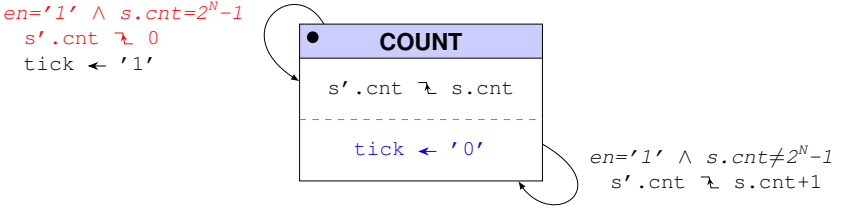


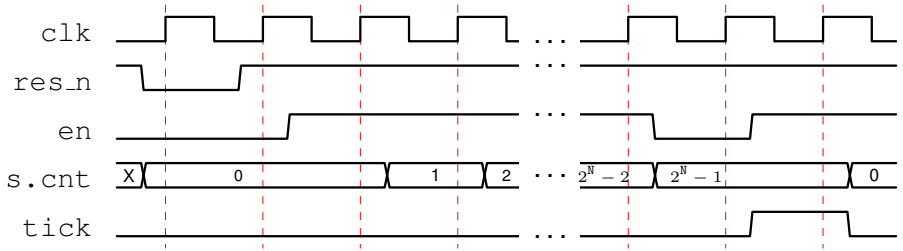
Synchronous FSM Behavior





Synchronous FSM Behavior





Example II: advanced_timer

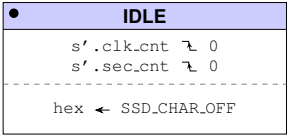
Behavior Description

After a press of button `btn_n` the synchronous `advanced_timer` module starts counting from zero to 15 seconds (inclusive). It shall display the current second as hex value on the seven-segment display `hex`. Before the button press and after it finished counting, the seven-segment display shall be turned off.

Example II: advanced_timer

Behavior Description

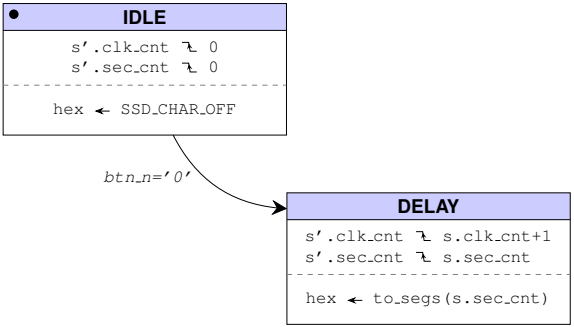
After a press of button btn_n the synchronous advanced_timer module starts counting from zero to 15 seconds (inclusive). It shall display the current second as hex value on the seven-segment display hex. Before the button press and after it finished counting, the seven-segment display shall be turned off.



Example II: advanced_timer

Behavior Description

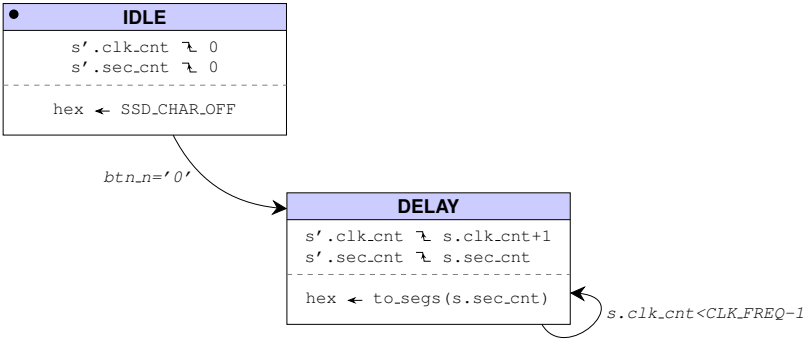
After a press of button btn_n the synchronous advanced_timer module starts counting from zero to 15 seconds (inclusive). It shall display the current second as hex value on the seven-segment display hex. Before the button press and after it finished counting, the seven-segment display shall be turned off.



Example II: advanced_timer

Behavior Description

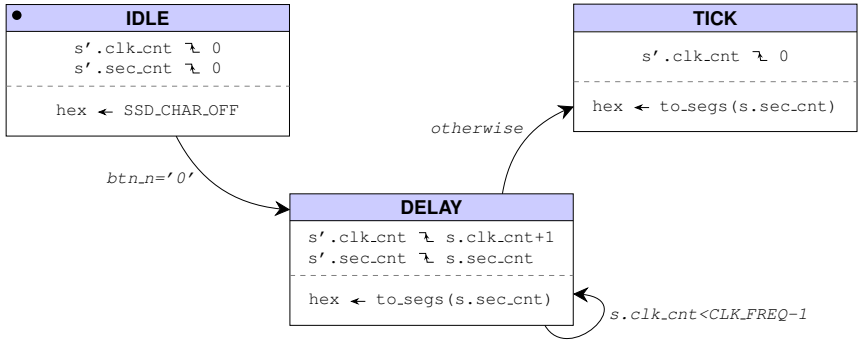
After a press of button `btn_n` the synchronous `advanced_timer` module starts counting from zero to 15 seconds (inclusive). It shall display the current second as hex value on the seven-segment display `hex`. Before the button press and after it finished counting, the seven-segment display shall be turned off.



Example II: advanced_timer

Behavior Description

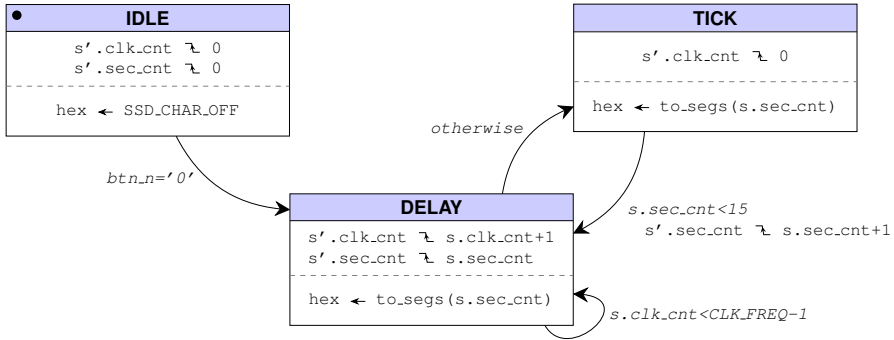
After a press of button btn_n the synchronous advanced_timer module starts counting from zero to 15 seconds (inclusive). It shall display the current second as hex value on the seven-segment display hex. Before the button press and after it finished counting, the seven-segment display shall be turned off.



Example II: advanced_timer

Behavior Description

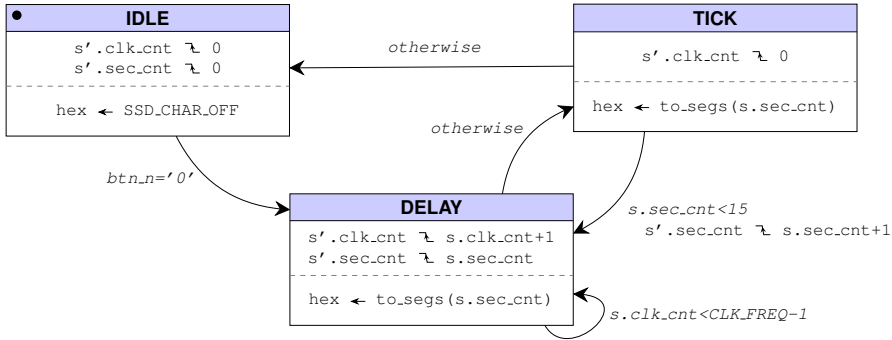
After a press of button btn_n the synchronous advanced_timer module starts counting from zero to 15 seconds (inclusive). It shall display the current second as hex value on the seven-segment display hex. Before the button press and after it finished counting, the seven-segment display shall be turned off.



Example II: advanced_timer

Behavior Description

After a press of button btn_n the synchronous advanced_timer module starts counting from zero to 15 seconds (inclusive). It shall display the current second as hex value on the seven-segment display hex. Before the button press and after it finished counting, the seven-segment display shall be turned off.



Lecture Complete!