

AN-1126 ASM to Synchronous Conversion

State machines are frequently used to make complex devices. They help in determining signal sequencing and highlight how parameters get changed on state transitions etc.

Two of the possible state machines types are: synchronous and asynchronous. Asynchronous state machines (aka ASMs, as found in GreenPAK™) perform transitions from one state to the other as soon as the required condition (High level signal) appears on the desired input. Synchronous state machines (SSM) do transitions on the edge of an incoming clock if the transition condition is met. A Synchronous state machine implementation is suitable for certain designs.

This application note describes how to convert the GreenPAK asynchronous state machine into a synchronous state machine.

Synchronous State Machine Design

It is difficult to define a generic approach to convert an ASM into a SSM, since every state machine is unique.

In the case of a simple state diagram (when there are different conditions for different transitions and each state is used only once) it is enough to just add DFFs (see Figure 1).

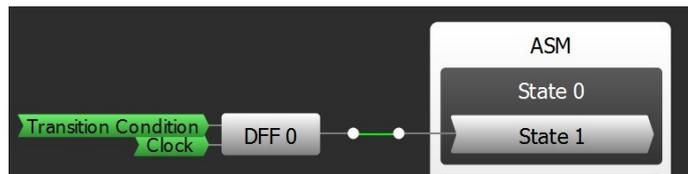


Figure 1. Adding synchronization to ASM

In the case of a complex state diagram, it is not enough to just add DFFs, because some states could get used many times and they should be reset periodically as needed.

Since the ASM in the GreenPAK makes a transition on the High level (and not on the edge), it seems like the simplest solution would be to use an edge detector for generating a clock signal with a logical AND gate (see Figure 2) for each transition condition.

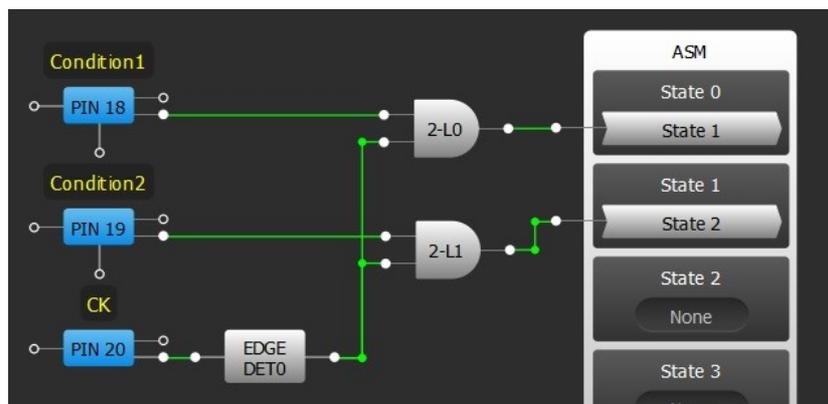


Figure 2. "Edge detector" solution

However, because it is possible for the edge detector pulse width to be greater than the ASM transition time, this solution doesn't guarantee that the ASM will make only one transition on a clock signal edge. The reliability of such circuits is based on the propagation delay difference, (between the ASM transition and the block which causes the transition: edge detector) which varies from chip to chip, depending on the VDD and temperature. All this makes such a solution unreliable and not recommended.

Conversely, we can try to monitor the ASM outputs OUT0..OUT7 to catch a transition event and to enable / disable further transitions. Unfortunately, this is also not the best solution since some blocks might be too slow to prevent the next transition. The propagation delay difference issue takes place in this case as well.

This means that a circuit should clearly know in which state the ASM currently is and to which state the transitions will take it to. Only in such a case will SSM be stable and reliable.

To do this, one more state machine should be created using the internal blocks.

On one hand, it seems inappropriate to create an additional state machine for the ASM to become a SSM; On the other hand though, the additional state machine will be simpler, and will have 64 bits of ASM RAM available.

In this case, for each ASM transition specific block, the circuit elements shown in Figure 3 should be added.

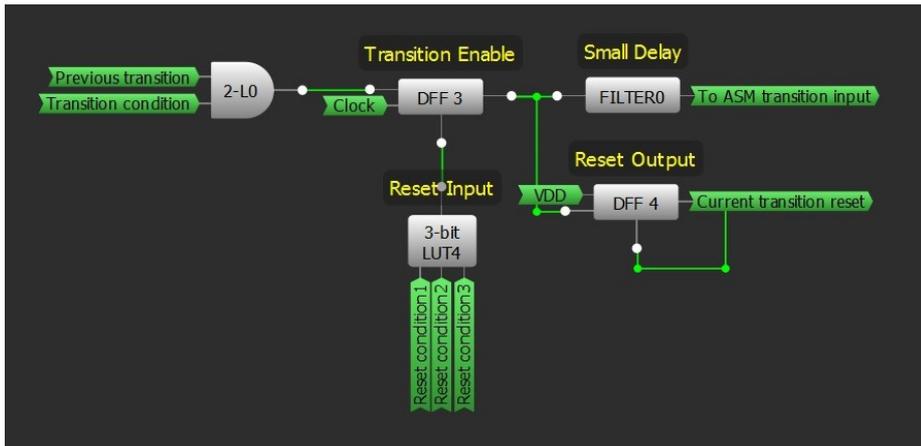


Figure 3. General synchronization block structure

RAM								
State name	Connection Matrix Output RAM							
	OUT7	OUT6	OUT5	OUT4	OUT3	OUT2	OUT1	OUT0
State 0	0	0	0	0	0	0	0	1
State 1	0	0	0	0	0	0	1	0
State 2	0	0	0	0	0	1	0	0
State 3	0	0	0	0	1	0	0	0
State 4	0	0	0	1	0	0	0	0
State 5	0	0	1	0	0	0	0	0
State 6	0	1	0	0	0	0	0	0
State 7	1	0	0	0	0	0	0	0

Figure 4. ASM RAM

The DFF3 provides the transition to the desired state from the previous one on the rising edge of clock signal if the defined (by 2-L0 connections) condition is asserted.

The DFF3 can generate (using DFF4) reset for other "transition DFFs" after ASM transitions (proper sequence is provided by adding small delay – Filter0), if necessary. Also it can reset itself and other "transition DFFs" (connections to 3-L4).

This circuit is universal and not all elements are required for each transition, so it can be simplified; if desired.

As an example, a state diagram (see Figure 6) was chosen. There are only two points where some of the DFFs will be reset: State0 -> State1 (DFF3, DFF4, DFF5) and State2 -> State3 (DFF4, DFF6, DFF7) transitions. Such a combination ensures the reset of all the DFFs when required except for DFF1 (as this transition can be used only once).

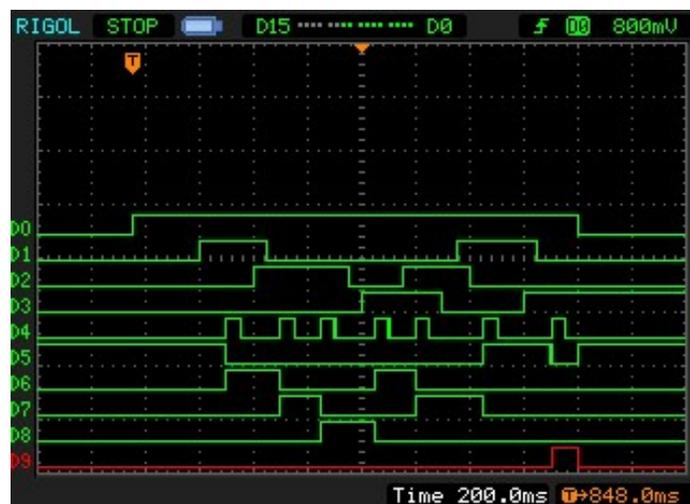


Figure 5. Synchronous State Machine transitions

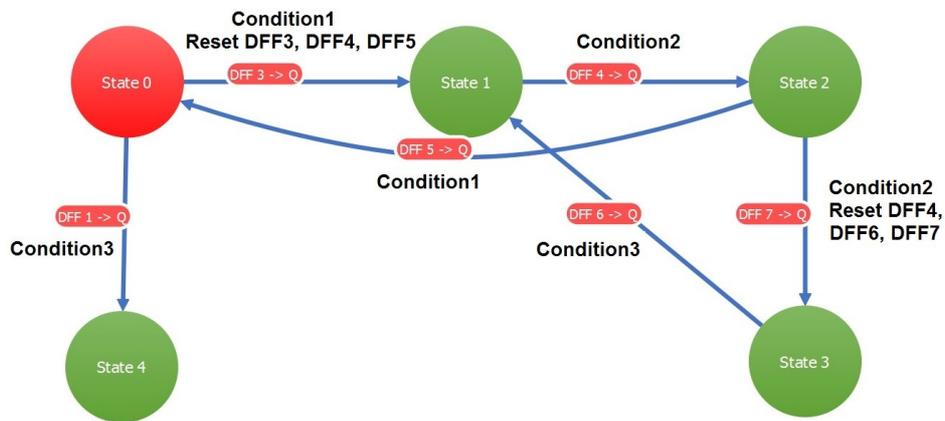


Figure 6. ASM state diagram

The GreenPAK Design circuit is shown in Figure 7, and the corresponding waveforms in Figure 5 where:

- D0 – PIN#10 (ASM Reset)
- D1 – PIN#18 (Condition1)
- D2 – PIN#19 (Condition2)
- D3 – PIN#17 (Condition3)
- D4 – PIN#20 (CK)
- D5 – PIN#3 (OUT0)
- D6 – PIN#4 (OUT1)
- D7 – PIN#5 (OUT2)
- D8 – PIN#6 (OUT3)
- D9 – PIN#7 (OUT4)

OUT0..OUT7 outputs reflect current state (see ASM RAM – Figure 4).

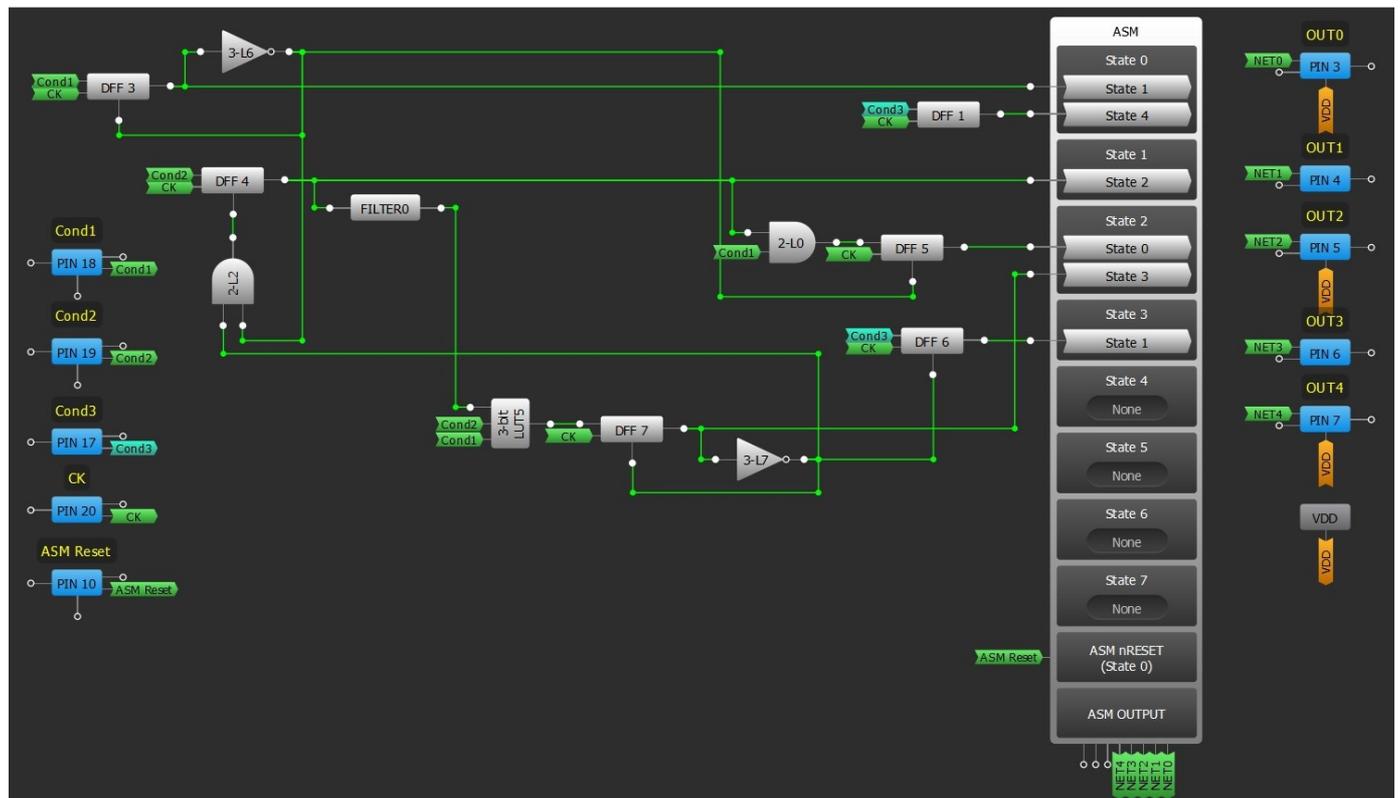


Figure 7. GPAK Design circuit

Conclusion

It was shown that by using additional internal blocks within the GreenPAK CMIC, its asynchronous state machine can be converted into a reliable synchronous state machine design.

About the Author

Name: Yurii Shchebel

Background: Yurii Shchebel graduated from Ivan Franko Lviv National University in 2015, studying at the Department Electronics. Presently he is working with Configurable Mixed Signal ICs (CMICs), also designing audiophile amplifiers based on ICs, transistors and vacuum tubes.

Contact: appnotes@silego.com

Files

- [AN-1126 ASM to Synchronous Conversion.gp5](#)- (50 KB)
- [AN-1126 ASM to Synchronous Conversion.pdf](#)- (395 KB)
- [AN-1126.zip](#)- (319 KB)

[See full list of Application Notes](#)