Application Note

Using State Machines in Programmable Logic

Owen Westfall

ABSTRACT

This application note provides a brief description of state machines before diving into applications that can use state machines. The focus is on TI's new programmable logic family TPLD, showing how to setup and create a complete design within InterConnect Studio, the software used to configure a TPLD, that contains a state machine as the center of the design.

Table of Contents

Trademarks

1 What is a State Machine

To understand a state machine we must first define what a state is. A state is a mode of operation that comes with predefined behavior and a trigger condition either transitioning from the state or into the state. That behavior can be a set of outputs expected for example one state can have a logic output of 0010, while another outputs 1011. A state machine (SM) is a system to control a device or program to step through these states. When looking at SM diagrams the circle denotes the state and the behavior of that state is defined either within or below that circle.

The simplest state machine is shown in Figure 1-1.The device sits idle in a power off mode. Once power is applied the device powers on, and outputs 0. Thinking about programs and devices as state machines can often simplify code, and enable easier debugging as the user targets specific states for specific predefined behaviors.

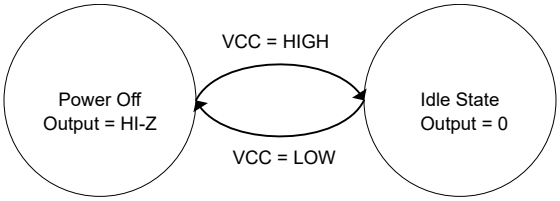

Figure 1-1. Basic State Machine

While many state machines are implicit some are explicit. An implicit state machine is one not clearly defined in the design process such as the SM from Figure 1-1. An explicit state machine is one where A great example of explicit state machines can be found in the TPLD family of devices, specifically the TPLD1202. The state machine in the TPLD1202 has eight states, can be used to control eight different outputs per state, and can transition from any single state to another. This state machine can operate as either an asynchronous state machine or a synchronous state machine, and allows for a global reset to bring the state machine to a predefined state at any point during operation.

2 Difference Between an Asynchronous and Synchronous State Machine

State machines in relation to hardware can often be divided into one of two main categories, an asynchronous state machine (ASM) or a synchronous state machine (SSM).

An ASM is a style of state machine that can transition between the states without a clock being present. Benefits of this style of state machine are the device can transition at any given time, and no clock is needed to be running and consuming power when the state machine is running. A downside of this state machine is that you cannot have two consecutive states tied to the same transition flag. An example of an ASM is a simple AND gate as seen in Figure 2-1. In this ASM A, B, and VCC are inputs into the device, and there is one output. The device powers on and remains in an idle state until the device sees one of the primary inputs go high. At this point the SM transitions into a half state where the output does not change. Once the other input goes high the SM transitions into the *AND* state. At which point the logic gate outputs a logic high.

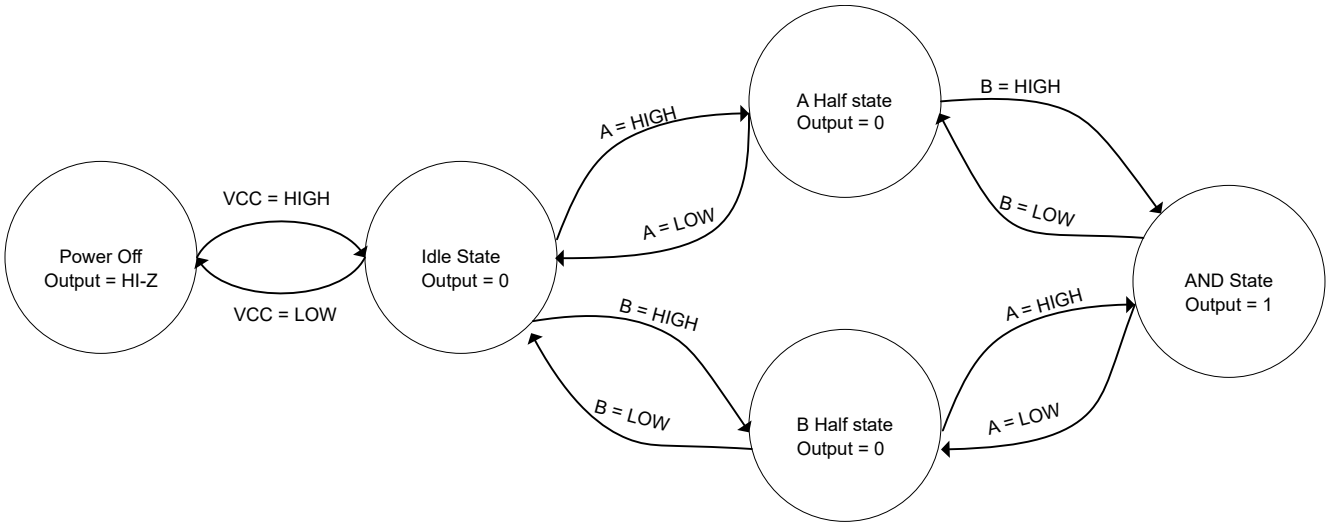

Figure 2-1. And Gate SM

A SSM is a state machine that has a clock attached to the transitions. This means that the state machine only checks the transition flags on the rising edge of the clock cycle. The benefit of this style is allowing consecutive state transitions to be run off the same transition flag. The clock allows for an inherent delay between switching states, and can assist in synchronizing multiple different SMs. A downside to this style of SM is a clock must always be running in the system consuming power.

An example of this style of state machine is a traffic light as shown in Figure 2-2. Using newer style traffic signals that have an induction loop to detect if a car is currently waiting at the light. Once a car is detected the trigger to transition states is set, but a traffic light can not transition without the cross traffic transitioning at the same time so the state machine inside them waits for the clock to signify that a change is allowed to happen. The SM enters the Green light stage and waits until a car is detected at the cross road. Once that happens the SM transitions to yellow then red at the next clock pulse. The SM starts all over again once at this point.

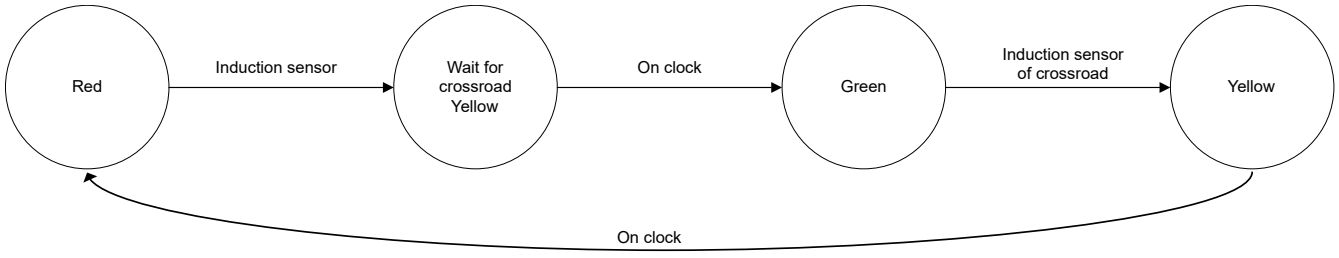

Figure 2-2. Traffic Signal State Machine

3 How to Configure a State Machine

InterConnect Studio (ICS) is the software used to configure a TPLD. Once ICS is launched and a device is selected adding a state machine to the design enables configuration of the dedicated state machine block. Figure 3-1 shows the default configuration when the state machine is added. To add additional states click the plus icon highlighted in Figure 3-1. The only input to the state machine by default is the NRST input. This input is active low and causes the state machine to revert back to the initial state. By default this is st0 and this input is asynchronous even when the synchronous mode of the state machine is selected.

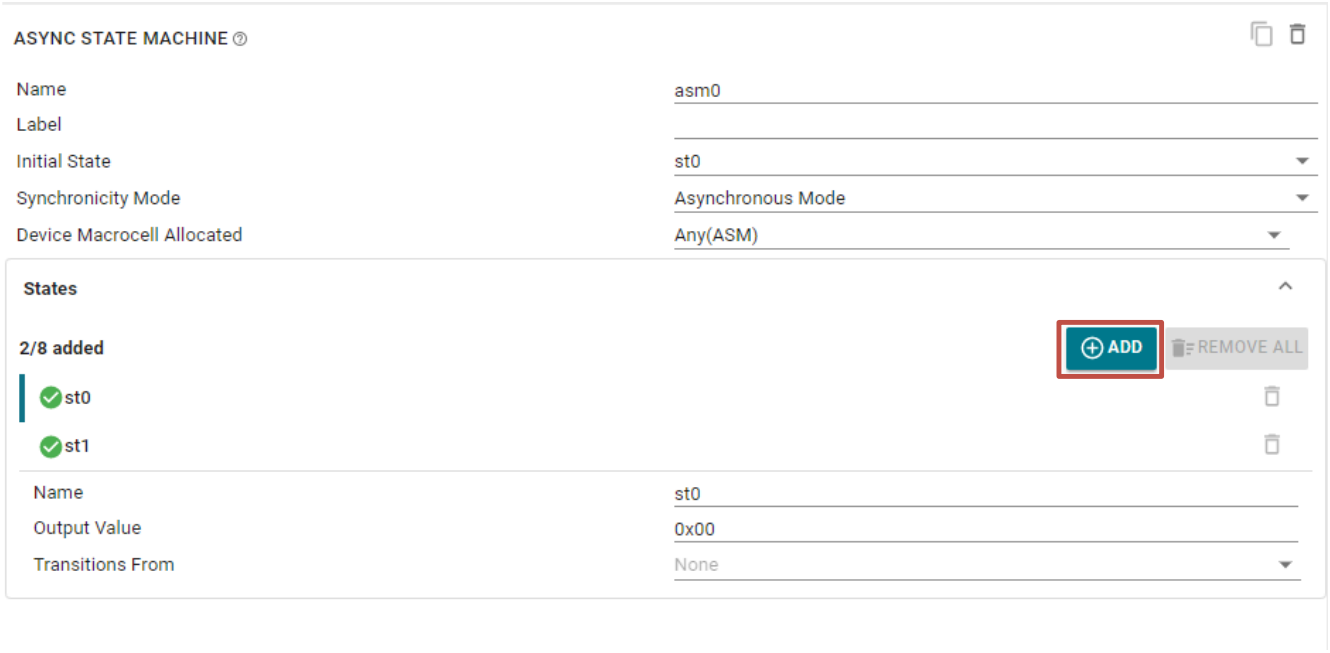

Figure 3-1. Default State Machine with Add State Button Highlighted

The state options in this state machine are name, output value, and transitions from shown in Figure 3-2. The Name configurable simply adjusts the reference of the state in the SM view and the configuration space, but has no affect on the actual design. Output value is a hex representation of the binary values present at OUT7 - OUT0 while the state machine is in that state. Each output can be connected to an internal port, or routed to output pins for external use. The Transitions From drop-down menu enables the transitions from one state to another. In this drop-down menu selecting any option adds a transition from the option into the current selected state For example, if the current state selected is st0, selecting st1 in that drop-down menu creates a transition event from st1 to st0 as shown in Figure 3-3.

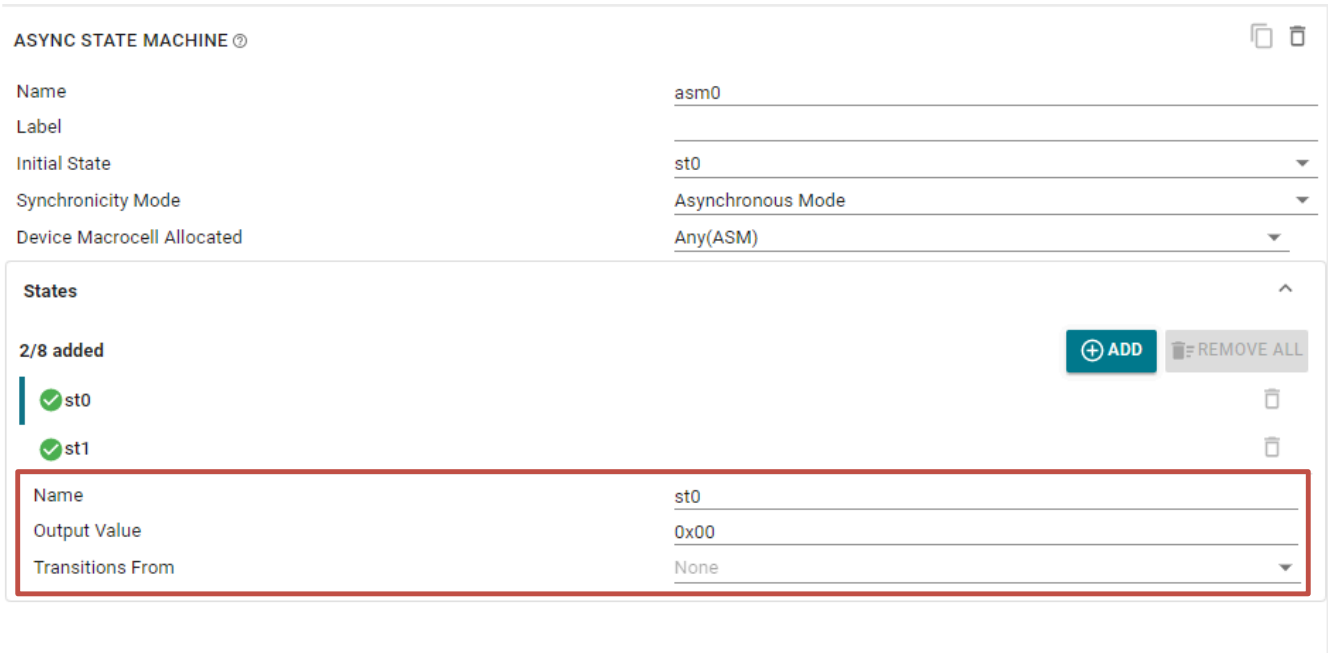

Figure 3-2. State Settings

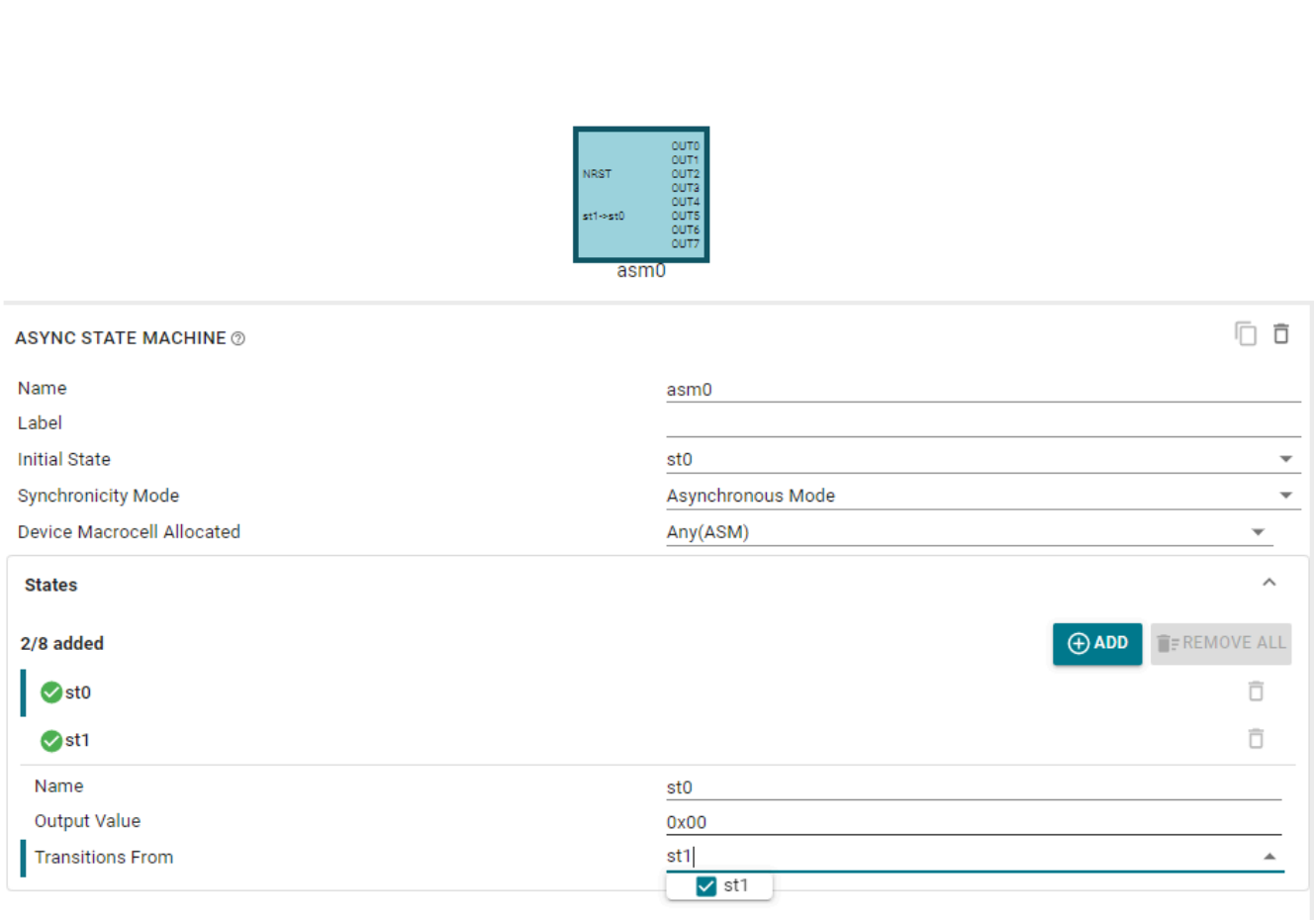

Figure 3-3. Example Transition From st1 to st0

Creating the transition events is not enough to cause the state machine to transition the state also needs a trigger to spark that transition. Figure 3-4 shows a very basic implementation of the SM at work. NRST is tied to POR so the state machine never resets. Once pin0 goes high this state machine transitions out of it's default state st1 into st0.

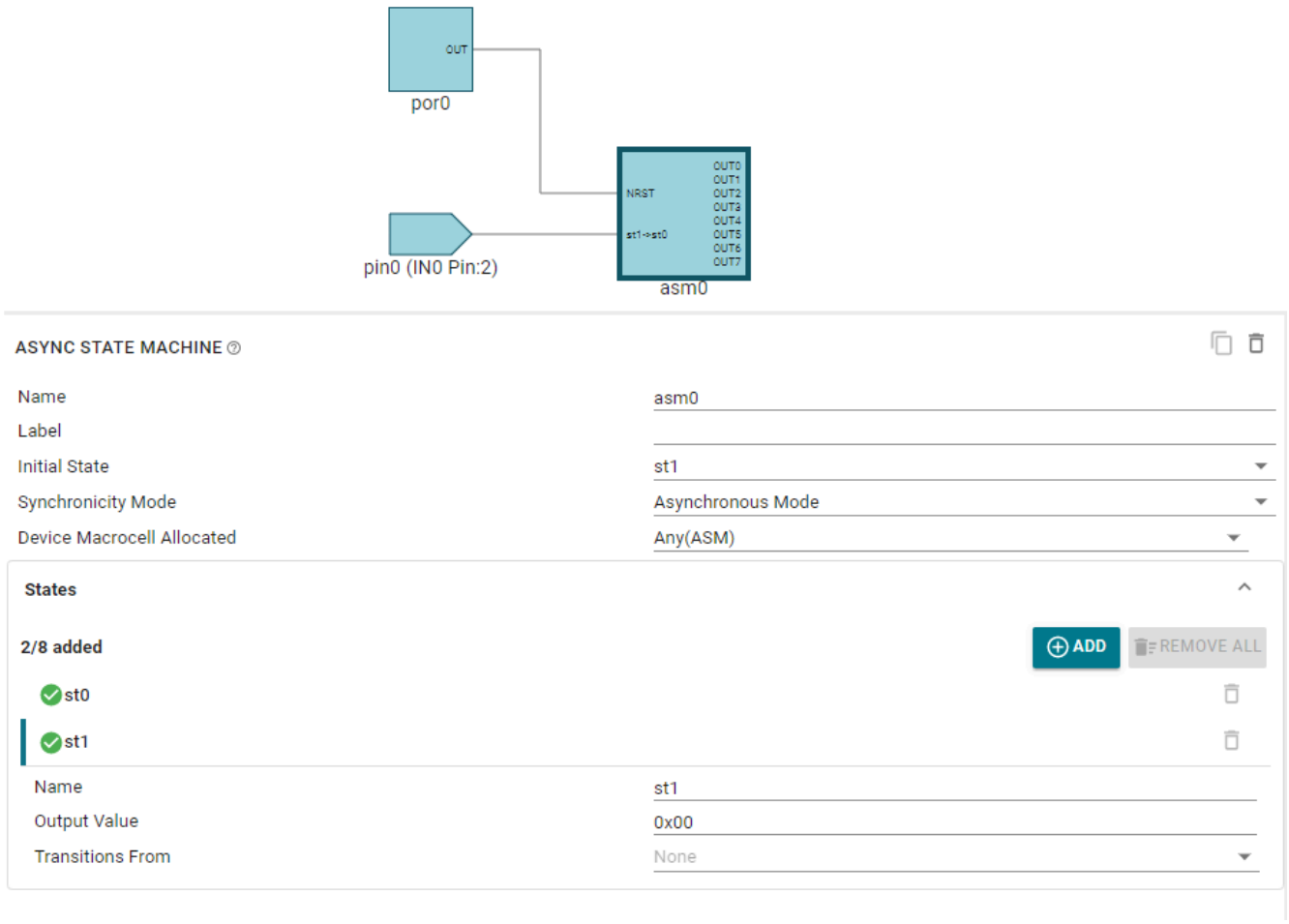

Figure 3-4. Basic State Machine

4 Triggering a State Machine with User Inputs

A state machine can be used to detect inputs from a single button to perform different actions. A design like this can be used as the primary controller in many simple systems, or alternately can be used to offload some complexity for the firmware development to an external TPLD device when using a microcontroller.

In Figure 4-1 you can see a state machine that is designed to sort the inputs from a single button into 4 different common input formats. Those options are a single press, 1s hold, 2s hold, and double press. This state machine can be utilized in the TPLD as shown in Figure 4-2. The outputs of the state machine are driving the timers for the double press timing, and some LED's on our EVM board to show the device transitioning into different states.

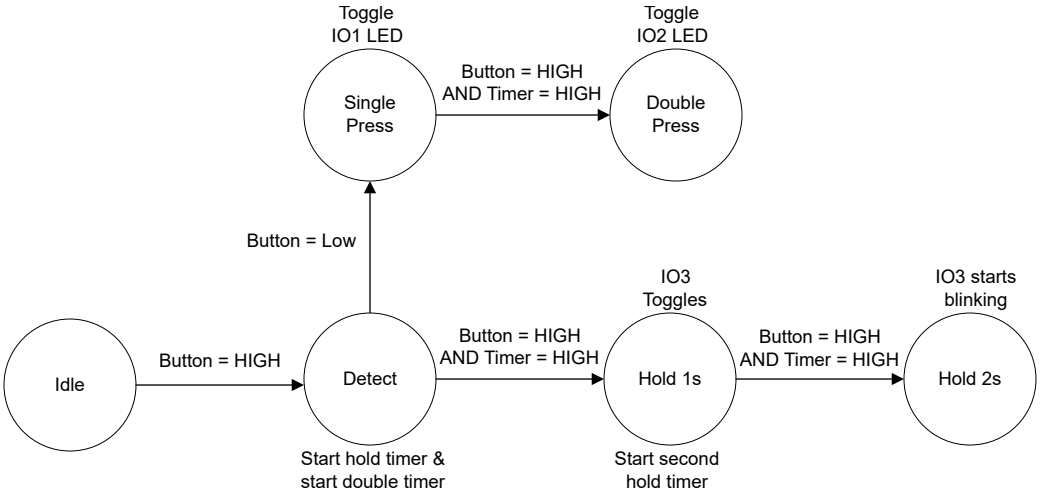

Figure 4-1. Single Button State Machine

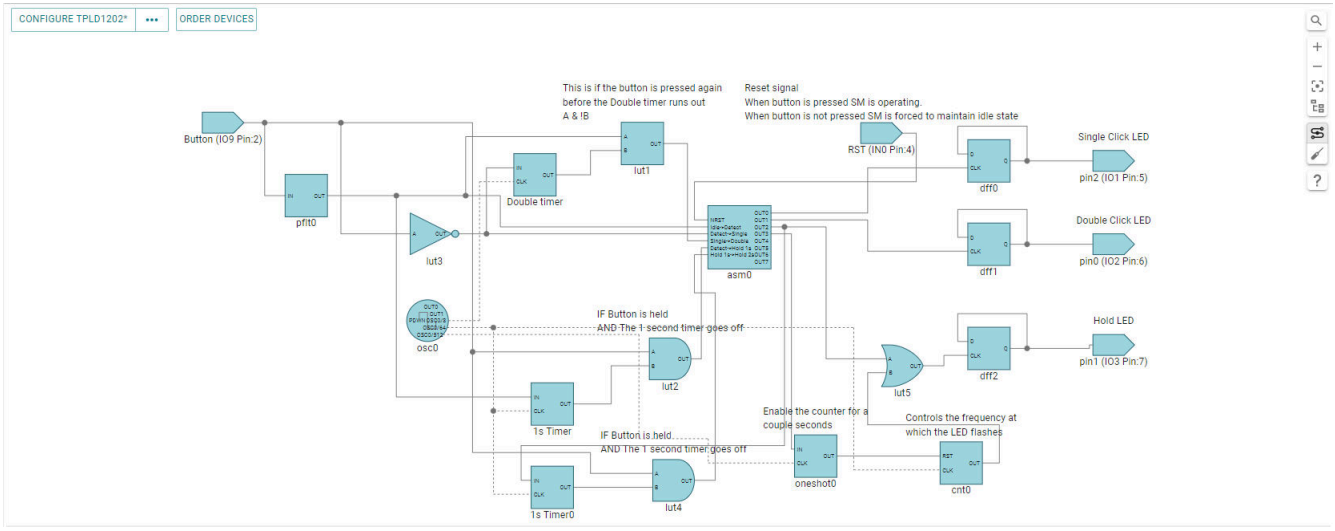

Figure 4-2. Button State Machine Built into TPLD Design

Whenever the RST pin from the design in Figure 4-2 is low the device resets into an idle state this is also the default startup state of the device. Whenever the RST pin is high (button on the EVM is pressed) the SM is active and watching for inputs albeit still in the idle state. Assume for the rest of this section the RST is being held high. A single click of the button triggers the double timer block, and transitions our SM to the Single state. The duration of this timer can be configured by increasing or decreasing the control data of the "Double timer" block. If the button is pressed again before the timer runs out the SM transitions from Single to Double. The outputs of Single and Double state are simply fed into a DFF CLK input thus transitioning the associated LED from ON to OFF or OFF to ON. If the button is pressed and remains pressed for the duration of the *1s Timer* block the SM transitions from Idle to Hold 1s. This does a similar toggling as the Single and Double states,

but also triggers a second *1s Timer* block. The reason the Hold 1s state does not retrigger the same block is because this SM is asynchronous and we want to prevent any accidental skips by connecting the input to two consecutive states to the same trigger. If the button remains held for another second the oneshot0 enables cnt0 to flash the DFF thus resulting in a flashing LED. The behavior of this design can be observed in Figure 4-3, and Figure 4-4.

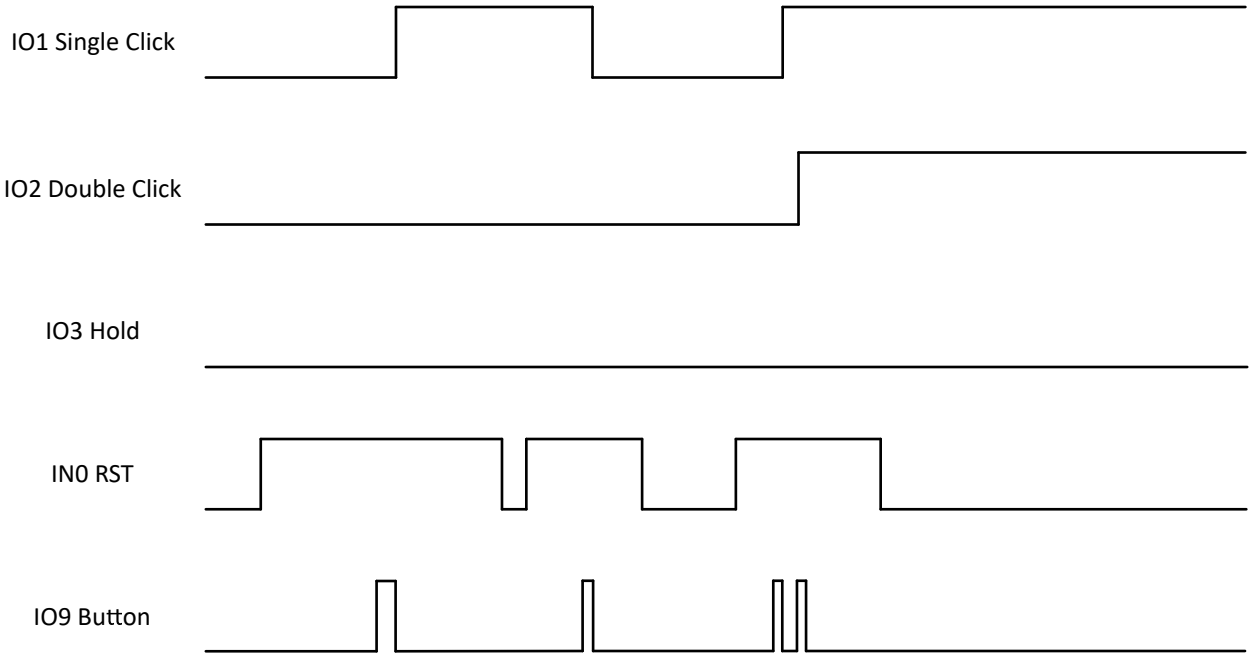

Figure 4-3. Waveform of Single and Double Click Behavior

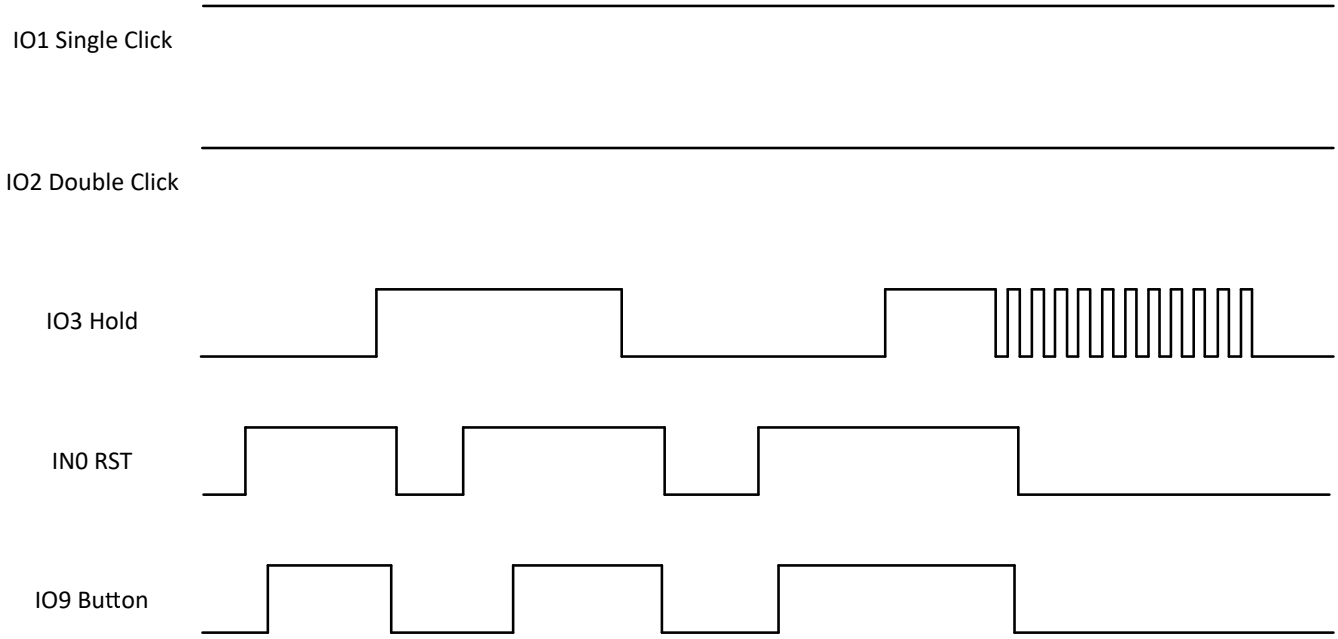

Figure 4-4. Waveform of Hold Behavior

The functionality of this circuit is just to highlight the SM, so the surrounding functionality while necessary to accomplish this task is largely simplified versus a final design.

5 Summary

State machines are central to many designs. The single press button design can be utilized as a starting point in your design by selecting the TPLD1202 device in InterConnect Studio, and searching for smart button controller. TI's new programmable logic family gives a great way to declare and design a state machine without having to use programming languages, an FPGA, or an MCU.

More information for the TPLD family of devices can be found at *Programmable Logic Devices (PLD)* and refer to Table 5-1 for some available devices and evaluation modules to prototype.

Table 5-1. Order Table

Device	EVM
All TPLD	TPLD-PROGRAM
TPLD1202DYYR	TPLD1202-DYY-EVM
TPLD1202RWBR	TPLD1202-RWB-EVM

6 References

- Texas Instruments, *InterConnect Studio*, user's guide.

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265