Keypad Scanner

Introduction

This reference design illustrates the implementation of an Matrix Keypad Scanner for an industry standard 7x7 matrix keypad. The key press is detected and the key location is provided to the user on a 7-bit data bus.

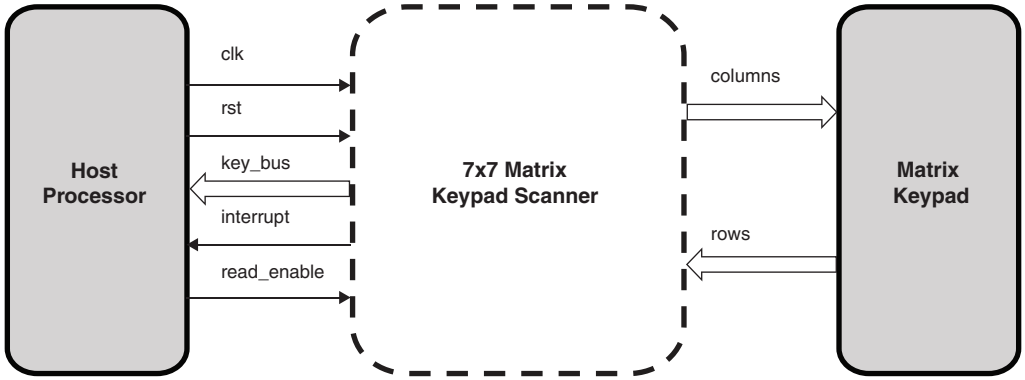
The design is implemented in VHDL. The Lattice iCEcube2™ Place and Route tool integrated with the Synopsys Synplify Pro® synthesis tool is used for the implementation of the design. The design can be targeted to other iCE40™ FPGA product family devices.

Features

- Parameter configurable key de-bounce time
- Parameter configurable auto sleep time
- Support for 7x7 matrix keypad

System Block Diagram

Figure 1. System Block Diagram



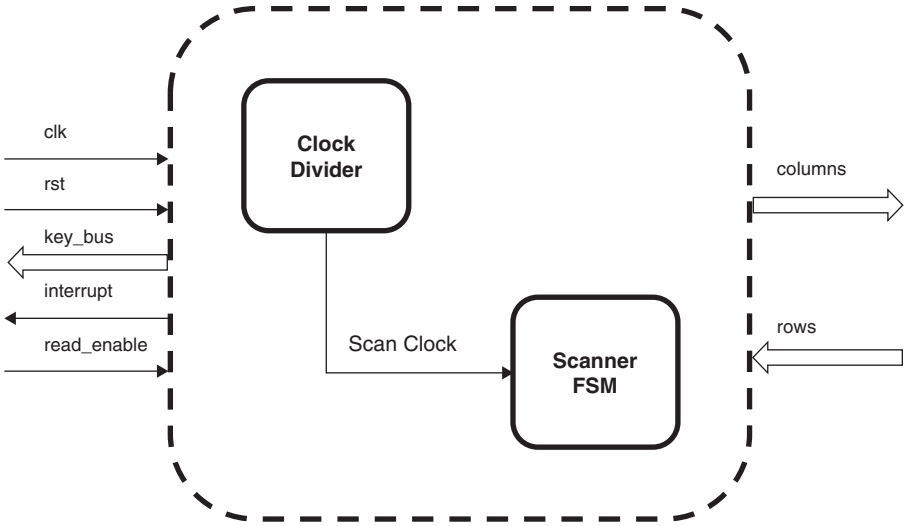
Signal Description

Table 1. Signal Description

Signal	Width	Type	Description
clk	1	Input	System Clock
rst	1	Input	Asynchronous active high system Reset
rows	7	Input	7 bit wide row input bus with pull-ups
columns	7	Output	7 bit wide column output bus
key_bus	7	Output	An active high on the key_bus(6) indicates the valid key. key_bus(5:3) indicates the columns address. Key_bus(2:0) indicates the row address.
interrupt	1	Output	High when FIFO becomes half full and remains high till FIFO is empty
read_enable	1	Input	An active high signal. When high, data present in the FIFO is sent on key_bus to processor

Design Module Description

Figure 2. Design Model



The following is a brief description of the internal blocks.

Clock Divider

Input clock is divided by 16 to generate Scan clock, used by the scanner FSM. This is a divide by 16 clock.

Scanner FSM

Whenever a key is pressed, the row and column corresponding to that key are connected, causing the row input to go low. The keypad scanner exits its sleep mode and starts scanning each column to determine the location of the key that was pressed. The column which is being scanned is driven low while others are tri-stated. If the key which is pressed lies in this column, one of the row inputs will go low, revealing its location. When the required column is detected, the scanner maintains it at ground and checks whether the same row is still low after a small delay. This is done to debounce the key.

After the location of the key has been determined and debounced, the keypad scanner outputs its location in terms of its row and column (3-bits each in the case of a 7x7 matrix) address along with a valid bit. The valid bit remains high as long as the key is pressed. Upon release of the key, this bit returns to zero. The key release event is debounced using a small delay, similar to the case of a key press.

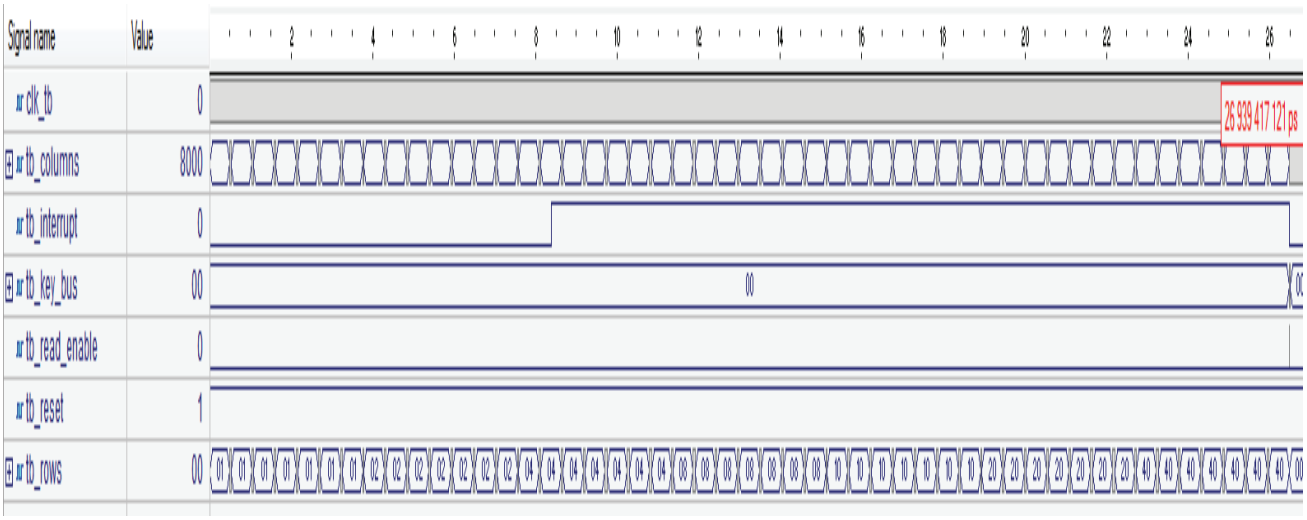
After exiting from sleep mode, the keypad scanner continues to scan each column repeatedly for some time before it goes back into its sleep mode.

Operation Sequence

1. The design is reset using the reset pin provided.
2. Whenever a key is pressed, the row and column corresponding to that key are connected, causing the row input to go low. The keypad scanner exits its sleep mode and starts actively scanning each column one by one to determine the location of the key that was pressed. The column which is being scanned is driven low while others are tri-stated. If the key which is pressed lies in this column, one of the row inputs will go low, revealing its location. When the required column is detected, the scanner maintains it at ground and checks whether the same row is still low after a small delay. This is done to debounce the key.
3. After the location of the key has been determined and debounced, the keypad scanner outputs its location in terms of its row and column (3-bits each in the case of a 7x7 matrix) address along with a valid bit. The valid bit remains high as long as the key is pressed. Upon release of the key, this bit returns to zero. The key release event is debounced using a small delay, similar to the case of a key press.
4. After waking up, the keypad scanner continues to scan each column repeatedly for some time before it goes back into its sleep mode.

Simulation Waveforms

Figure 3. Simulation Waveforms



Implementation

This design is implemented in VHDL. When using this design in a different device, density, speed or grade, performance and utilization may vary.

Table 2. Performance and Resource Utilization

Family	Language	Utilization (LUTs)	f_{MAX} (MHz)	I/Os	Architectural Resources
iCE40[1]	VHDL	174	>50	35	(46/160)PLBs

1. Performance and utilization characteristics are generated using iCE40LP1K-CM121 with iCEcube2 design software.

References

- iCE40 Family Handbook

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-503-268-8001 (Outside North America)
e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

Revision History

Date	Version	Change Summary
May 2013	01.0	Initial release.