EECS2070 02
Digilent Basys3 FPGA Board
Part 4

Ref: Digilent Basys3™ FPGA Board Reference Manual

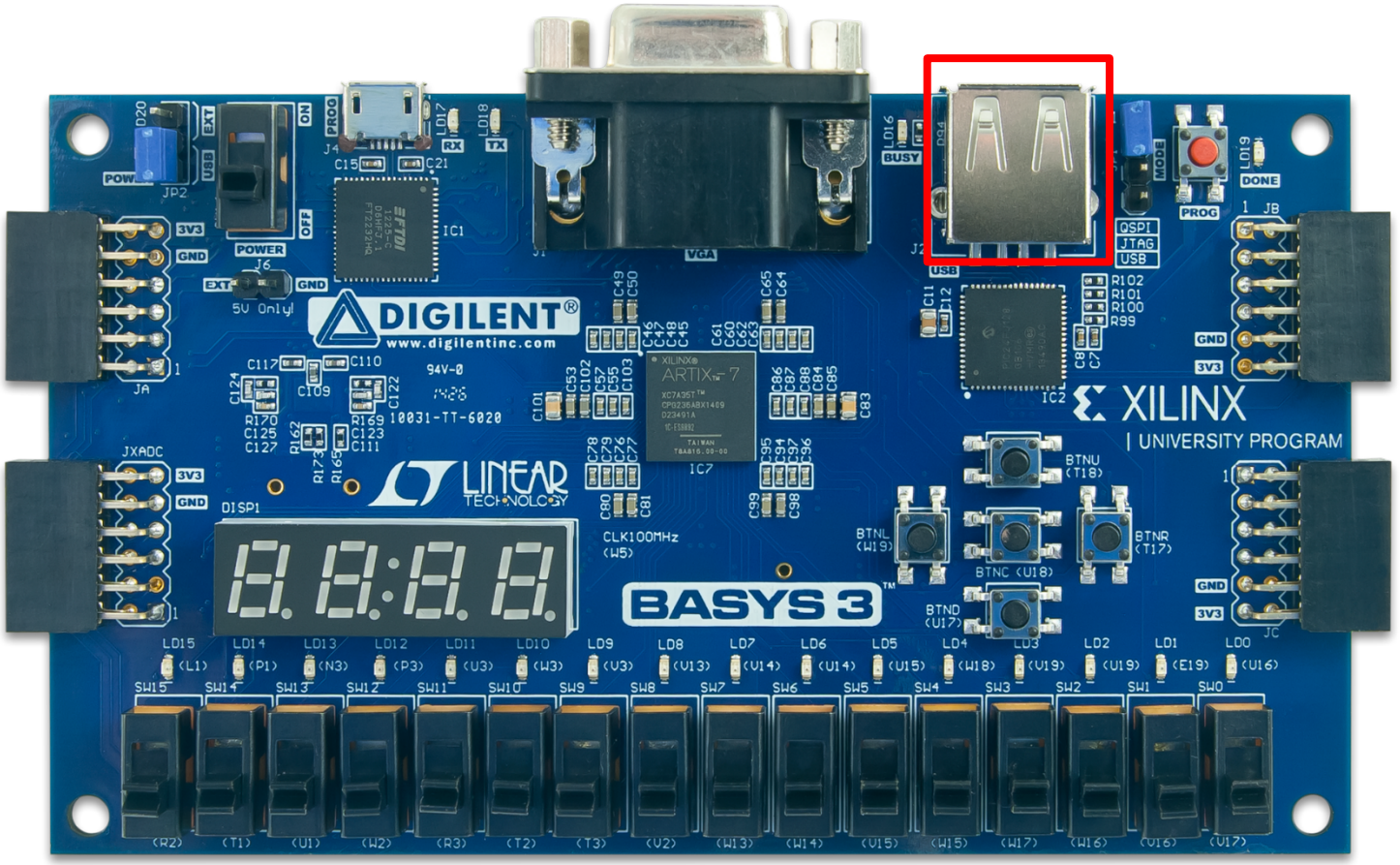
黃稚存

國立清華大學
資訊工程學系

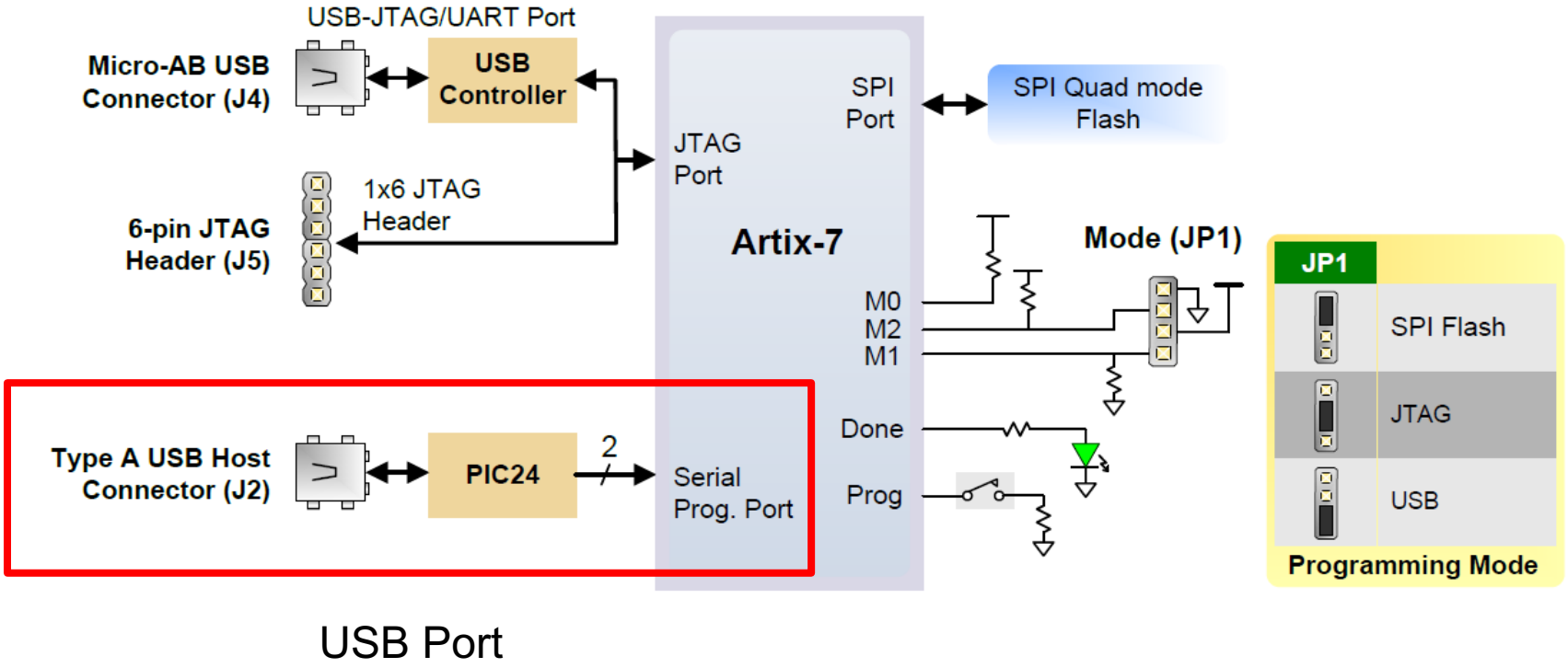
Lecture 09

Basic Protocols of Keyboard Control

USB HID (Human Interface Device) Host (1/3)



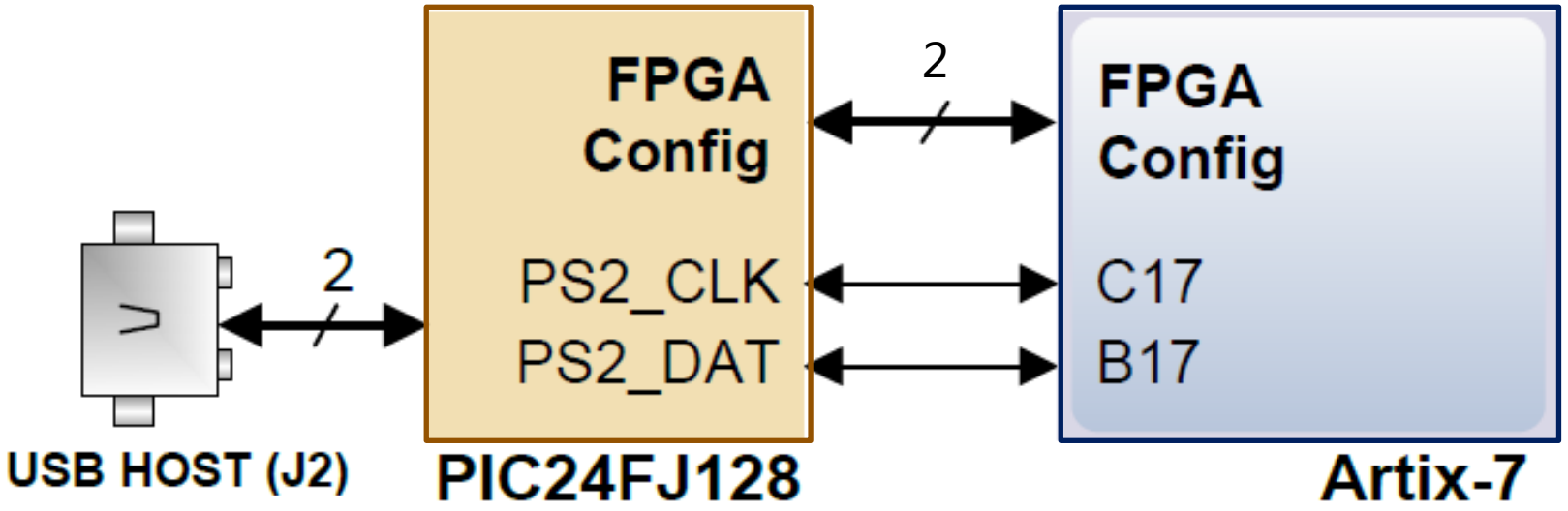
USB HID Host (2/3)



USB Port

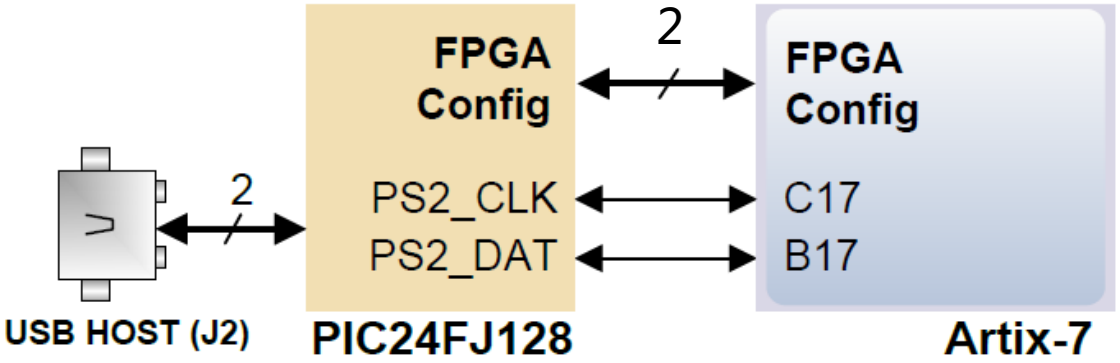
USB HID Host (3/3)

- The USB port is controlled via a PIC24 chip
  - The PIC24 chip translates USB signals to PS2 signals
  - The two interface ports on the FPGA are B17 and C17



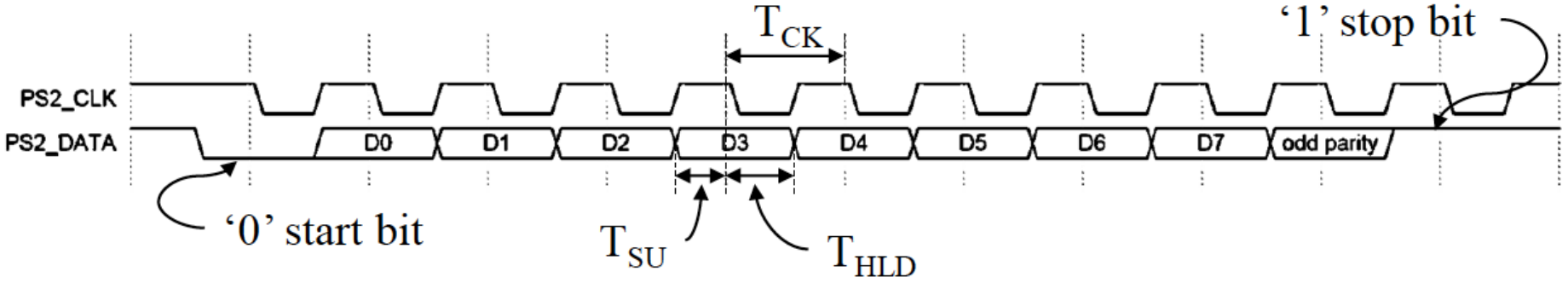
Microchip PIC24FJ128

- Configuration mode
  - Download the bitstream to the FPGA.
- Application mode
  - On Basys 3, this is called USB HID Host mode
  - Hub support is not currently available
  - Only a single mouse or a single keyboard can be used
  - PS2_CLK and PS2_DATA are used to implement a standard PS/2 interface



HID Controller

- Every time the keyboard transmits 8 bits to FPGA
  - The 8 bits are followed by a parity bit
  - Each bit is read on the falling edge of the clock



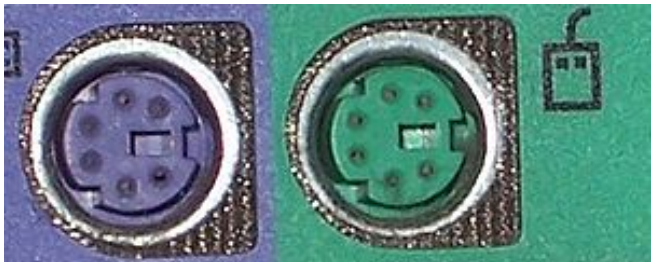
Symbol	Parameter	Min	Max
T_{CK}	Clock time	30 us	50 us
T_{SU}	Data-to-clock setup time	5 us	25 us
T_{HLD}	Clock-to-data hold time	5 us	25 us

Initialization

- When a keyboard or mouse is connected to the Basys 3, a “self-test passed” command (0xAA) is sent to the Basys 3
  - The command (0xAA) is automated sent by the keyboard to FPGA
  - This command informs FPGA that the keyboard is ready
  - Only sent once when the system turns on
- The Basys 3 board may request the keyboard to do several times of self-test by command (0xFF)
  - The keyboard uses 0xFA to respond to the requests, and may send “self-test passed” again to Basys 3
  - Keyboard : 0xFA (0xFA 0xAA)
- The protocol is already implemented
  - You don’t need to worry about it
  - But it is still good to know the concept

PS/2 Port

- PS/2 port was formerly used for keyboards and mice
- Nowadays, they are replaced by USB ports
- Their protocols are still widely used
- Scancodes are used to communicate with the PS/2 port



Example PC compatible (IBM PS/2) scancodes

key	set 1 (IBM PC XT)		set 2 (IBM PC AT)		set 3 (IBM 3270 PC)	
	press	release	press	release	press	release
A (normal letter)	1E	9E	1C	F0 1C	1C	F0 1C
Return / Enter (main keyboard)	1C	9C	5A	F0 5A	5A	F0 5A
Enter (numeric keypad)	E0 1C	E0 9C	E0 5A	E0 F0 5A	79	F0 79
Left Windows key	E0 5B	E0 DB	E0 1F	E0 F0 1F	8B	F0 8B
Right Windows key	E0 5C	E0 DC	E0 27	E0 F0 27	8C	F0 8C

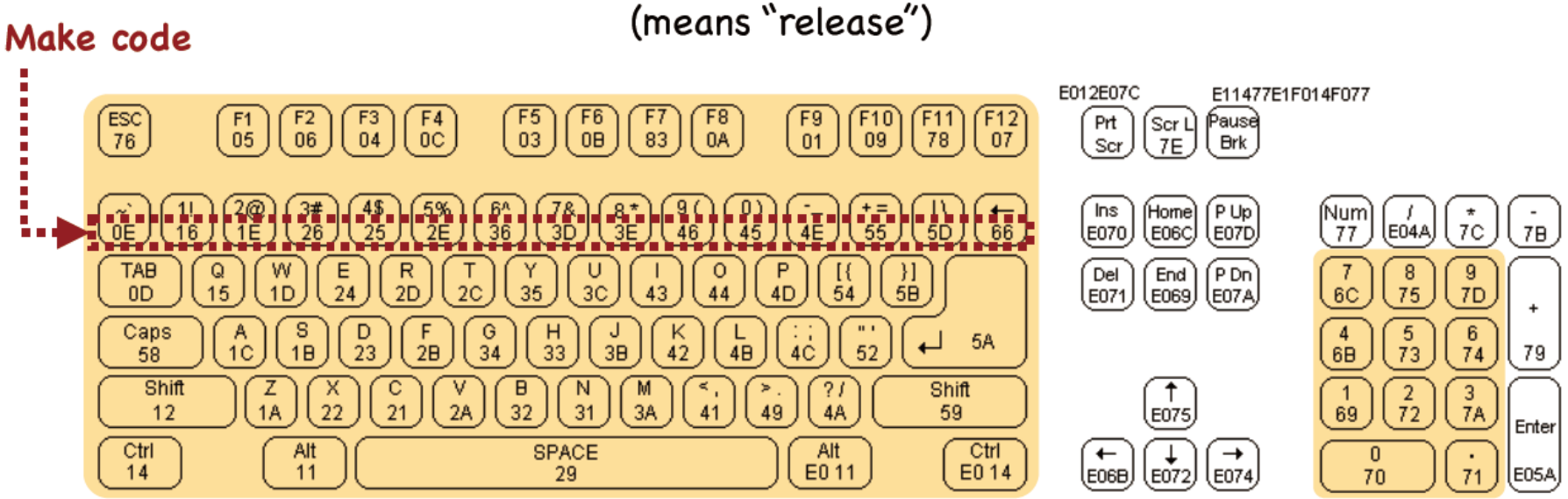
from Wiki

PS/2 Scancode

- Three types of scancodes
  - Make codes represent the key values
  - Break code represents the action of “release the key”
  - Extend code represent the duplication of a key

Extend Code	Break Code	Make code
E0	F0	XX

(means “release”)

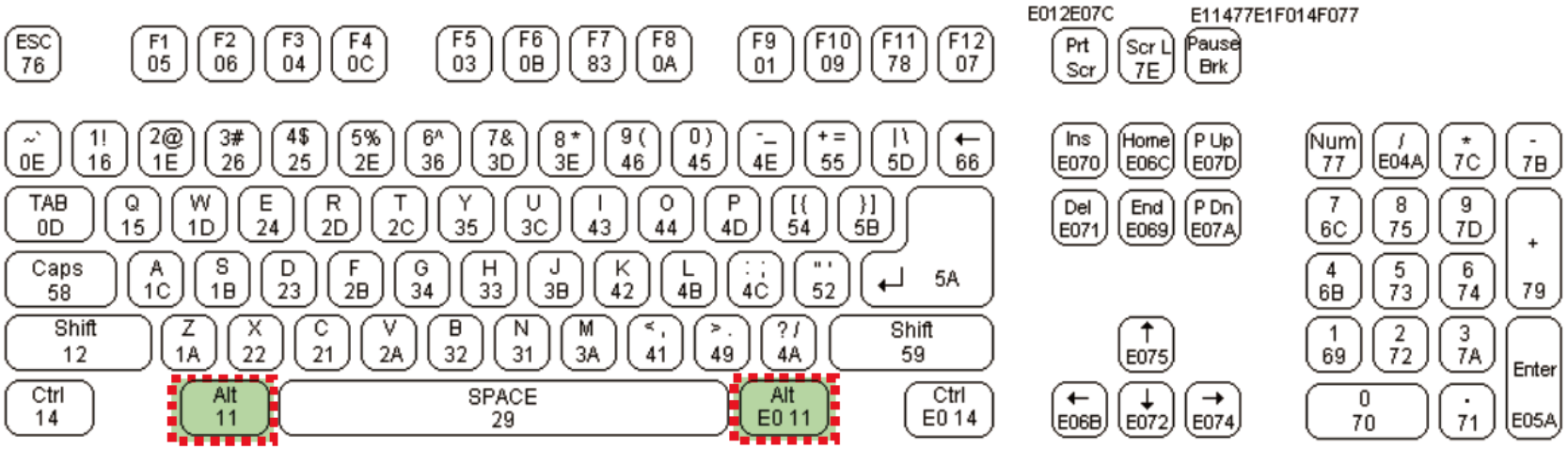


We only use the yellow parts of the keyboard

PS/2 Scancode (Example)

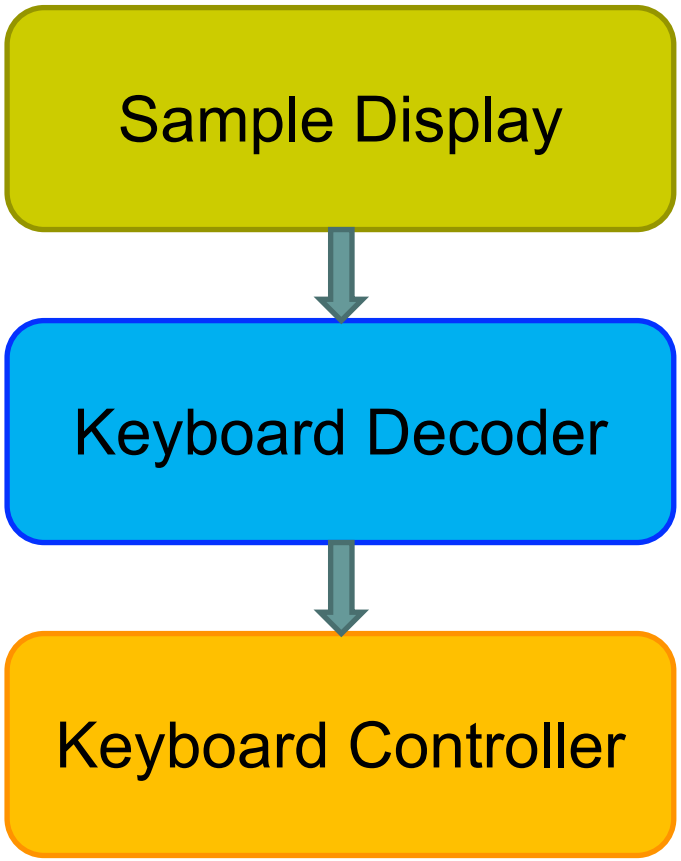
- Two “Alt” key are available on the keyboard
  - Additional keys are encoded as <extend code> + <make code>
  - When releasing a key, a <break code> is inserted

L Alt press			11
L Alt release		F0	11
R Alt press	E0		11
R Alt release	E0	F0	11



Keyboard Controller

Program Hierarchy



SevenSegment.v

SampleDisplay.v

KeyboardDecoder.v

KeyboardCtrl.v

Ps2Interface.v

Vivado IP

OnePulse.v

KeyboardCtrl.v

Verilog Module: KeyboardCtrl (1/2)

- In Keyboard-Controller
  - Ps2Interface.v
  - KeyboardCtrl.v
- I/Os for KeyboardCtrl.v

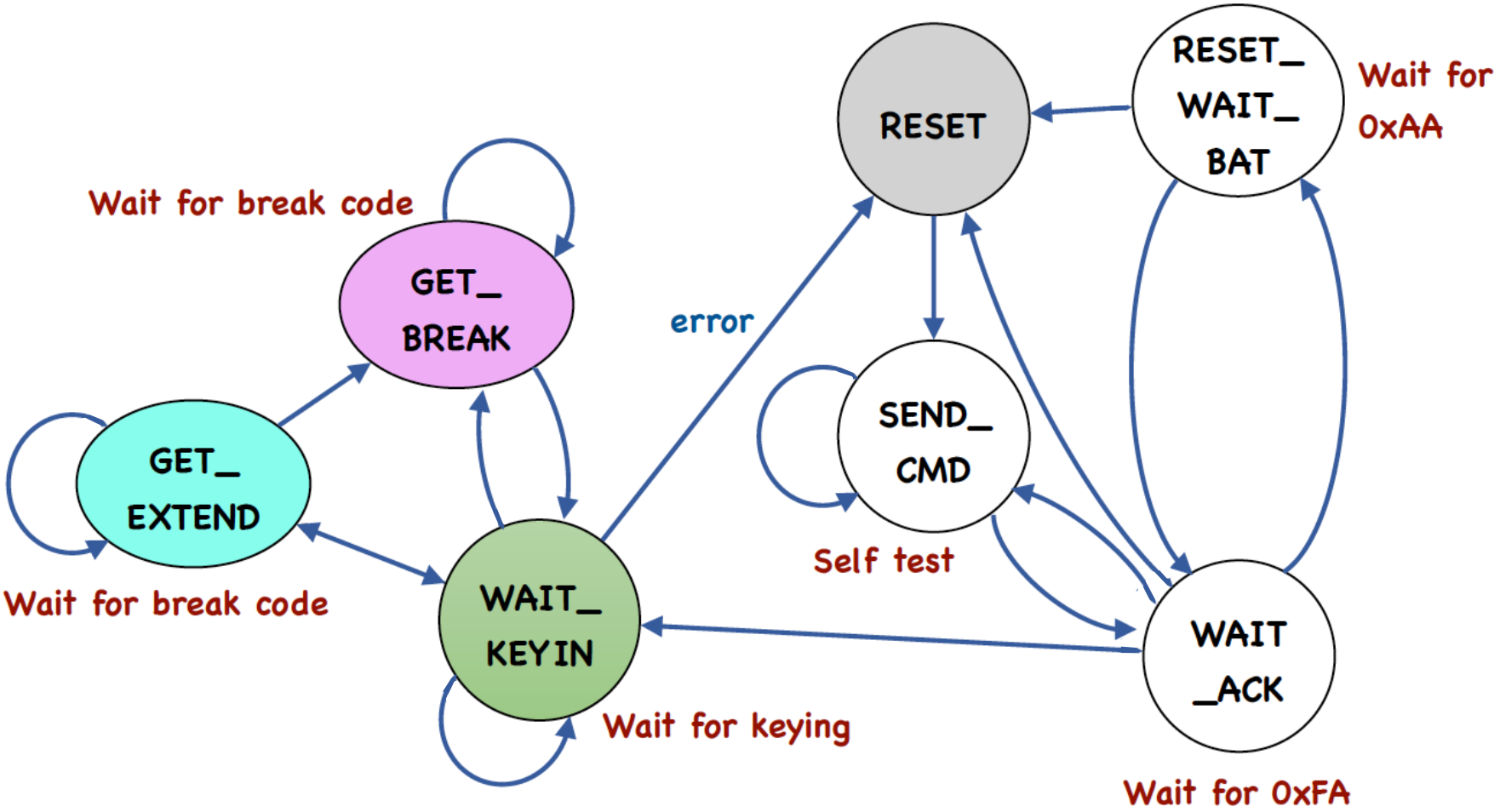
Output

- key_in
- is_extend
- is_break
- valid
- err

Input

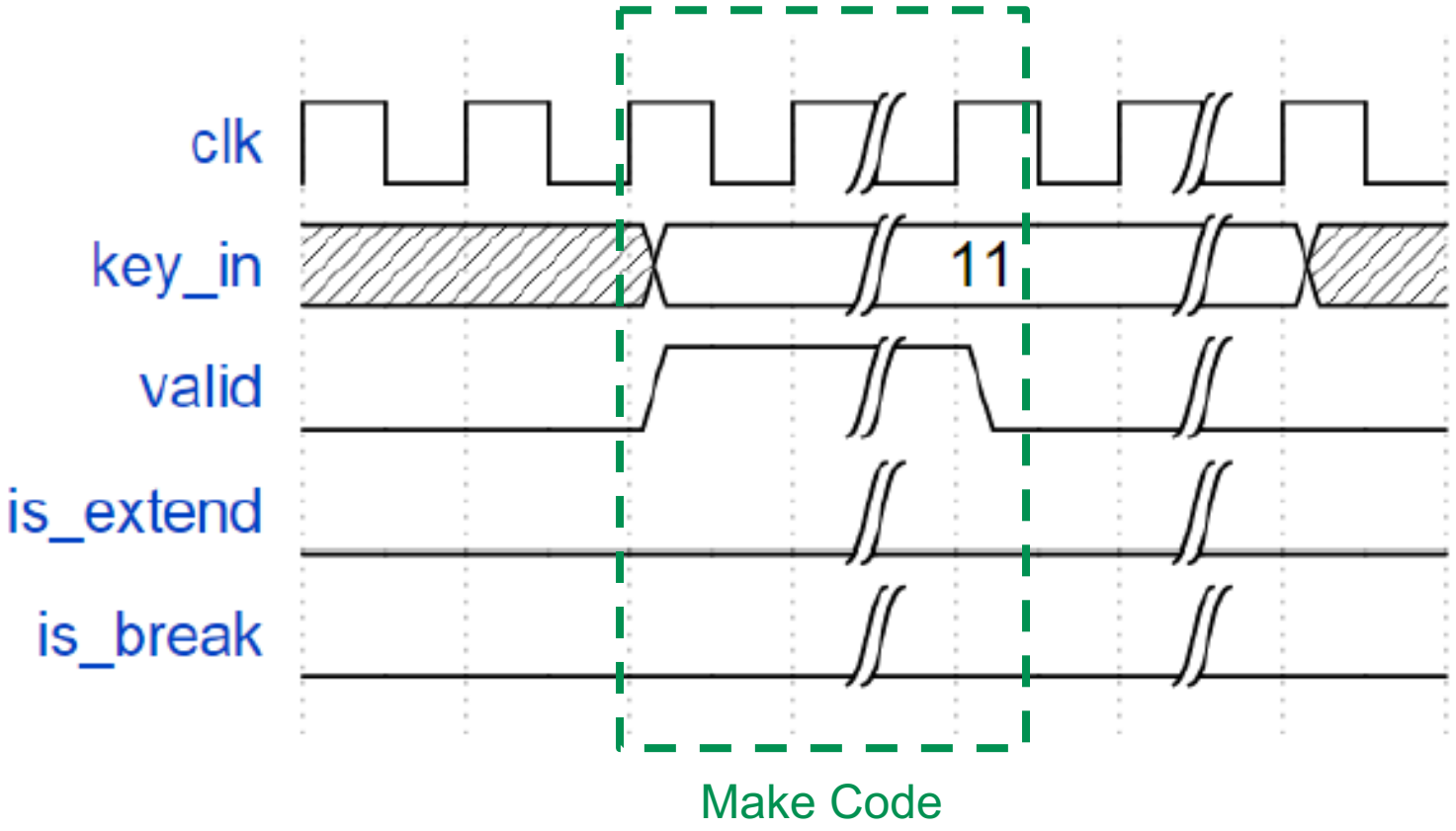
- PS2_CLK
- PS2_DATA
- rst
- clk

Verilog Module: KeyboardCtrl (2/2)



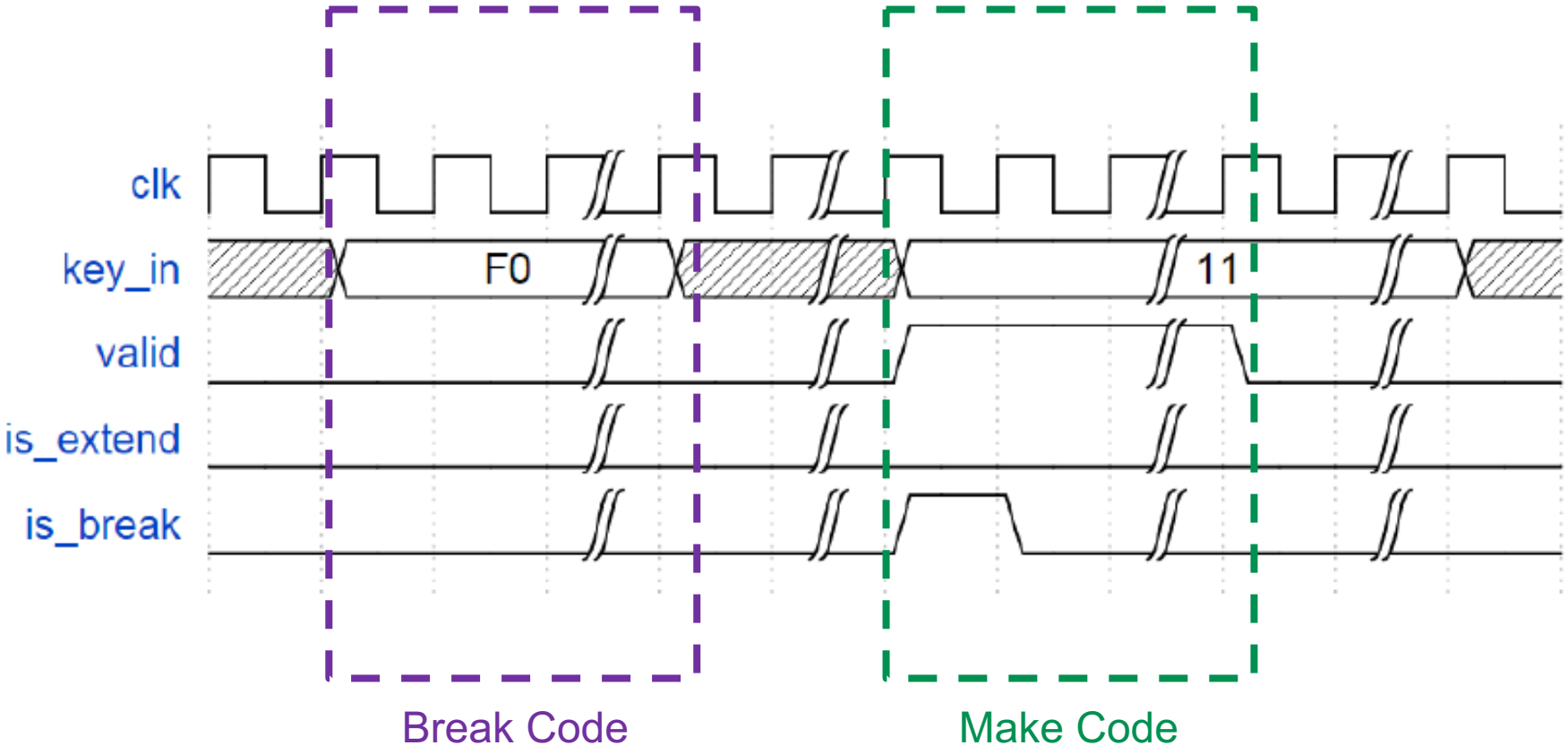
KeyboardCtrl (Output Example 1)

- L Alt pressed



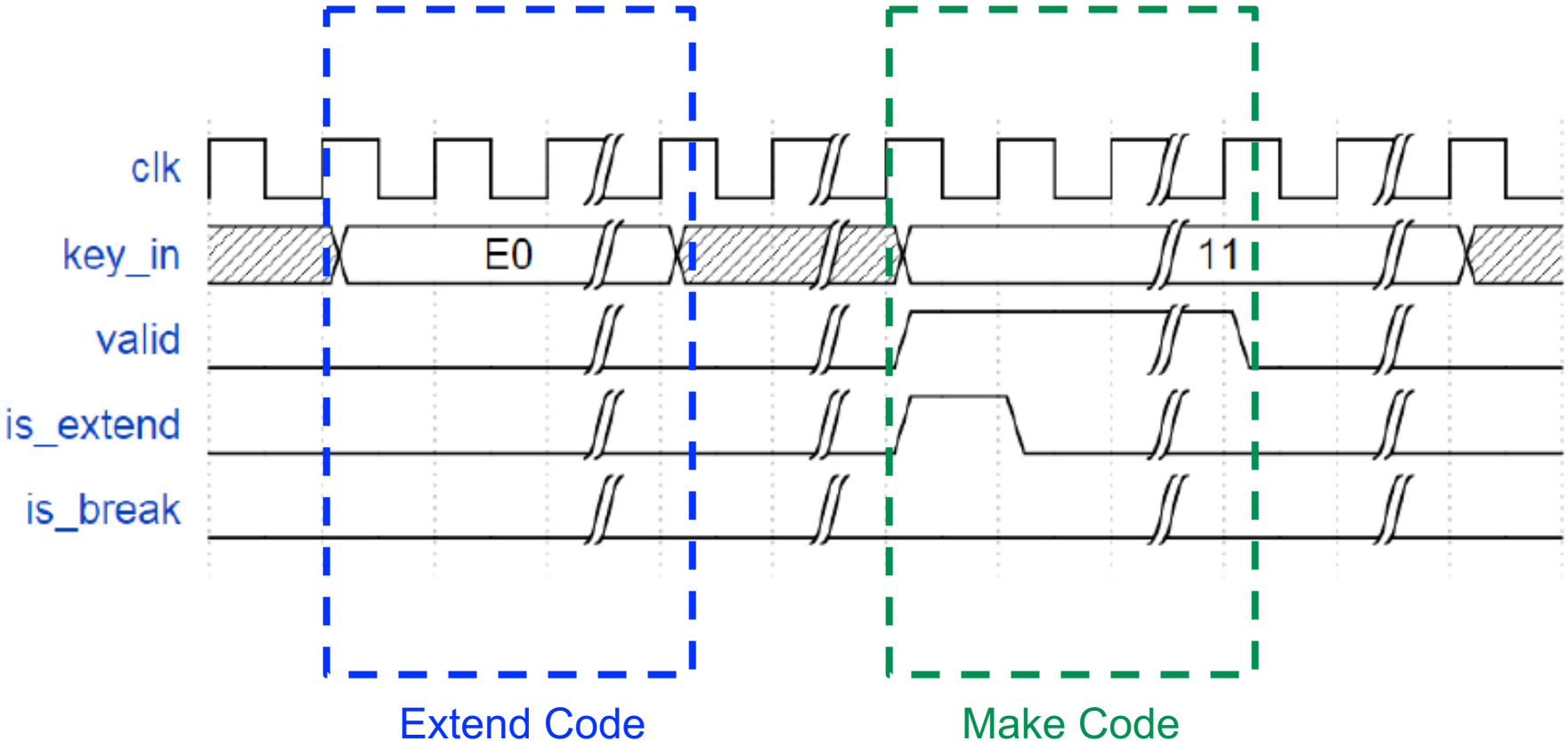
KeyboardCtrl (Output Example 2)

- L Alt released



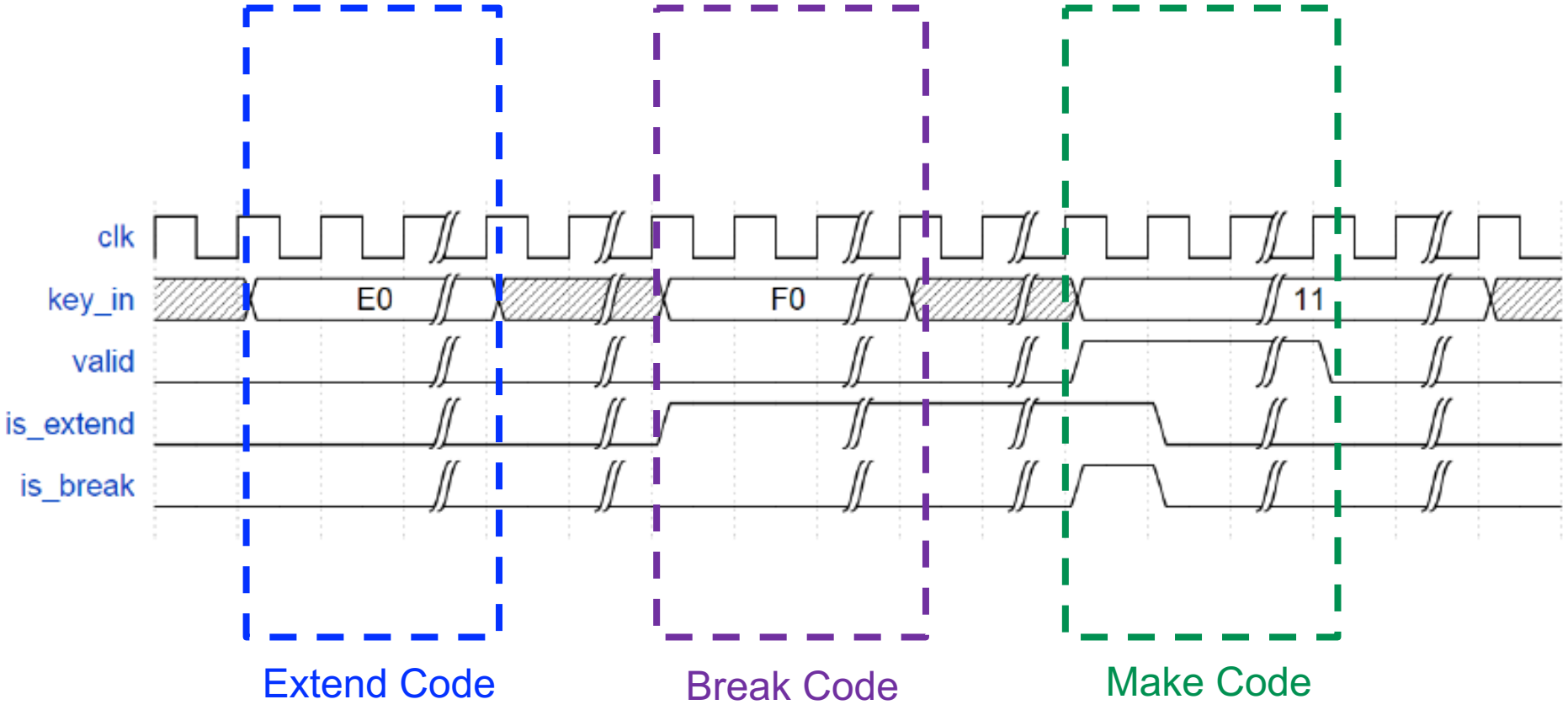
KeyboardCtrl (Output Example 3)

- R Alt pressed



KeyboardCtrl (Output Example 4)

- R Alt released



KeyboardDecoder.v

Verilog Module: KeyboardDecoder (1/5)

- In Keyboard Sample Code
  - KeyboardDecoder.v
  - SampleDisplay.v
- I/O for KeyboardDecoder

Output

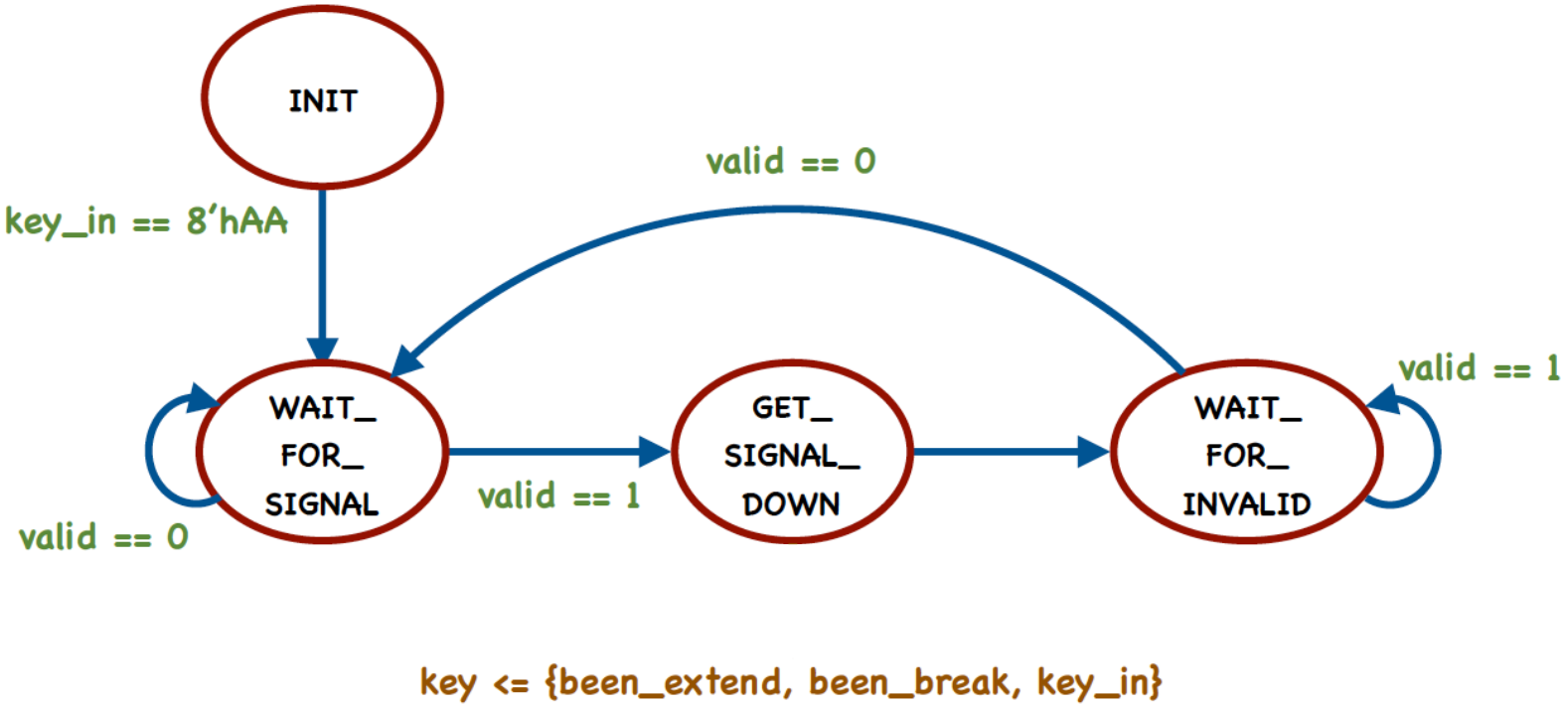
- key_down
- last_change
- Key_valid

Input

- PS2_CLK
- PS2_DATA
- rst
- clk

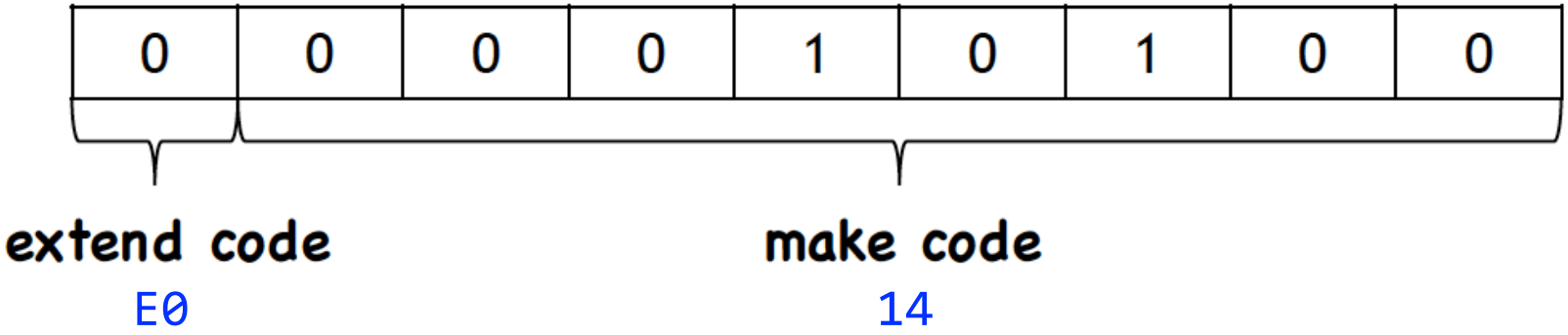
Verilog Module: KeyboardDecoder (2/5)

- Retrieving “extend”, “break”, and “key” from KeyboardCtrl
  - Getting value only when valid = 1’b1



Verilog Module: KeyboardDecoder (3/5)

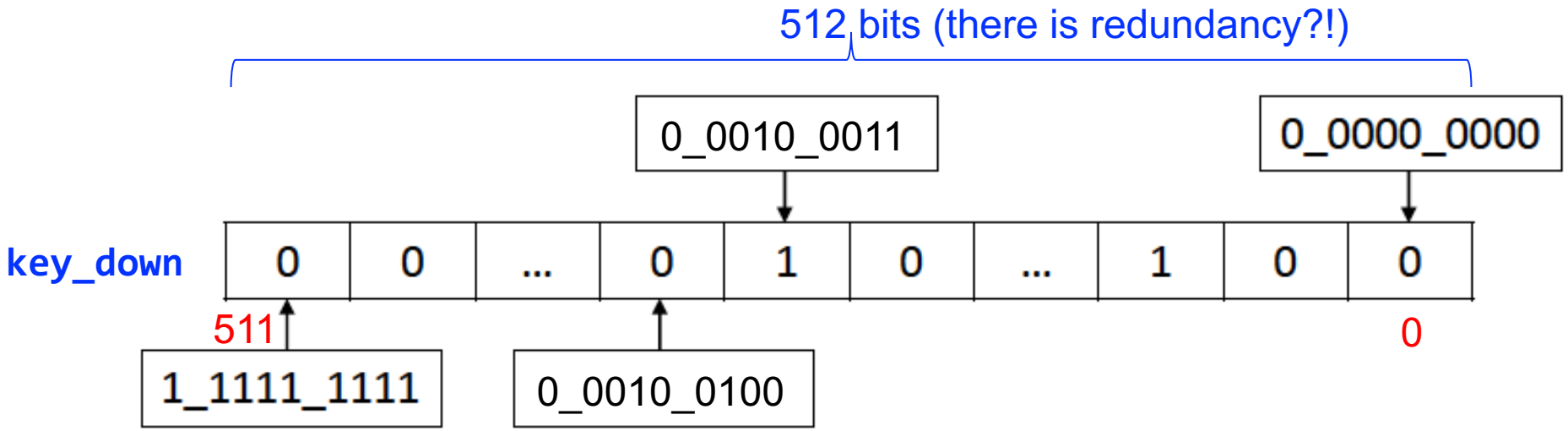
- last_change: 9 bits
  - Represent the key which has been pressed or released



- key_valid: 1 bit
  - Should be active for one clock period (100MHz) when any key is pressed or released

Verilog Module: KeyboardDecoder (4/5)

- Use an array "**key_down**" of 512 bits to record which key is currently being pressed
  - **1** means "key is pressed", while 0 means "key is released"
  - Ex: the key indexed by "0_0010_0011" ("D": code 23) is pressed
  - Ex: the key indexed by "0_0010_0100" ("E": code 24) is released
  - Ex: the key indexed by "1_0101_1010" is "Right Enter" (Num Pad) (code E0 5A)



- Use another array "**key_decode**" (also 512 bits) to represent the key that was just pressed or released

Verilog Module: KeyboardDecoder (5/5)

- To represent that a key is pressed
  - `key_down <= key_down | key_decode;`

key_down		0	1	1	0	1
key_decode	or	0	0	0	1	0
		0	1	1	1	1

- To represent that a key is released
  - `key_down <= key_down & (~key_decode)`

key_down		0	1	1	0	1
key_decode		0	0	1	0	0
~key_decode	and	1	1	0	1	1
		0	1	0	0	1

Remark

- You don’t really need an array of 512 bits to track the status of keyboard
  - Especially when you only need a few keys in the design
  - 512 + 512 = 1024 for key_down and key_decode!!
- You may optimize this scheme to save resource on FPGA

Verilog Module: SampleDisplay

- In Keyboard Sample Code
  - KeyboardDecoder.v
  - SampleDisplay.v
- I/O for KeyboardDecoder

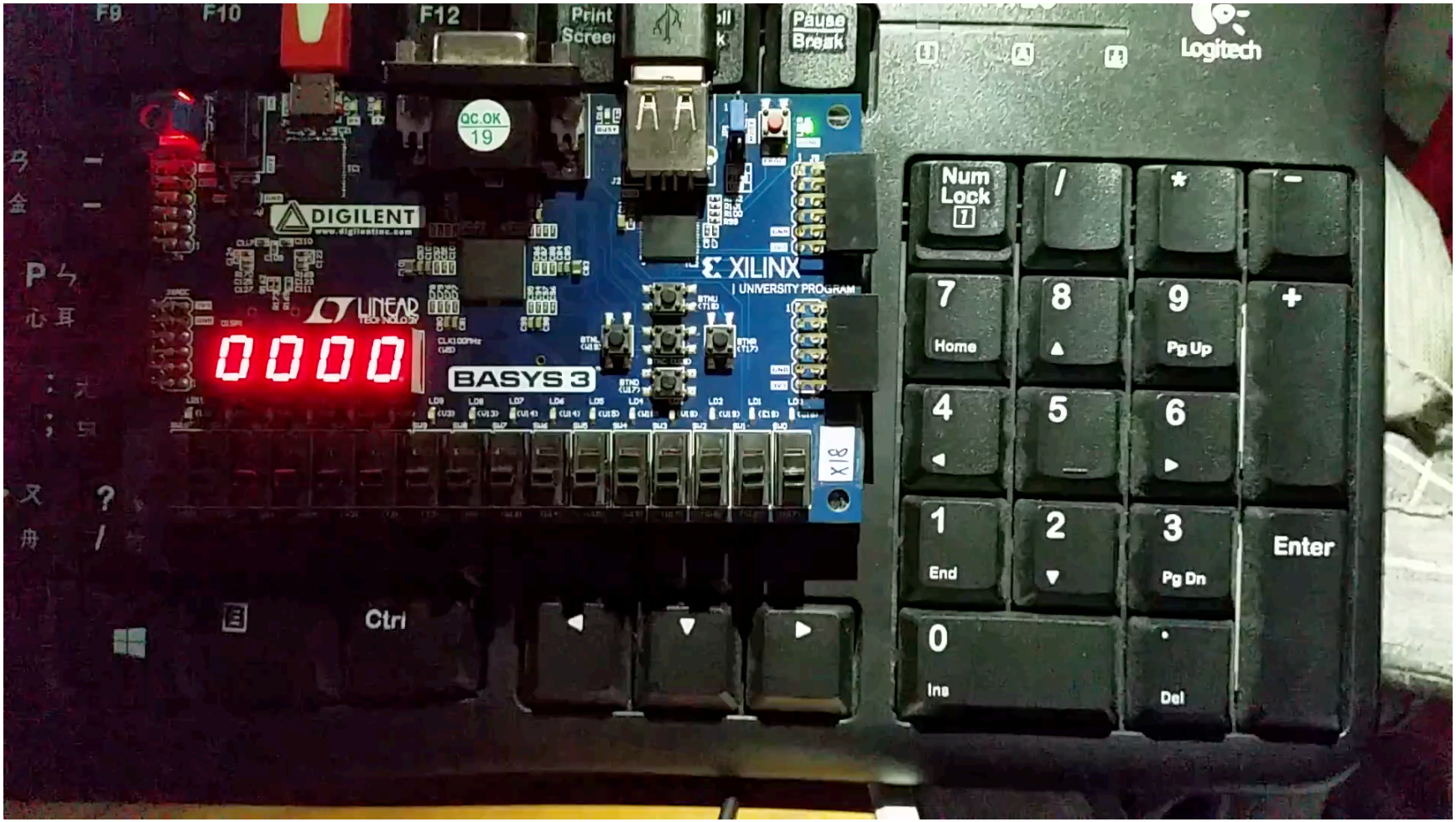
Output

- display
- digit

Input

- PS2_CLK
- PS2_DATA
- rst
- clk

Sample Display with Keyboard

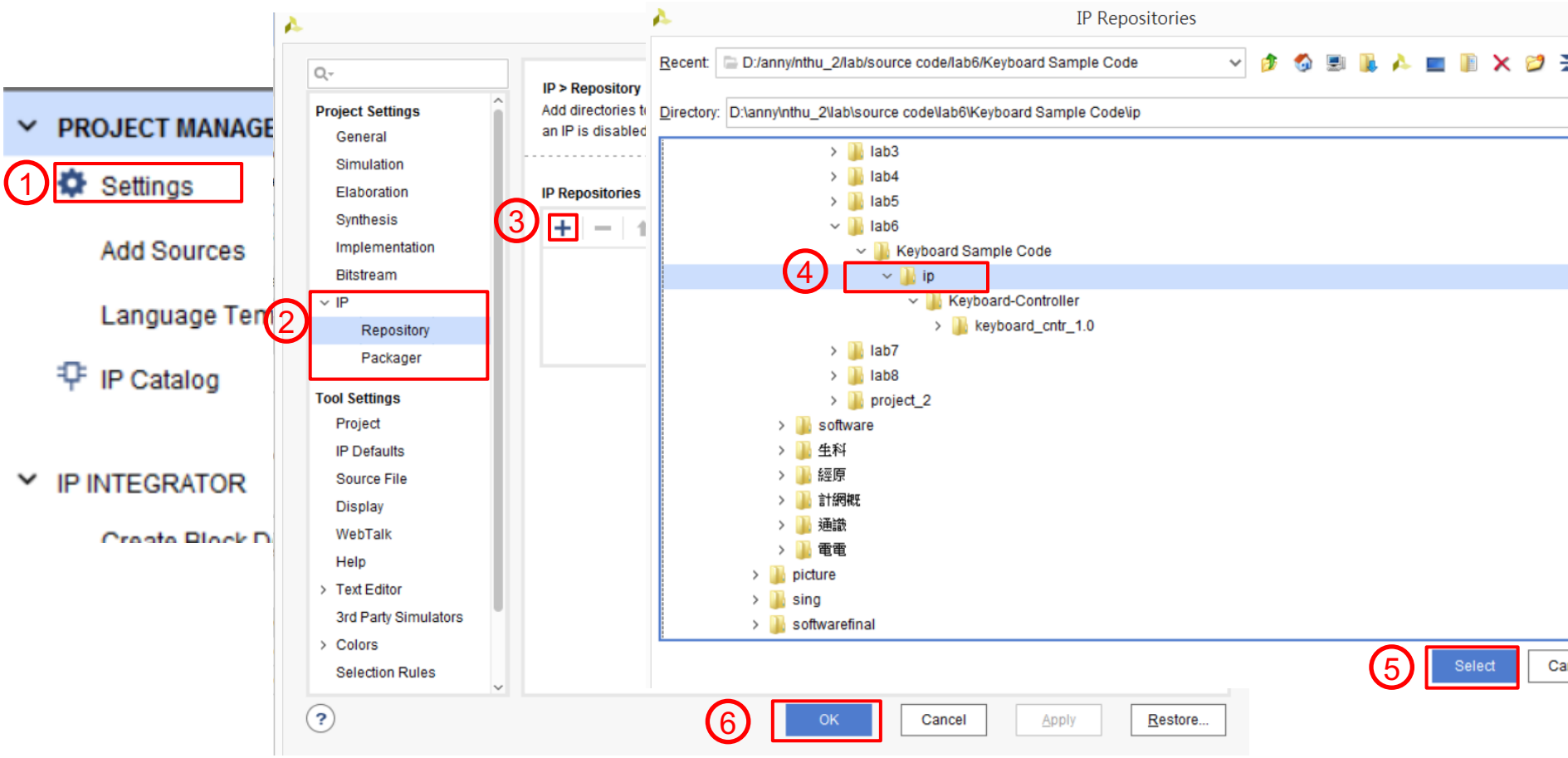


Keyboard Controller IP from Vivado

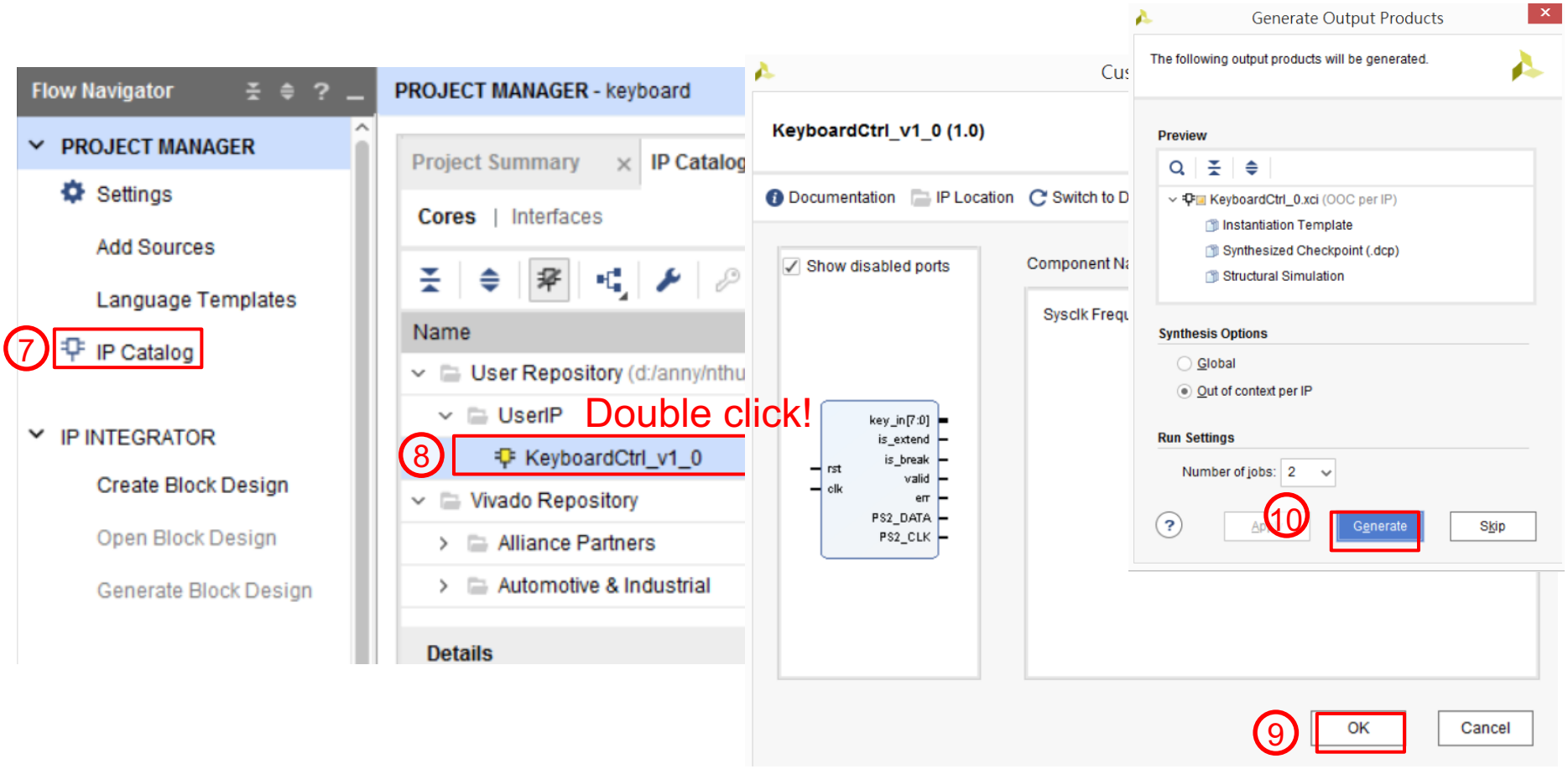
Remarks

- IP: Intellectual Property
  - Refer to a software (source / library) or a hardware design (also in its high-level description form)
- Vivado supports a large variety of IP blocks that you can use
  - Verilog RTL blocks
- You may use the follow steps to add the keyboard controller IP, or
  - Simply integrate the Verilog source we provide

How to Use IP (1/3)



How to Use IP (2/3)



How to Use IP (3/3)