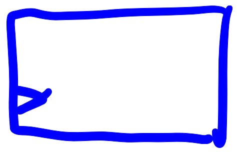
BBBs 

Process:

CSA:

ECE 383 - Embedded Computer Systems II Lecture 11 - Datapath and Control

UNITED STATES AIR FORCE ACADEMY

$$\begin{tikzcd}
 & & \text{Mini-C} \arrow[dl] \arrow[dr] & \\
 & \text{CU} \arrow[d] & & \text{DP} \arrow[d] \\
 & \text{FSM} \arrow[dl] \arrow[dr] & & \text{BBBs} \\
\text{STATE Transistion} & & \text{Output table} &
\end{tikzcd}$$

CS/SW

- Like Button / ACTION

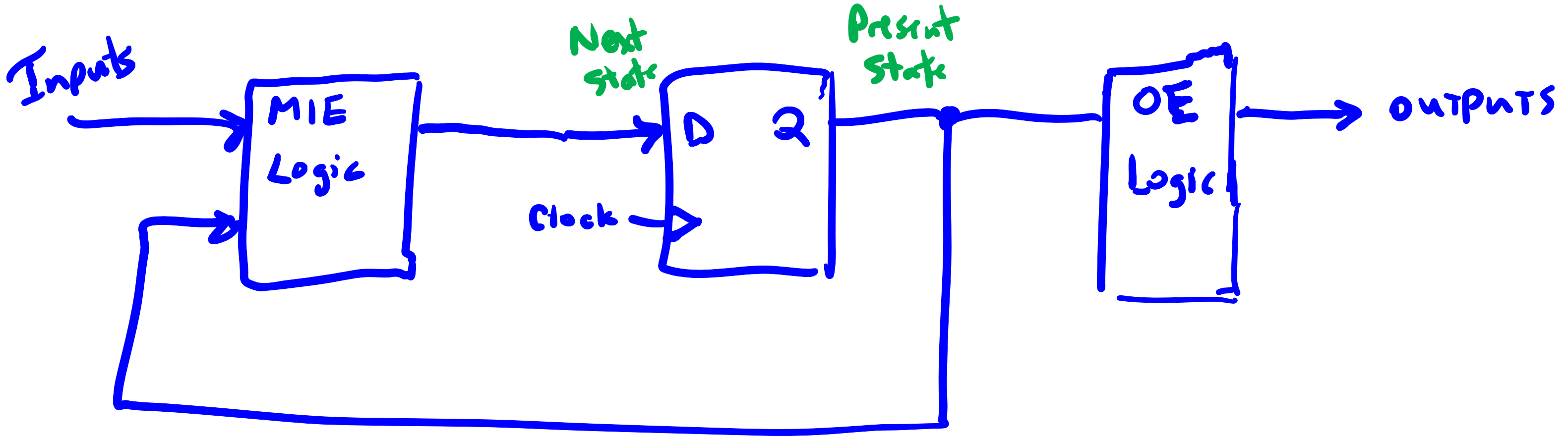
UNITED STATES AIR FORCE ACADEMY

Lesson Outline

- [redacted]
- Datapath and Control – Timing
- VHDL Instantiation
- Keyboard serial to parallel converter

Before: State Machine timing

① OE in timing?
② Hold Time in timing?



UNITED STATES
AIR FORCE
ACADEMY

Datapath and Control - Timing

Datapath and Control - Timing

- Datapath and Control Design Methodology
  - Datapath - responsible for data manipulations
  - Control - responsible for sequencing the actions of the datapath

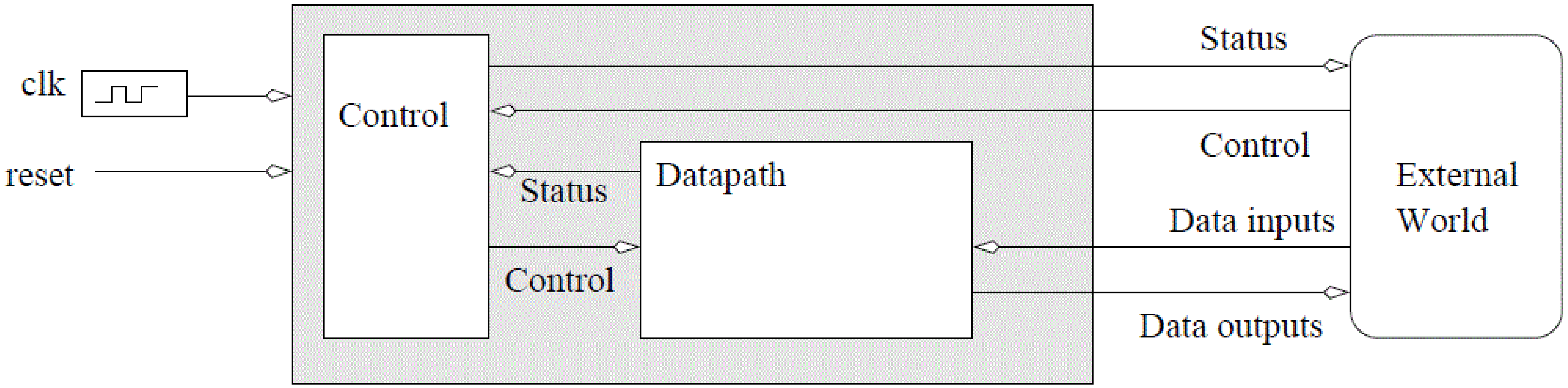


Fig 10.0 - An abstract digital system constructed from a datapath and a control unit.

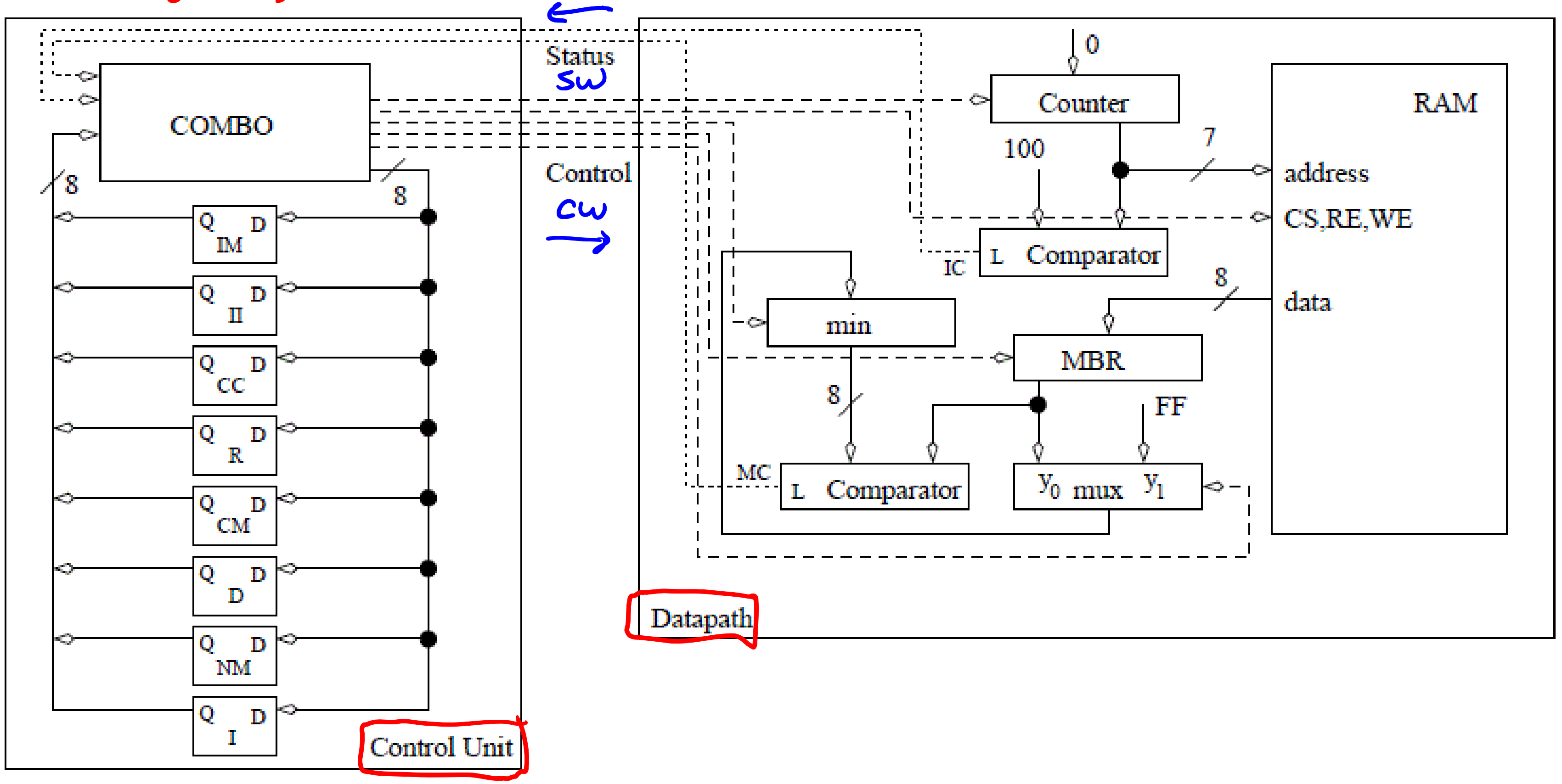
Datapath and Control - Timing

- Reasons to examine the timing behavior of a datapath and control circuit.
  1. First, so that we can make informed predictions about the expected clocking frequency of our circuits.
  2. Second, so that we can identify critical paths in our circuit.
  3. Third, so that we can develop our intuition about the operation of these complex circuits.

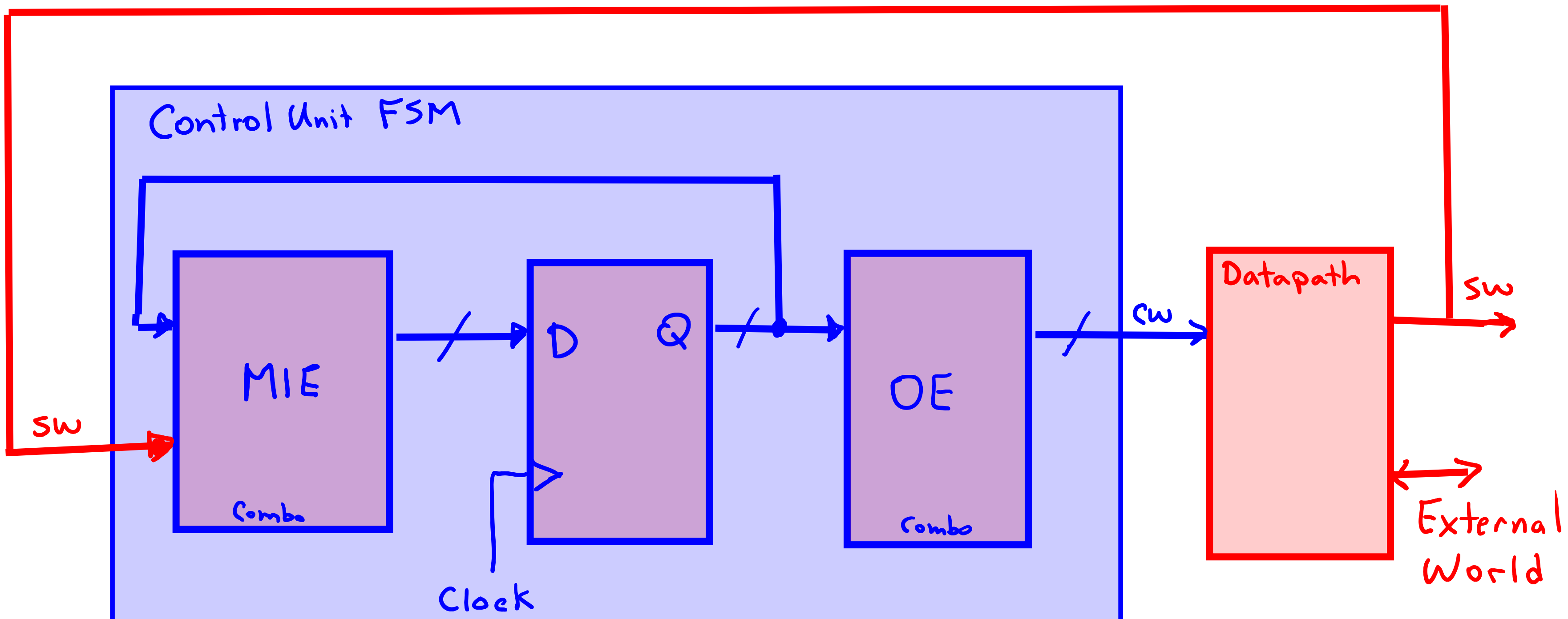
Datapath and Control - Timing

- Circuit from Lesson 10 – Search algorithm for minimum

Anything new in the critical path (from lesson 9)?



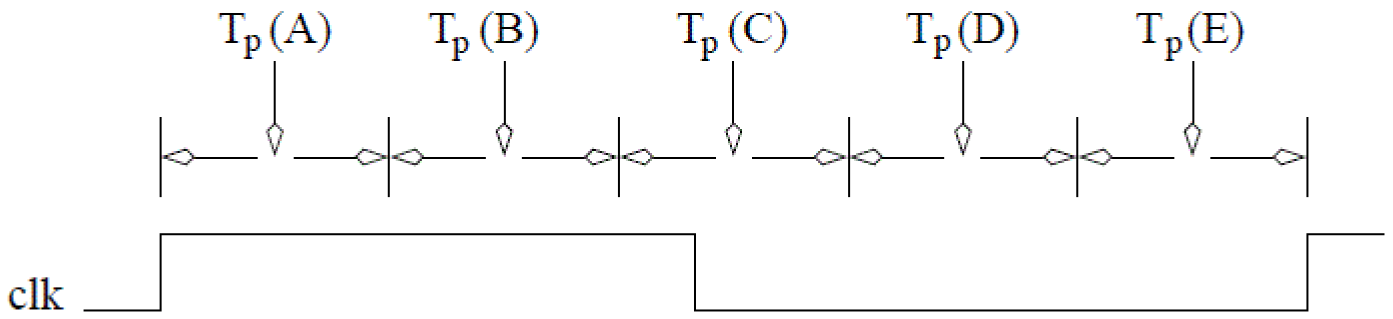
Integrity - Service - Excellence


Control Unit FSM
MIE
Combo
D
Q
Clock
OE
Combo
cw
Datapath
sw
sw
External World

Before: 3 times in critical path --- Now 5 times!

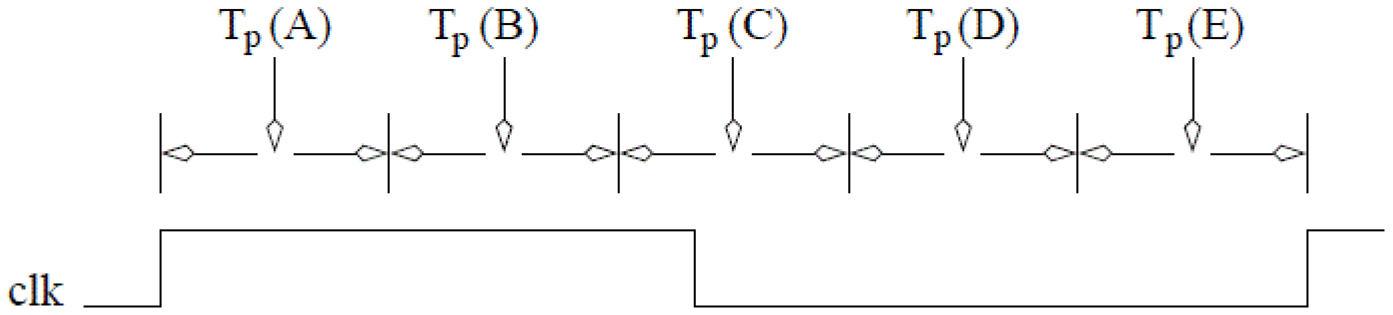
Datapath and Control - Timing

- T_p(A) – T_{pd}(FF) – Propagation Delay of Flip Flops
  - FF is input to OEs
- T_p(B) – Q_{valid} – OEs assert their new values
- T_p(C) – SW_{valid} – time difference between the application of a valid control word to the datapath and the status input to the control unit becoming valid



Datapath and Control - Timing

- T_p(D) – D_{valid} – delay between the status inputs becoming valid and the MIEs becoming valid
- T_p(E) – T_{setup} – Once the memory inputs have stabilized, they must be allowed some setup time



UNITED STATES
AIR FORCE
ACADEMY

VHDL Instantiation

VHDL Instantiation

- **Binding - technique of assigning signals in the top-level entity (caller) to the signals in the instance**

```
uut: lec10
      Generic map(5)
      PORT MAP (
        clk => clk,
        reset => reset,
        crtl => crtl,
        D => loadInput,
        Q => cntOutput);
```

  - **Port signals clk, reset, crtl, D and Q were defined inside the lec10 component**
  - **signals clk, reset, crtl, loadInput, and cntOutput were defined as signals in the higher-level testbench**

VHDL Instantiation

- We could shorten this instantiation by using the default binding calling convention shown in the code below

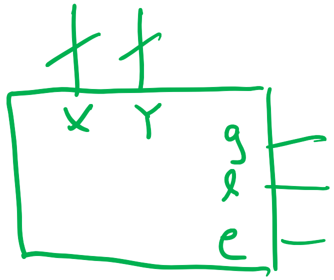
```
uut: lec10
          Generic map(5)
          PORT MAP (clk, reset, crtl, loadInput, cntOutput);
```

- Important Note: When you use the default binding, the order of the signals must match the exact same order that is defined in the entity description.
- Generates a more compact instantiation statement.

Unused outputs and OPEN keyword

- **However, we could shorten this instantiation by using the default binding calling convention below:**

```
entity compare is
    generic(N: integer := 4);
    port(x,y : in unsigned(N-1 downto 0);
        g,l,e: out std_logic);
end compare;
```

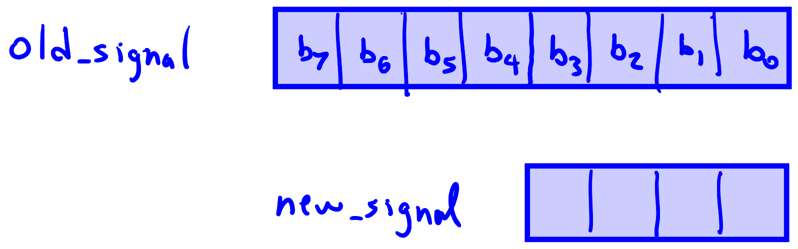


  - **example: compare port map (A, B, OPEN, OPEN, equal);**
    - Synthesis engine can remove the logic associated with any of the OPEN signals and reduce the resources used on the FPGA.

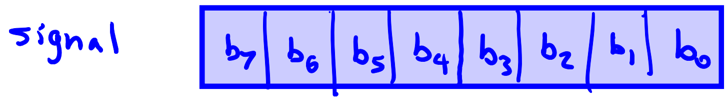
Subvectors and Concatenation

- There are times when we will need to rebuild a std_logic_vector from pieces of other vectors.
- Vector is defined as signal(7 downto 0), you can replace the limits with anything in between to get a small subvector
- For example, you could ask for signal(5 downto 2) for a 4-bit sub-vector of signal.

```
new_signal(3 downto 0) <= old_signal(5 downto 2);
```

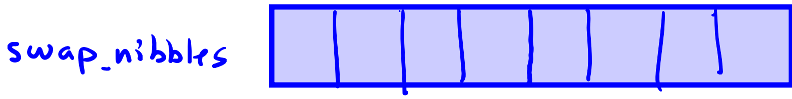


- The concatenation operation, &, is a way to "glue" two vectors together.
- For example, to build a 8-bit vector you could legally write in VHDL



4bits & 4-bits

```
swap_nibbles <= signal(3 downto 0) & signal (7 downto 4);
swap_nibbles <= lower_nibble & upper_nibble;
```



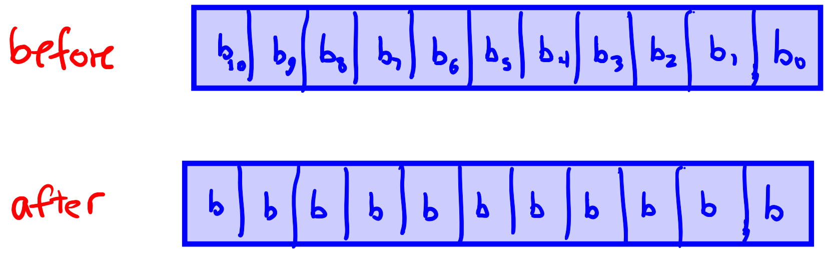
- These two concepts come together in the shift register used in the lecture 11 code, which contains the following line of VHDL code.

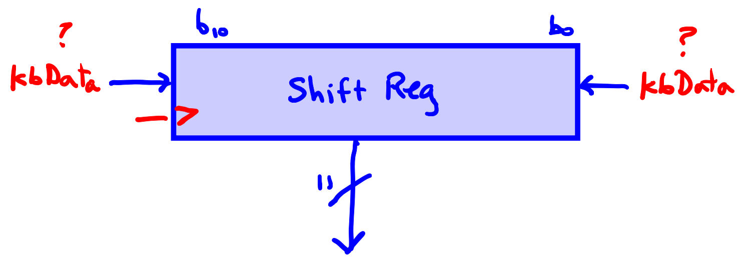
Is this shift left or shift right?

1 bit & 10-bits

```
shiftReg <= kbData & shiftReg (10 downto 1);
```

BBB

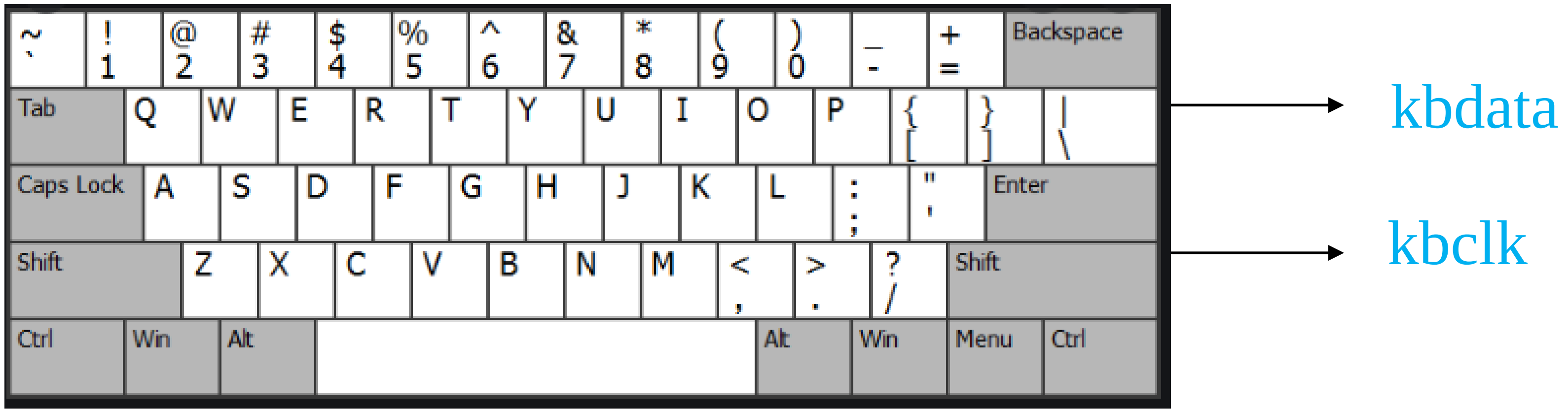




How would you shift left?

shiftReg <=

How do you make a Serial to Parallel Converter?



Keyboard Serial to Parallel Converter

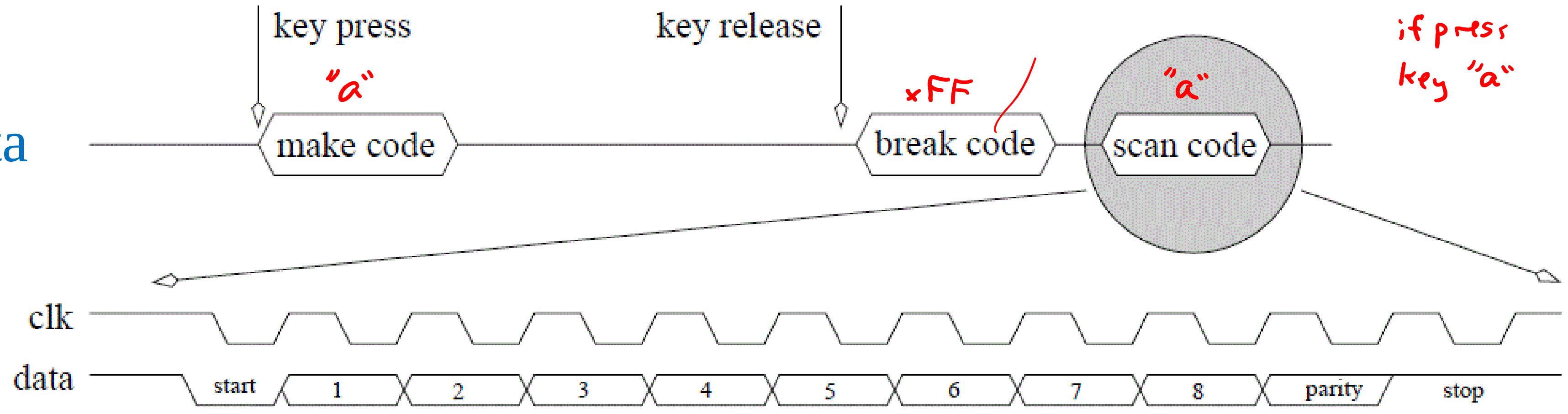
UNITED STATES AIR FORCE ACADEMY

Keyboard Serial to Parallel Converter

Nomenclature:	PS/2 Keyboard
Data Input:	none
Data Output:	1-bit data, nominally logic 1
Control:	none
Status:	none
Others:	1-bit clk, nominally logic 1
Physical Input:	key press and key release events
Physical Output:	none
Behavior:	When a key is pressed, its 8-bit make code is transmitted. When a key is released, an 8-bit break code is transmitted, immediately followed by the key's 8-bit scan code.

ready: ___

data



Example: how do you type "A"?

"shift"

"a"

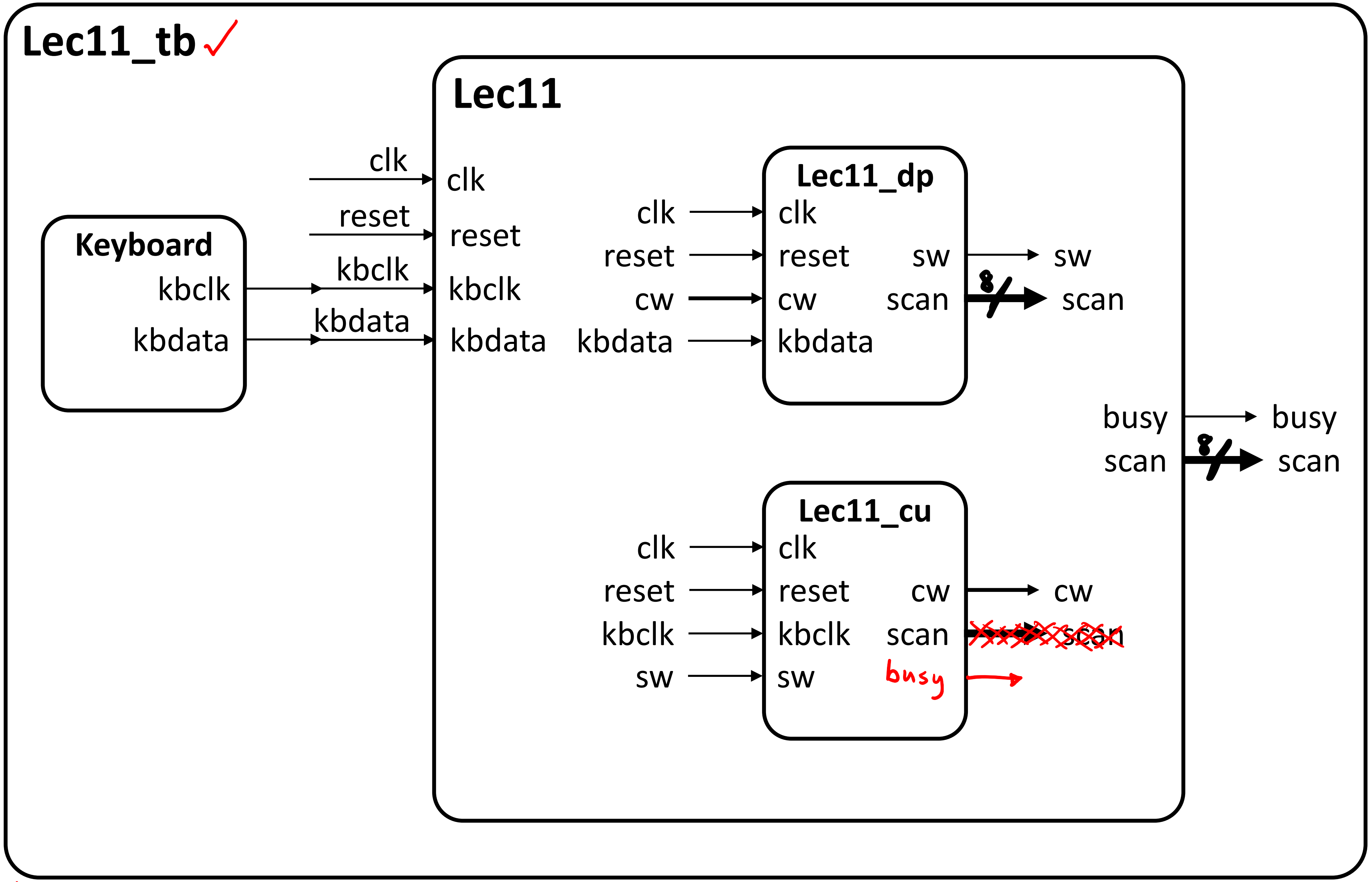
When we press one key, how many bits come out? ________

… and what do we want to decode (for HW08)? ______________

For decoding, do we want to save a bit on the rising or falling edge?

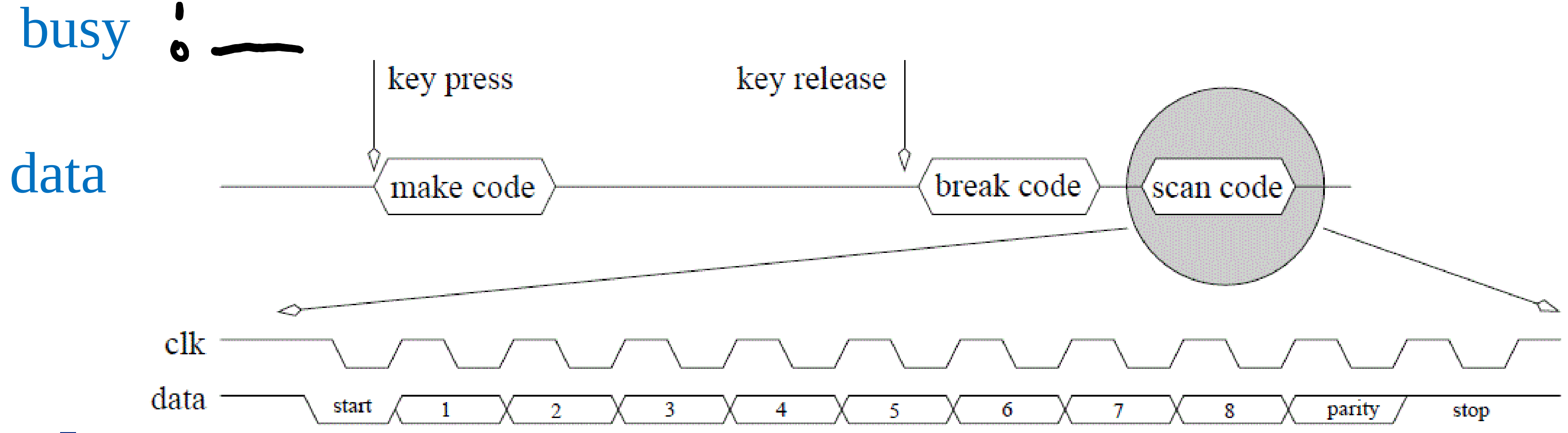
Keyboard Serial to Parallel Converter

Nomenclature:	Keyboard scan code reader
Data Input:	1-bit kd data, nominally logic 1 1-bit kd clk, nominally logic 1
Data Output:	8-bit scan code
Control:	none
Status:	1-bit busy, nominally logic 0
Others:	1-bit clk, nominally logic 1
Behavior:	Interprets the PS/2 keyboard clk and data signal from a keypress event and outputs the associated scan code. The busy signal goes high when the first data bit arrives and stays high until the last data bit is received. Busy is low only when there is a valid scan code on the output.



✓ Given: testbench, keyboard, lec11, and datapath VHDL code, and CU mini-C

HW08: You design control FSM CU and create block diagram of DP (BBBs!)

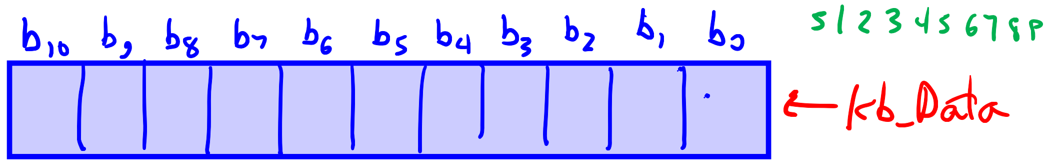


mini-C algorithm to decode:

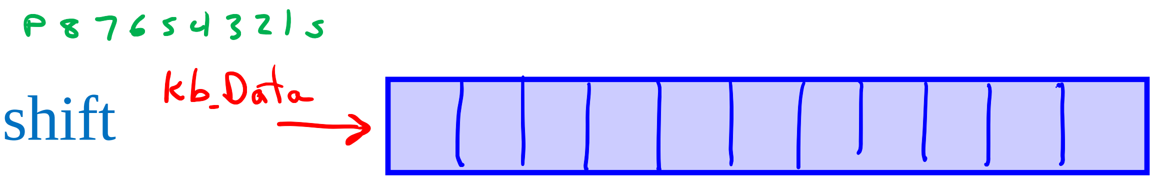
```
while(1) {
    busy=0;
    while (kb_clk == 1);
    busy=1;
    for (count=0; count<33; count++) {
        while(kb_clk == 1);
        shift = (shift << 1) | kb_data; // oops, should be right not left shift? Why?
        while(kb_clk == 0);
    } // end for
    scan = shift[9..2];
} // end while 1
```

Homework #8

If we do left shift



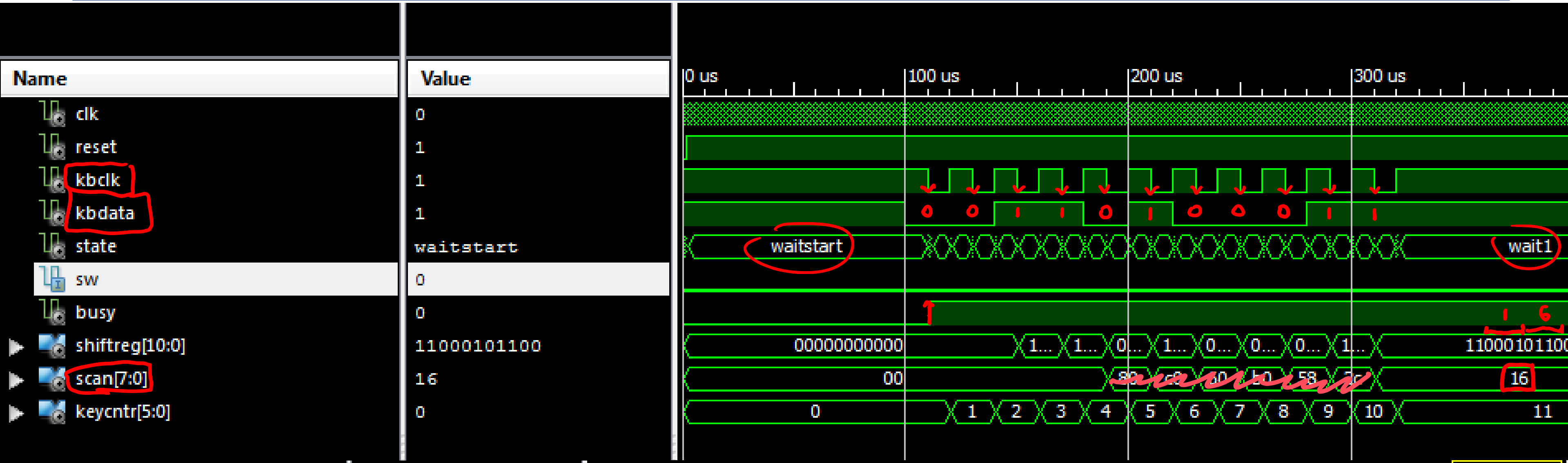
If we do right shift



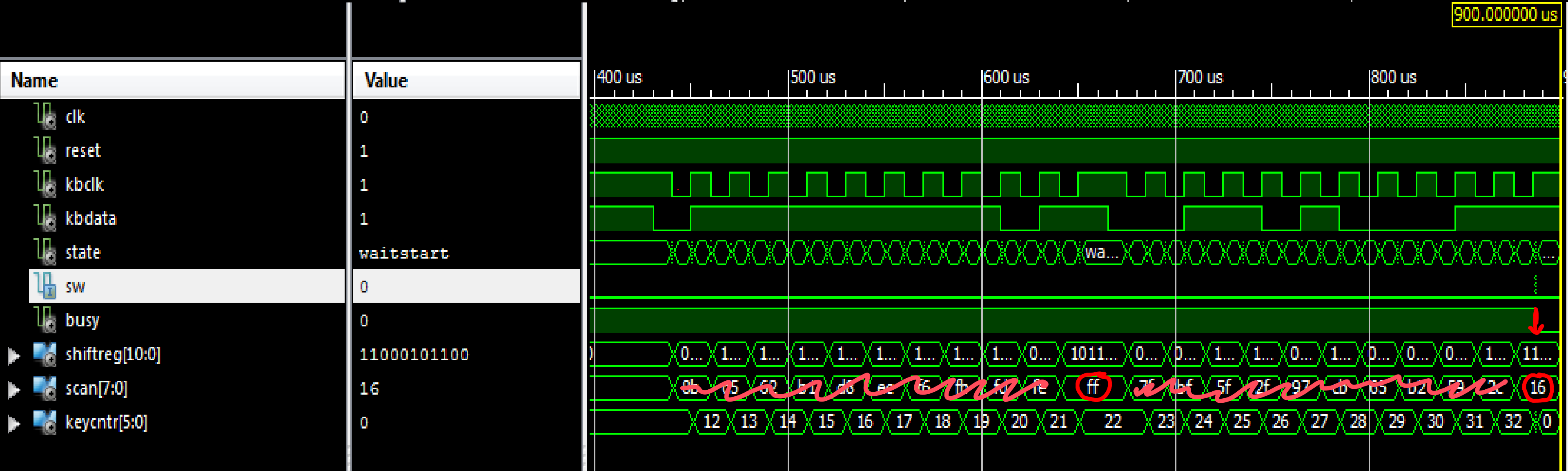
- Now lets build the datapath and control using the technique learned in lecture 10.
- Your homework is to build the control unit for the keyboard scancode reader.

Homework #8

✩ see given code



Name	Value
clk	0
reset	1
kbclk	1
kbdata	1
state	waitstart
sw	0
busy	0
shiftreg[10:0]	11000101100
scan[7:0]	16
keycntr[5:0]	0



Name	Value
clk	0
reset	1
kbclk	1
kbdata	1
state	waitstart
sw	0
busy	0
shiftreg[10:0]	11000101100
scan[7:0]	16
keycntr[5:0]	0