DigitLabwork#7
Fixed-table decoder and application

= =

[1] Training goals

Through the lab-work, class members are expected to get acquainted with the following matters.

a)) Issues concerning a module using fixed-table codes for receiving 3D coordinates (x, y, z) of a sets of locations and performing distance computation between successive points required by the application

** fixed-table decoding by FSM,

** geometrical 3D distance computation;

b)) the use of Verilog for logic circuit description and simulation, including

** circuit module building up in Verilog,

** setting up of the test data;

c)) the interactive commands required in operating the Verilog-code development system:

** compilation for a syntax error-free Verilog description and test data set;

** simulation of the Verilog-coded circuit module;

[** synthesis of the Verilog-coded circuit module].

[2] Preparation

Every class member should get prepared with the following knowledge prior to attending the lab-work sessions.

a)) how the task could be properly partitioned so that functionality of each subtask could be fulfilled by a circuit module of reasonable complexity:

** state-transition of the main module? Or other control mechanism over the data flow processing?

** state-transition diagram for decoding fixed-table codes?

** data preparing in a way that would facilitate both receiving process and computing at the same time, whenever situations allowing?

** receiving inputs and generating outputs as required?

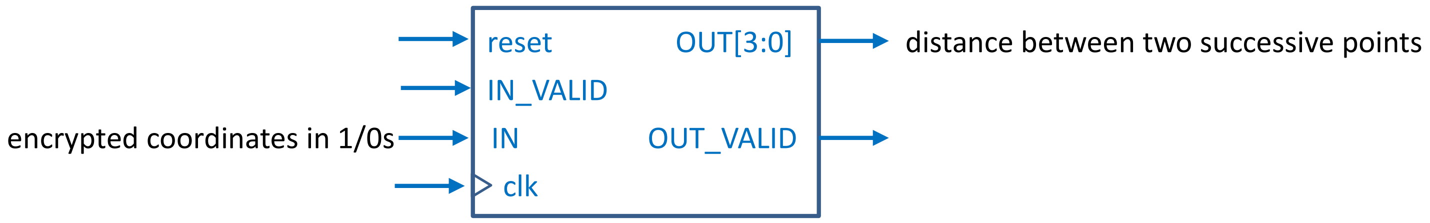
b)) skills of generating required outputs during state transitions in the operation of a sequential circuit.

[3] Lab-task instruction

In digitlabwork#5, 1 task is assigned to every class members as given below.

[TASK1]

(1) Write a Verilog code for a module capable of reading encrypted coordinates of a set of 3D points for distance computation in compliance with the requirements given below.


reset
IN_VALID
encrypted coordinates in 1/0s
IN
clk
OUT[3:0]
distance between two successive points
OUT_VALID

(2) Put the codes under simulation; observe and interpret the waveforms of output signals

(3) Try drawing the state transition diagram (or FSM) for the fixed-table decoder (FTD hereafter) of your own design,

(4) Try drawing the circuit diagram of your design using counters, logic gates and components one deems as necessary.

a)) circuit operational specifications

** All signals are high-active

** All inputs arrive at the falling edge of the clock

** All outputs appear at the rising edge of the clock

** I/O layout:	inputs	reset, clk, IN_VALID, IN;
	outputs	OUT_VALID, OUT[3:0];

** signal implications:

signal	implications
reset	** initiation of the system at clock-synchronization; ** appears only once for every power-up operating cycle
IN_VALID	** indicating the arrival of a leading bit, followed by a sequence of encoded coordinates at IN port ; ** remaining active as long as the incoming of encoded coordinates continues; ** rising and falling at the negative edge of the clock
IN	** consisting of a leading bit and a series of 0 and 1 representing encoded (X,Y,Z) coordinates of a set of 3D points, leading bit: 0 code-table0 is to apply to the decoding of the subsequent encrypted coordinates, 1 code-table1 is to apply; ** 1 bit per clock input rate; ** valid only when IN_VALID is active, 0 otherwise; ** input data interpretations: leading-bit + 1/0 bit-sequence (as long as IN_VALID remaining active) 1/0 sequence ⇔ 1/0 frames (1 frame ≡ X, Y or Z, to be decoded by FTD codebook)

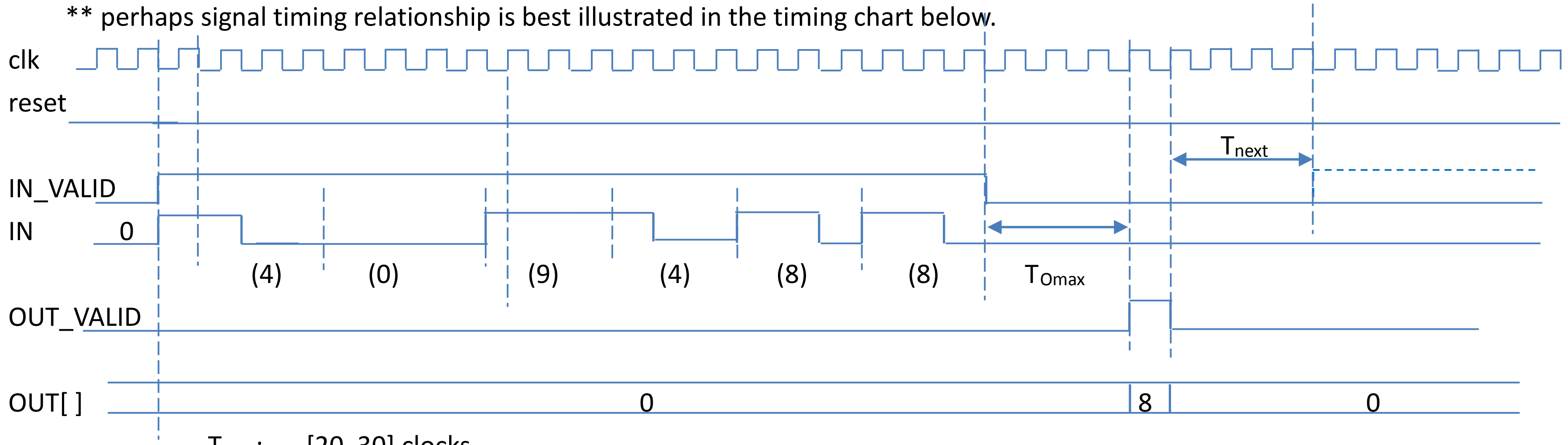
	* size of frames varies * 3N frames in total for each input cycle (N: number of points entered in the cycle) * every 3 frames, starting from the 1st, are interpreted as (X,Y,Z) coordinate of a particular point <table><tr><td>frame^{X1}</td><td>frame^{Y1}</td><td>frame^{Z1}</td><td>. . .</td><td>frame^{Xj}</td><td>frame^{Yj}</td><td>frame^{Zj}</td><td>. . .</td><td>frame^{XN}</td><td>frame^{YN}</td><td>frame^{ZN}</td></tr></table>* value of X, Y, Z ranging from 0 to 9 (i.e., single digit decimal only)
clk	** all inputs arriving at falling edge of the clock; ** the system samples inputs at rising of the clock and reacts accordingly;
OUT_VALID	** indicating the availability of data at OUT-port, i.e., the module starts pouring out a sequence of 4-bit values, each being the distance between two successive points in the N-point input sequence; ** remaining active as long as the distance-outpouring continues
OUT[3:0]	** remaining 0 after reset and whenever OUT_VALID is low;

** coding tables

	Code-table0	Code-table1
decimals	code sequence	code sequence
0	1	0000
1	01	0001
2	000	001
3	00100	01
4	00101	100
5	00110	10100
6	0011100	10101

7	00111010	1011
8	00111011	110
9	001111	111

** perhaps signal timing relationship is best illustrated in the timing chart below.



T_{next}: [20, 30] clocks

T_{Omax}: less than 20 clocks

b)) coding specifications:

** write behavior-level descriptions for the module of your own design fulfilling the requirement given below.

** the module has reset, clk, and IN_VALID as inputs, and OUT_VALID, OUT[3:0] as outputs.

** all input receiving and output generation undergo at the rising edge of the clock.

Event	Reactions
at receiving reset (reset (clk^))	entering initial state: 1) OUT_VALID: 0 2) OUT[]: 0 3) initiation for internal use
at receiving IN_VALID (IN_VALID(clk^))	** receiving the leading bit, determining the code-table to be used ** receiving subsequent 1/0 bit sequence until the deactivation of IN_VALID
generating OUT_VALID (OUT_VALID(clk^))	** becoming active at the rising edge of the clock. ** the raising of OUT_VALID should not be later than 20 clocks after the falling of IN_VALID. ** remaining active for N-1 clocks in the case of receiving N-point coordinates.
generating OUT[ ] (OUT[ ](clk^))	** sending out $DIST_k$ (k=1~N-1), 1 unit per clock, where $DIST_k = ((X_{k+1}-X_k)^2 + (Y_{k+1}-Y_k)^2 + (Z_{k+1}-Z_k)^2)^{0.5}$.
readiness for the next round input sequence	** after the falling of OUT_VALID, the module should enter the state the same as after being reset. ** after the falling of OUT_VALID, next round input sequence may arrive within 20 to 30 clocks.

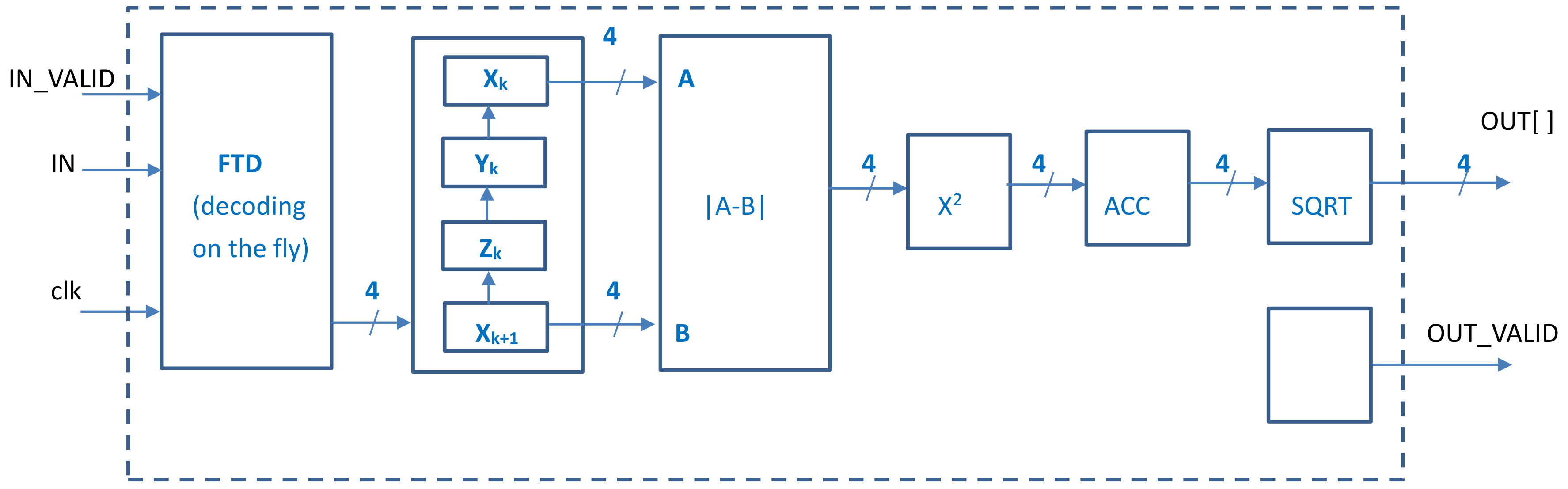
** Timing: no delay(s) to be considered in this task

c)) testing data:

to be supplied by TAs.

[4] some tips

a)) plausible system structure



[notes] for each module * how’s the function to be done?

* when to start the operation?
* how much time to take in the operation (in terms of clock number)?
* when to end the operation?
* any flags ON/OFF handling?

. . .

b)) plausible mechanism in FTD for handling variable-length codes

** would the use of mealy or moore machine make a difference?

[4] Observations

** If the task is carried out by your own design, compare your design to the one coded by TAs in the following aspects.

1)) Is the data processing flow governed by a state machine?

* If yes, which of the two state machines is more compact?

* If mechanism other than state-transition is exploited in your work, does the control simpler than the state-machine approach?

2)) As the length of the input bit sequence is not known till the input stops and the computation could get started as soon as coordinates of two consecutive points are available, so computation on the fly would be inevitable. Compare how it is done in your work and TA's, and tell which one you prefer.

3)) Any approaches to decoding fixed-table codes of variable length other than state-machine?