FPGA Embedded System Lab

FPGA Muse Dash

DE2-115 Team D1

Chen Zhiyi 21307130003 Wang Tianchen 21307130031

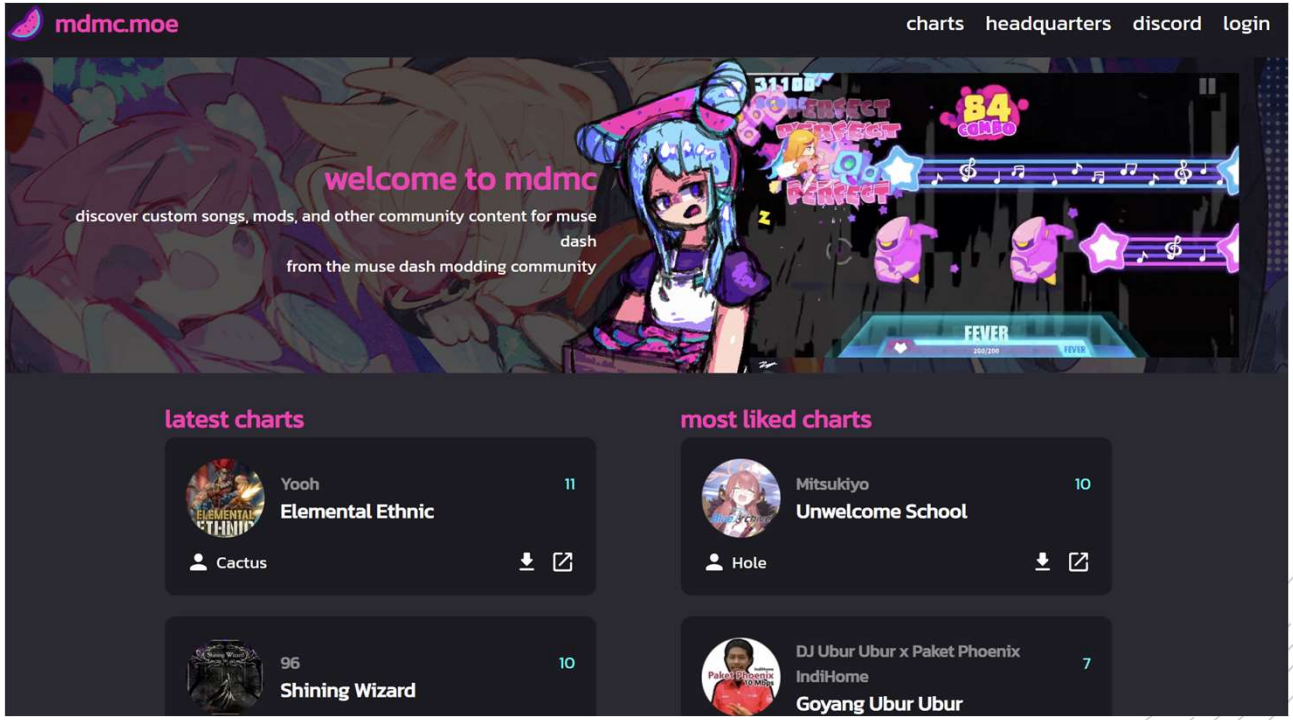
Dong Yuxuan 21307130081 Chen Yikuan 21307140002

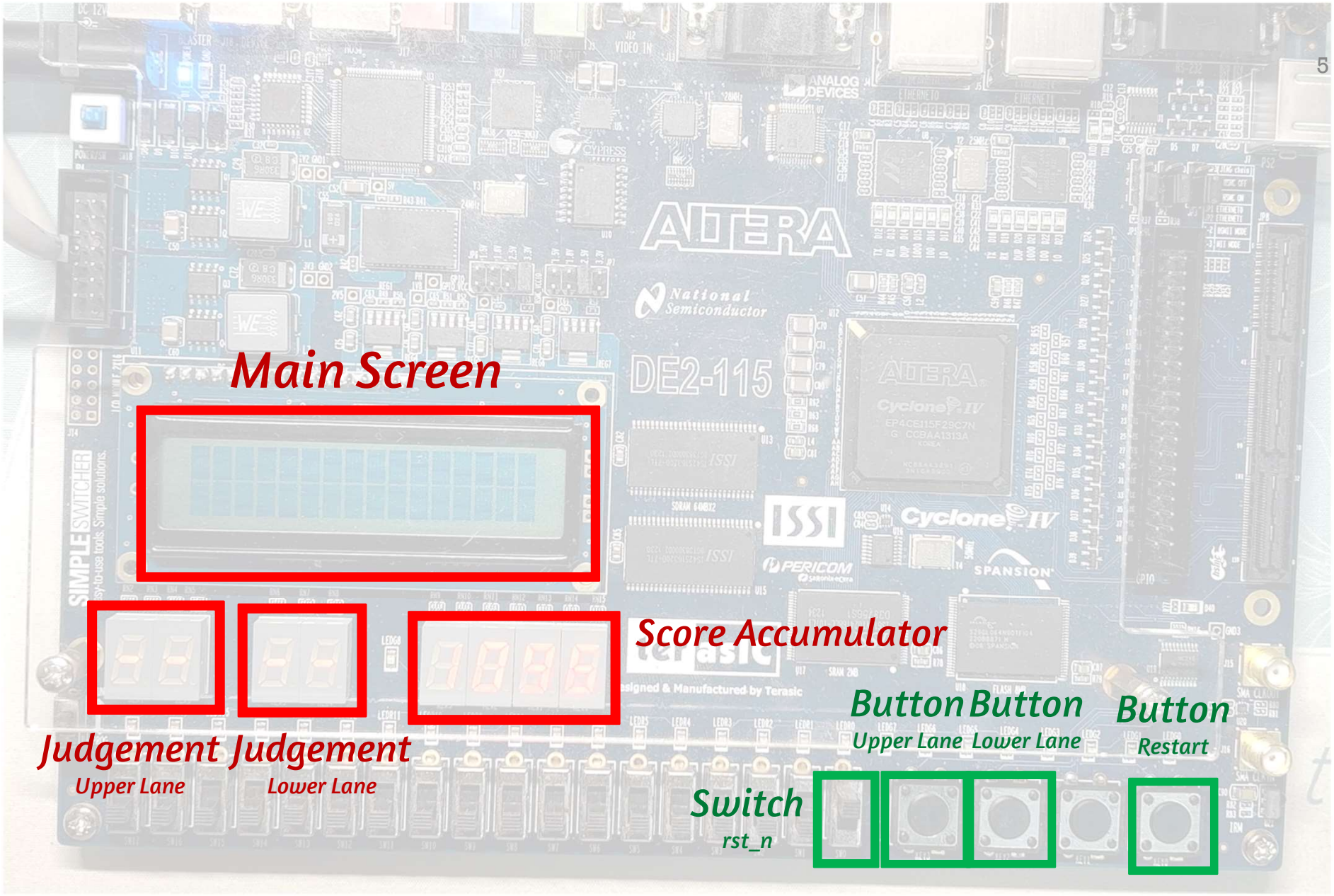
Contents

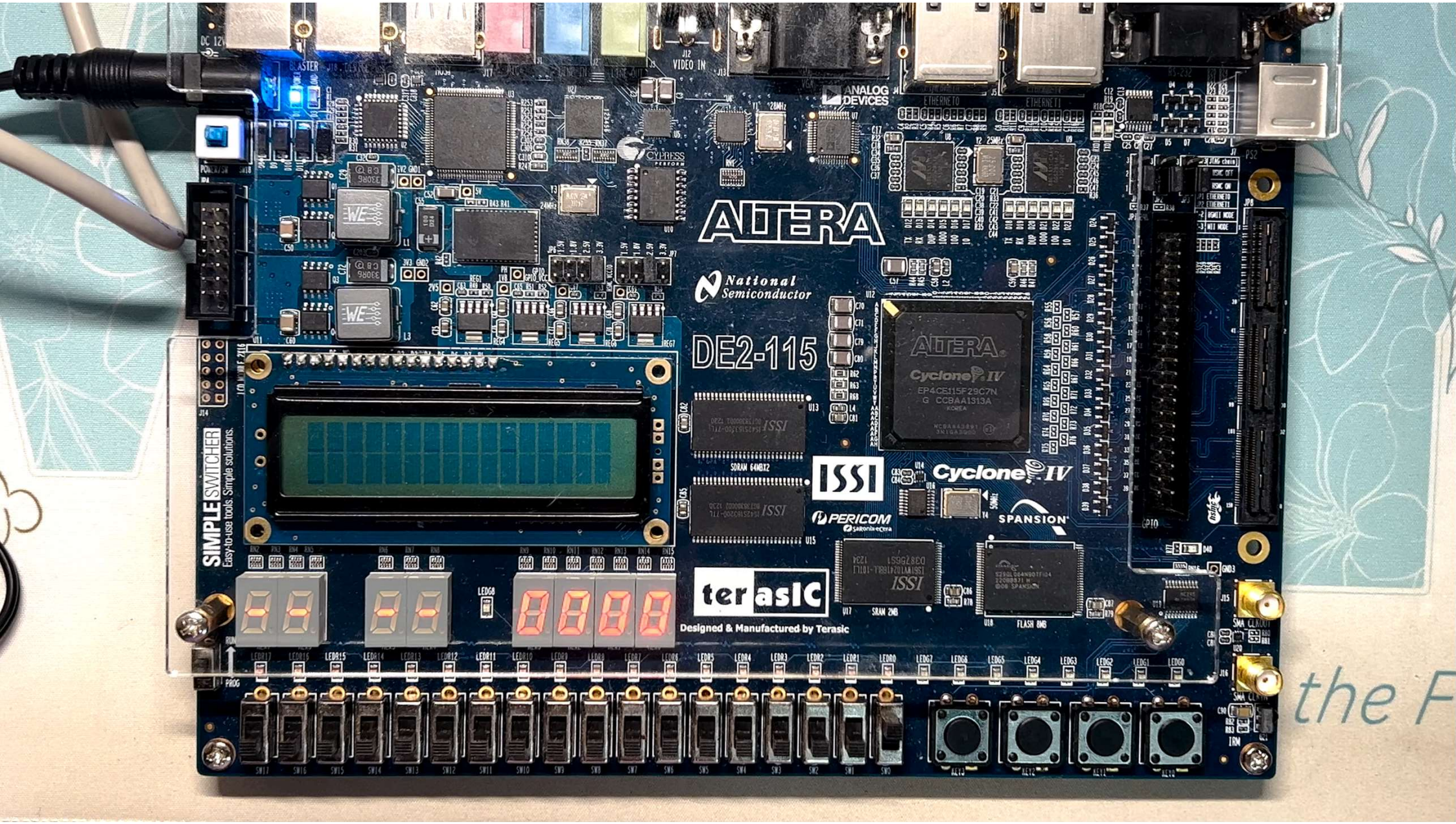
Introduction

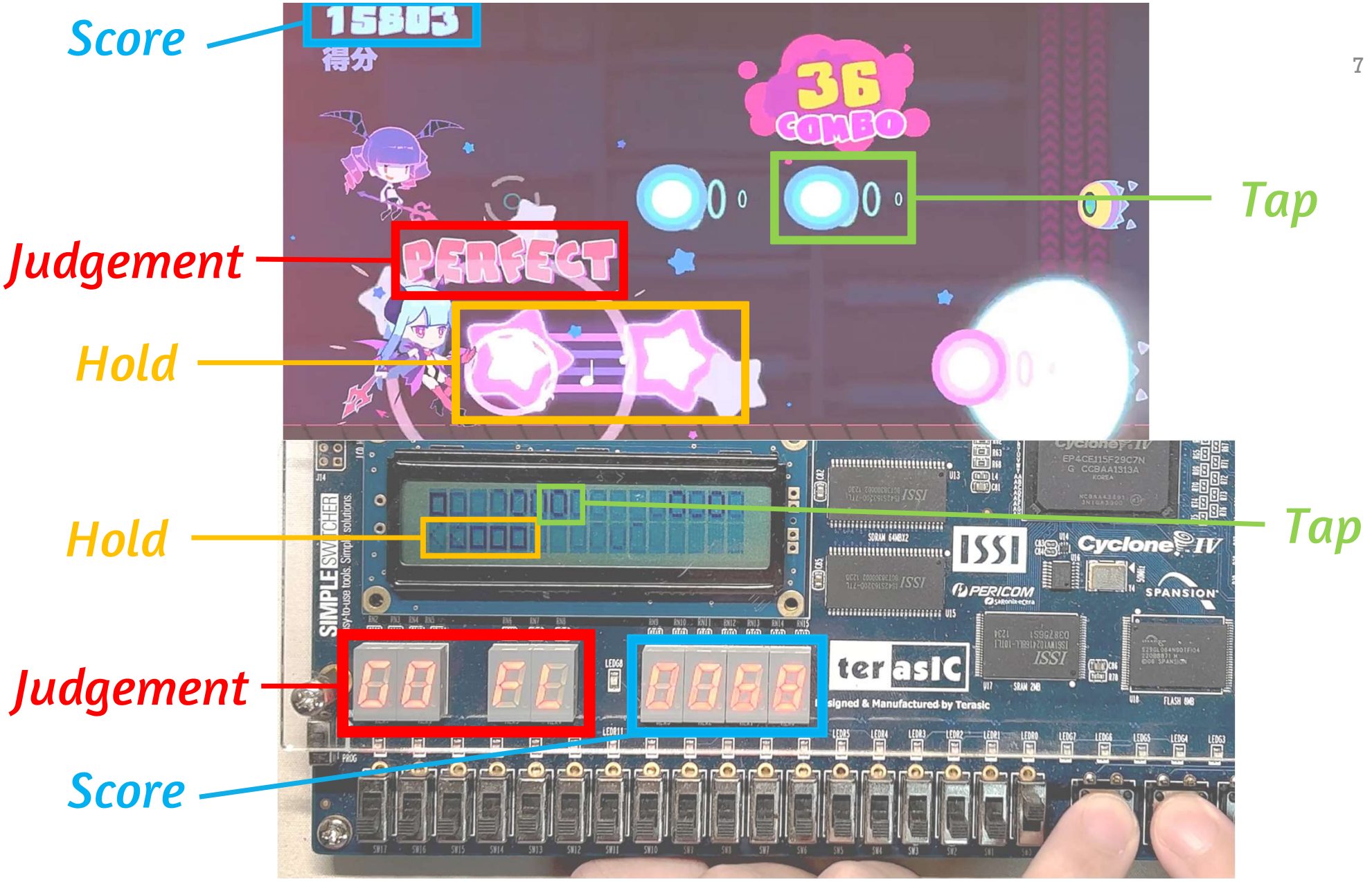
Idea

- Hardware: Muse Dash
- Software: Fanmade charts




Main Screen
Score Accumulator
Judgement
Upper Lane
Judgement
Lower Lane
Button
Upper Lane
Button
Lower Lane
Button
Restart
Switch
rst_n


ALTERA
DE2-115
National Semiconductor
ANALOG DEVICES
CYPRESS
ISSI
PERICOM
SPANSION
Cyclone IV
EP4CE115F29C7N
terasIC
Designed & Manufactured by Terasic
SIMPLE SWITCHER
Easy-to-use tools. Simple solutions.
VIDEO IN
ETHERNET0
ETHERNET1
SRAM 64MB32
SRAM 2MB
FLASH 8MB


Score
Judgement
Hold
Tap
15803
得分
36
COMBO
PERFECT
Hold
Tap
Judgement
Score
SIMPLE SWITCHER
ISSI
Cyclone IV
PERICOM
SPANSION
terasIC
Designed & Manufactured by Terasic

Design

Software

- Input Chart Format
  - Standardized Chart Format
  - Note Type
  - Example
- Chart to ROM
  - Procedure
  - Output Function

Standardized Chart Format

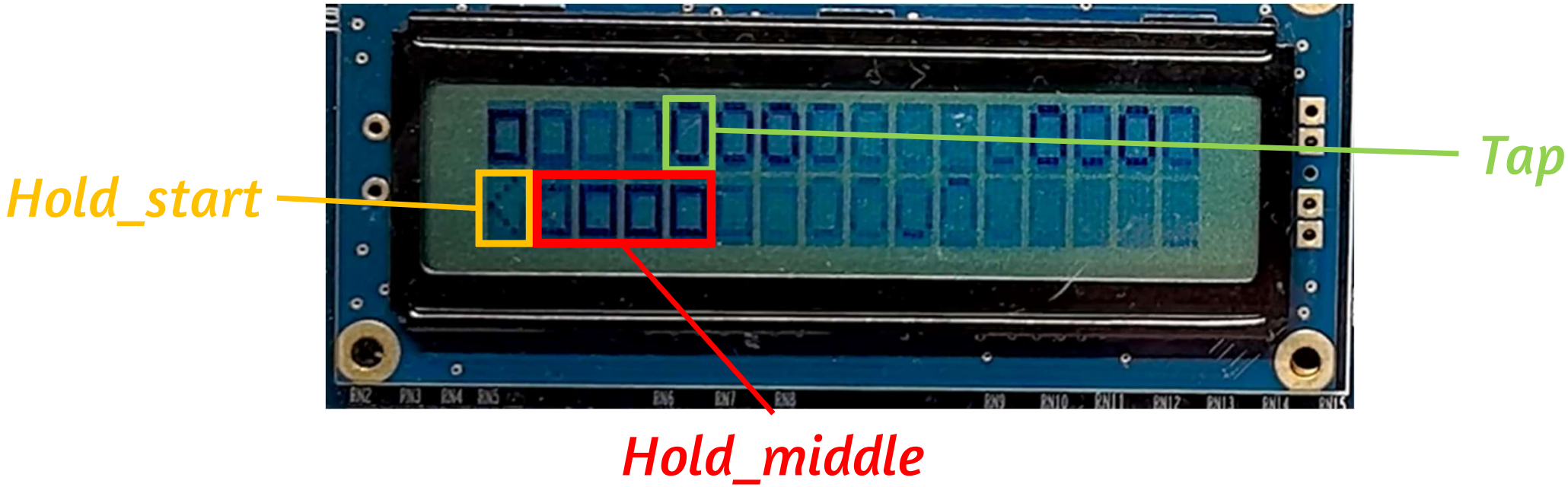
Music BPM + Note Description

Note = (time, type, track)

- Note time: when the note will be judged
- Note type: how the note will be judged
- Note track: where the note will be

Note Type

- TAP note (O): Click
- HOLD_START note (<): Click and hold
- HOLD_MIDDLE note (□): Keep holding



Example

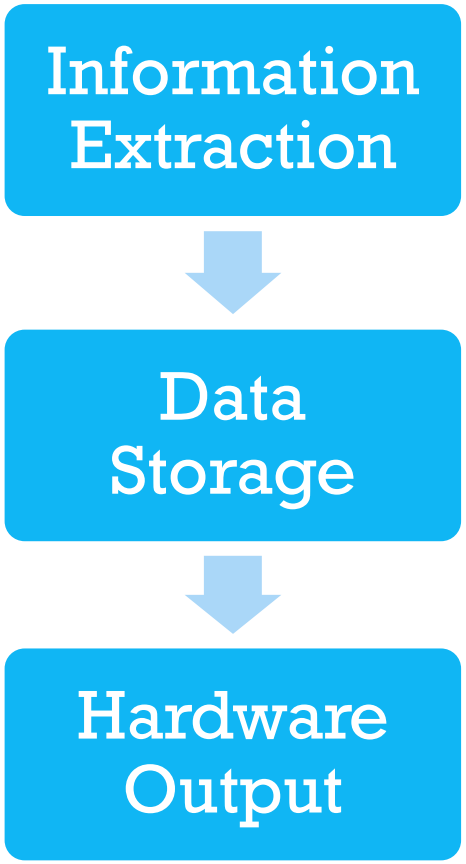

```
bpm=150
(16,tap,0)
(18,tap,1)
(32,hold_start,0)
(33,hold_mid,0)
(34,hold_mid,0)
(35,hold_mid,0)
```

Note type	Code
NO_NOTE	2'b0
TAP	2'b1
HOLD_START	2'b2
HOLD_MIDDLE	2'b3

Track type	Code
Lower Track	1'b0
Upper Track	1'b1

Procedure

Standardized Chart → Configured Hardware



- Int “bpm”
- Class “data”
- Intermediate vector “rom”
- Top module modification
- ROM generation

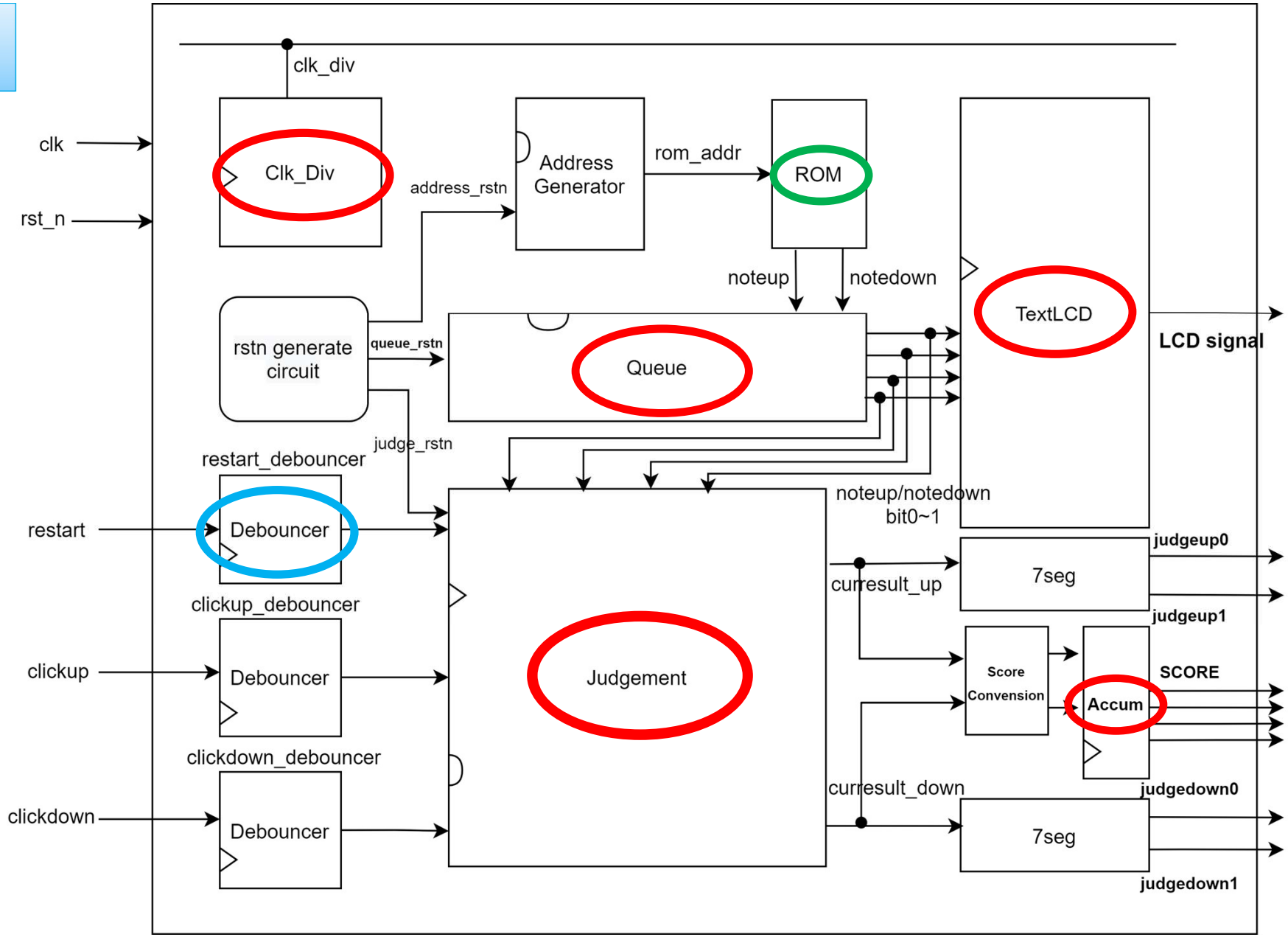
Output Function


```
52  void outputROM(const vector<vector<int>>& rom, const string& outputFilename) {
53      ofstream outfile(outputFilename);
54      outfile << "module ROM (\n";
55      outfile << "    input [11:0] addr,\n";
56      outfile << "    output reg [1:0] noteup,\n";
57      outfile << "    output reg [1:0] notedown\n";
58      outfile << ");\n\n";
59
60      outfile << "reg [3:0] ROM [0:4095];\n\n";
61      outfile << "initial begin\n";
62      for (int i = 0; i < 4096; ++i) {
63          outfile << "\tROM[" << i << "] = 4'b" << rom[i][3] << rom[i][2] << rom[i][1] << rom[i][0] << ";" << endl;
64      }
65      outfile << "end\n\n";
66      outfile << "always @(*) begin\n";
67      outfile << "    {noteup, notedown} = ROM[addr];\n";
68      outfile << "end\n\n";
69      outfile << "endmodule\n";
70  }
```

Hardware

- Top Module
- Adjustable Clk_div
- Queue
- TextLCD
- Judgement FSM
- BCD Accumulator

Top


clk_div
clk
rst_n
Clk_Div
address_rstn
Address Generator
rom_addr
ROM
noteup
notedown
TextLCD
LCD signal
rstn generate circuit
queue_rstn
Queue
judge_rstn
restart_debouncer
restart
Debouncer
noteup/notedown bit0~1
judgeup0
7seg
judgeup1
curresult_up
clickup_debouncer
clickup
Debouncer
Judgement
Score Convension
Accum
SCORE
clickdown_debouncer
clickdown
Debouncer
curresult_down
judgedown0
7seg
judgedown1

Clk_div

```
module MuseDash #(
    parameter div_cnt = 1760563
    // div_cnt = 50,000,000 / (bpm * 4 / 60) / 2 = 375,000,000 / bpm;
) (
```


Song BPM Parameterization

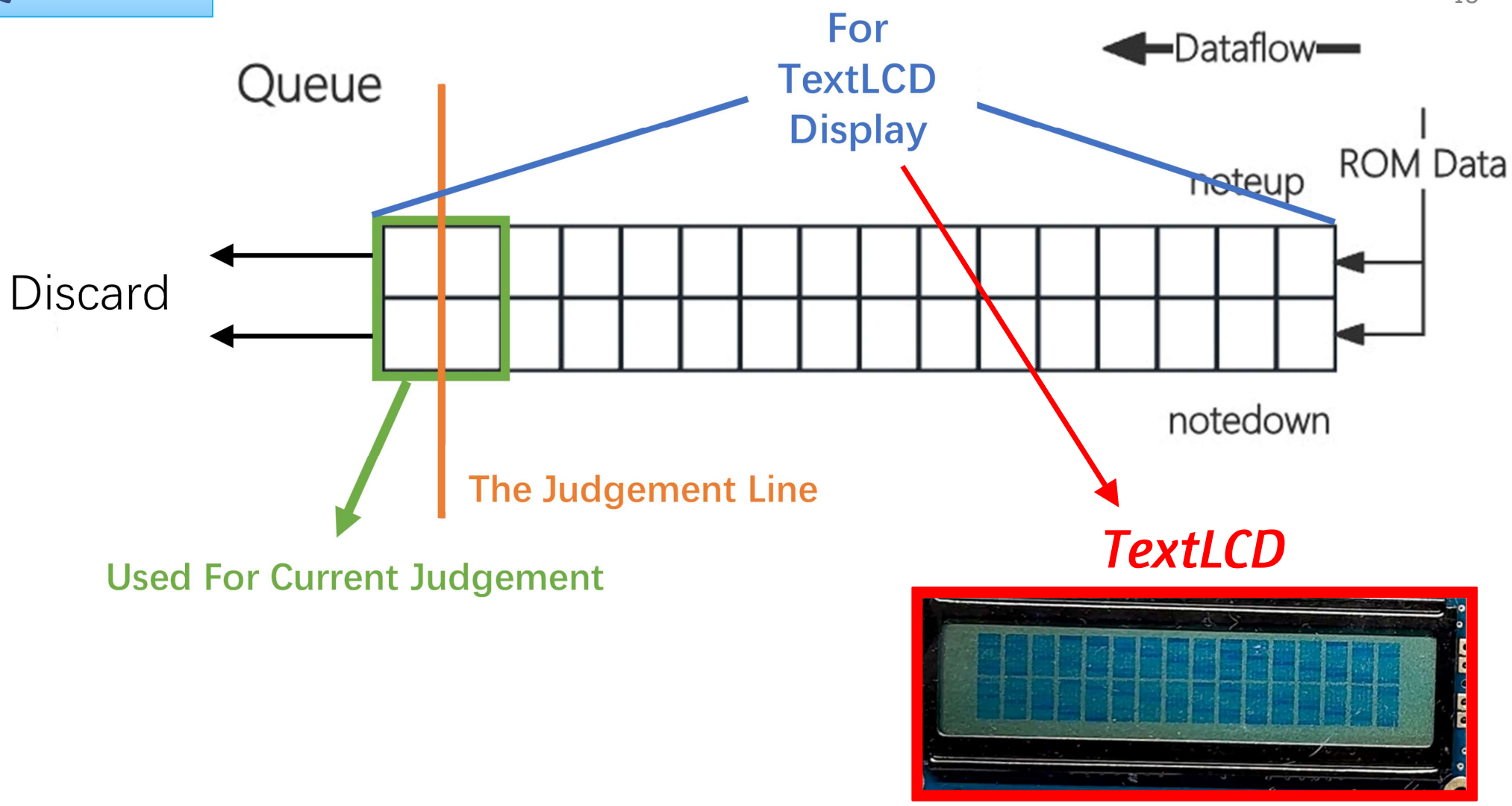
```
Clk_Div #(
    .div_cnt(div_cnt)
) clock_divider (
    .clk (clk),
    .rst_n (rst_n),

    .clk_div (clk_div)
);
```

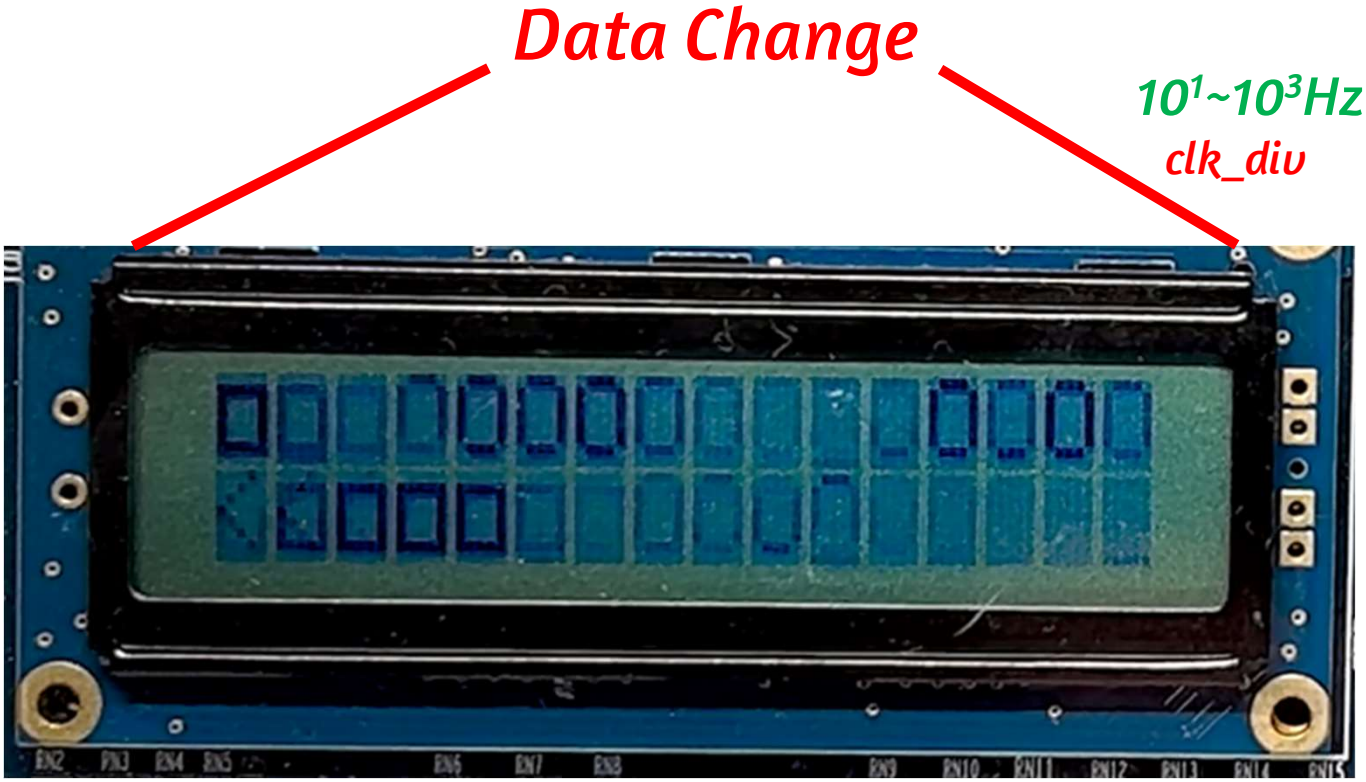
```
module Clk_Div #(
    parameter div_cnt = 2500000 // 50MHz to 10Hz (100ms)
    // div_cnt = 50,000,000 / (bpm * 4 / 60) / 2 = 375,000,000 / bpm;
)(
    input clk,
    input rst_n,

    output reg clk_div
);
```

Queue

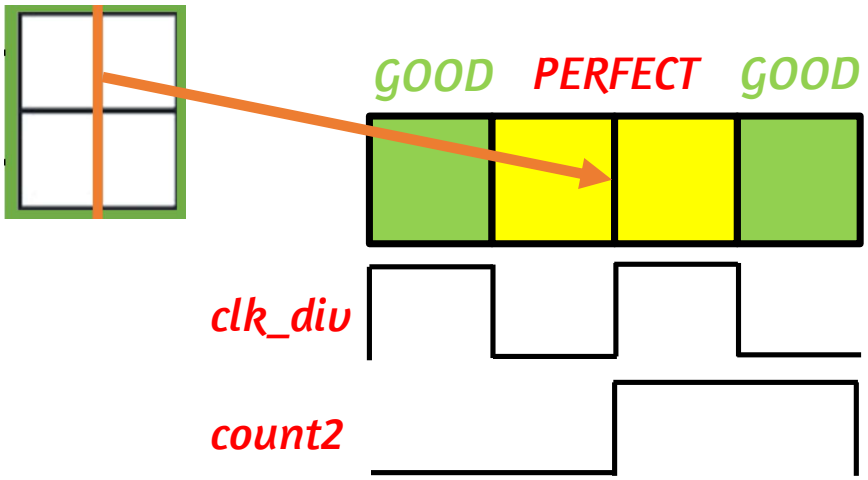

Queue
For
TextLCD
Display
Dataflow
ROM Data
noteup
Discard
notedown
The Judgement Line
Used For Current Judgement
TextLCD

TextLCD



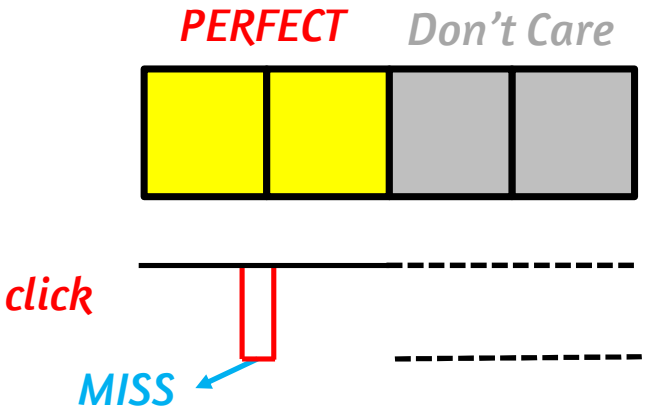
Judgement FSM

TAP/HOLD_START Judgement



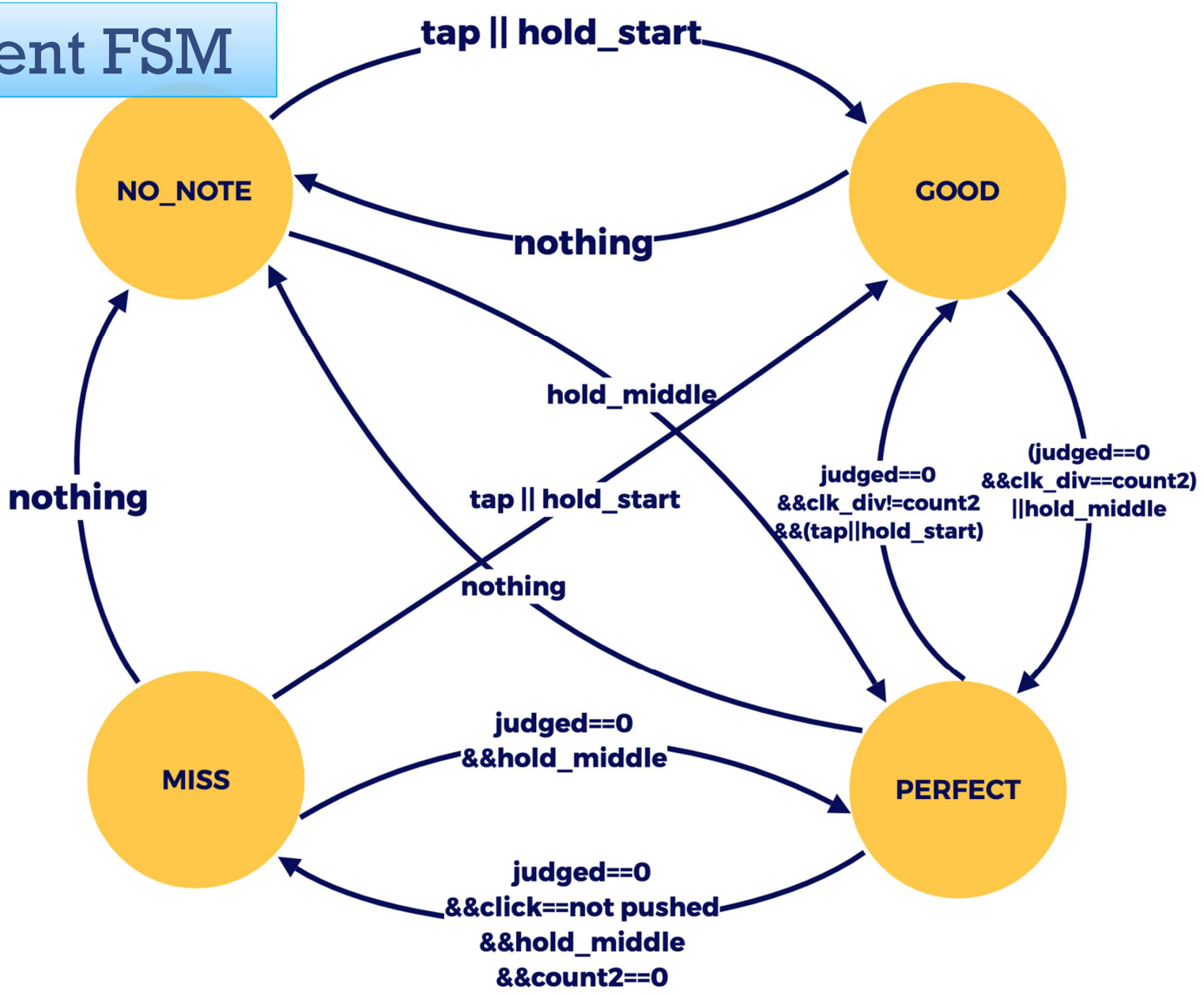
Monitoring Click Edge

HOLD_MIDDLE Judgement

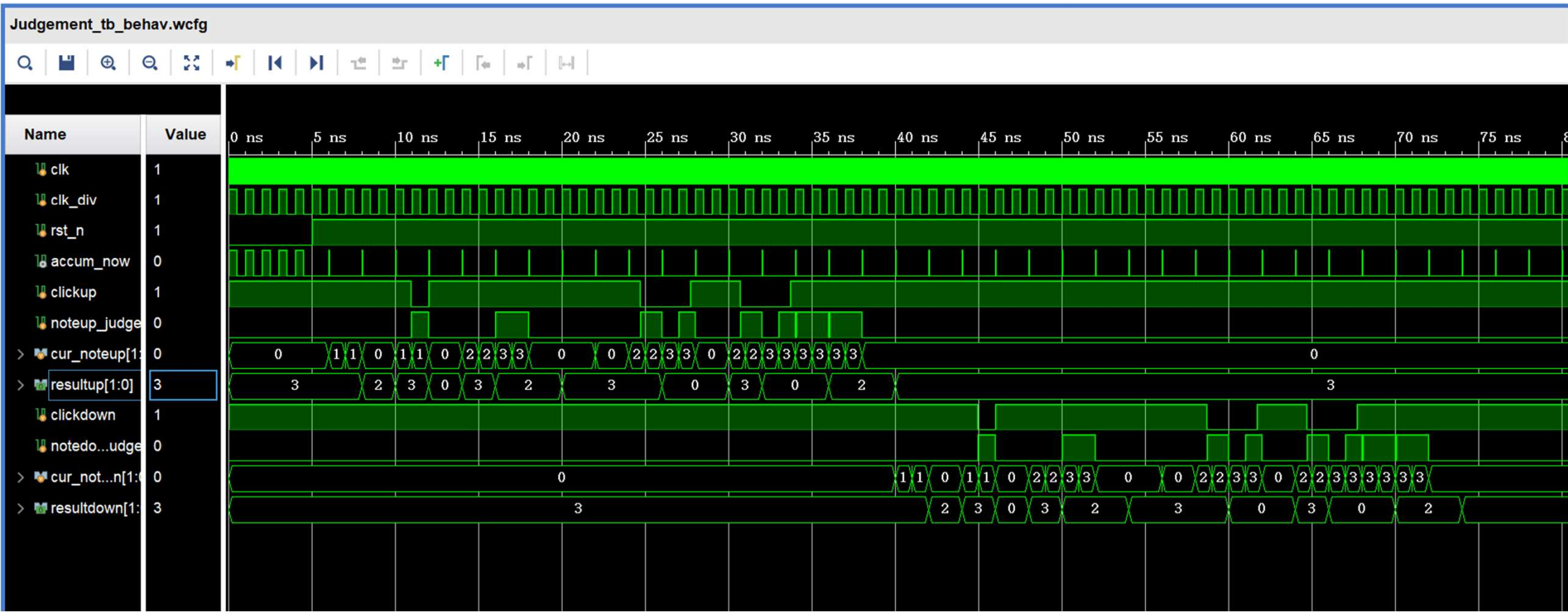


Monitoring Not-Clicked Moment

Judgement FSM



Judgement FSM



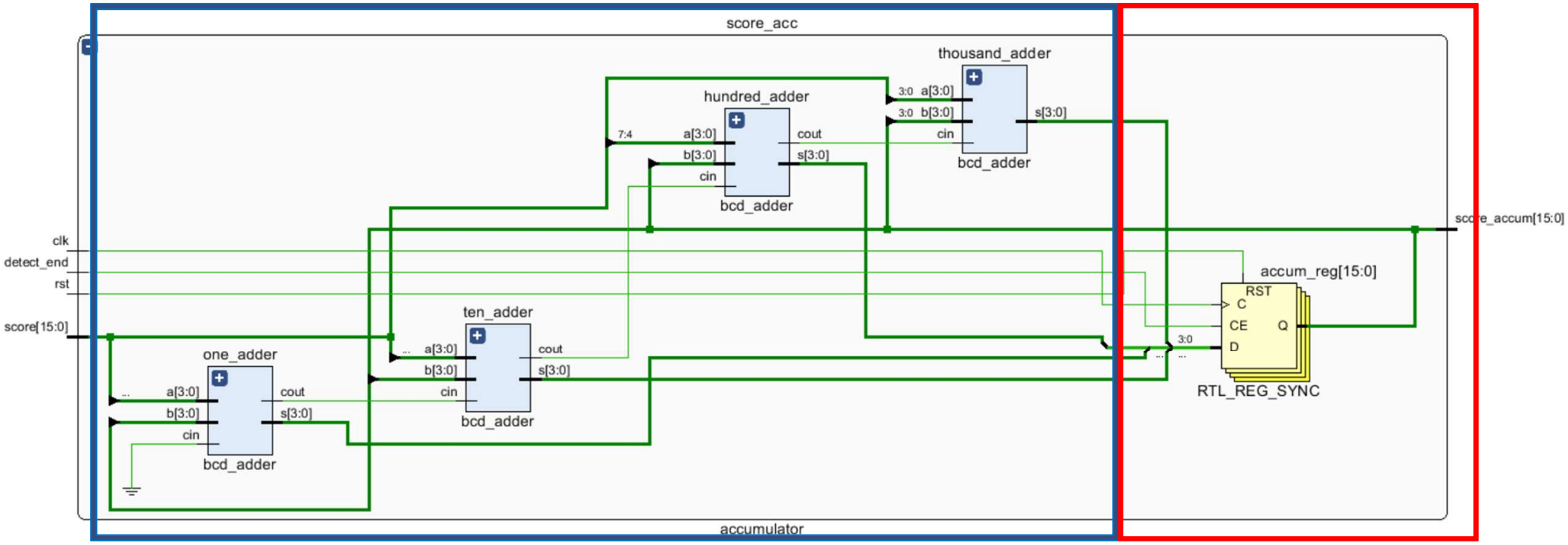
Score Accumulator

4-digit BCD Adder

Cascaded 1-digit BCD Adders

Accumulation Controller

Accumulate when ready



BCD Adder

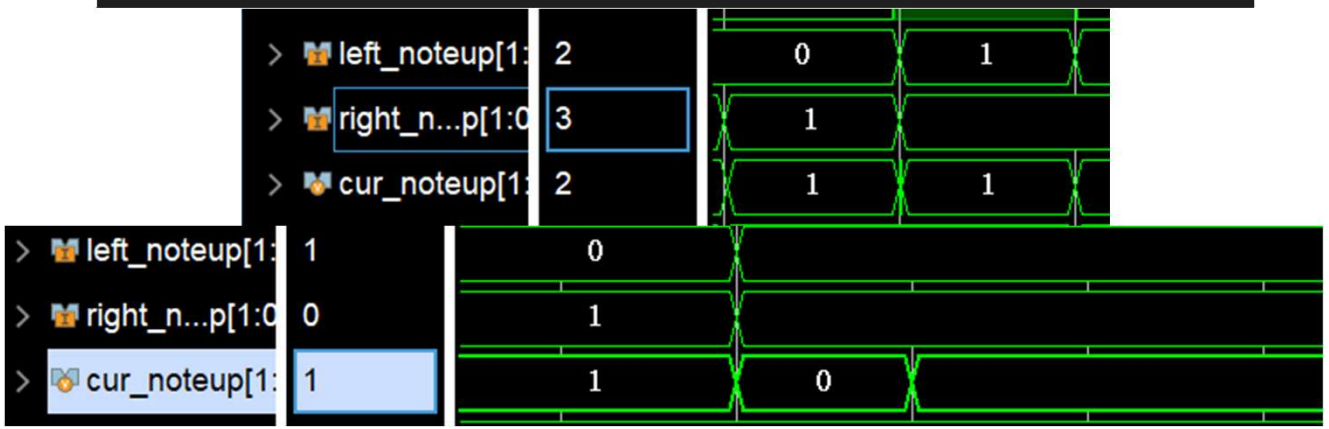

```
20   wire    [4:0] temp;
21   wire [4:0] temp_out;
22
23   assign temp             = a_extend + b_extend + cin_extend;
24   assign temp_out         = (temp > 5'd9) ? (temp + 5'd6) : temp;
25   assign cout             = temp_out[4];
26   assign s                = temp_out[3:0];
27
28   endmodule
```

Main Algorithm

Discussion

Signal Penetration

```
// cur_noteup = count2 ? left_noteup : right_notup;
```




```
if(count2 == 1'b1 || (count2_negedge && !clk_div_posedge)) begin
    cur_noteup <= left_noteup;
    cur_notedown <= left_notedown;
end else if(clk_div_posedge) begin
    cur_noteup <= `NOTHING;
    cur_notedown <= `NOTHING;
end else begin
    cur_noteup <= right_noteup;
    cur_notedown <= right_notedown;
```

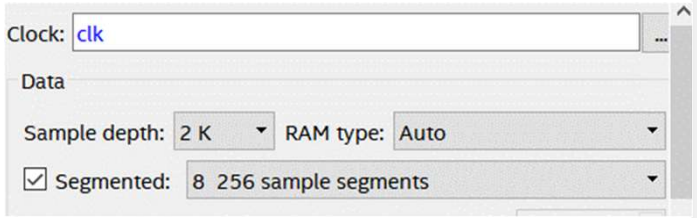
Solution

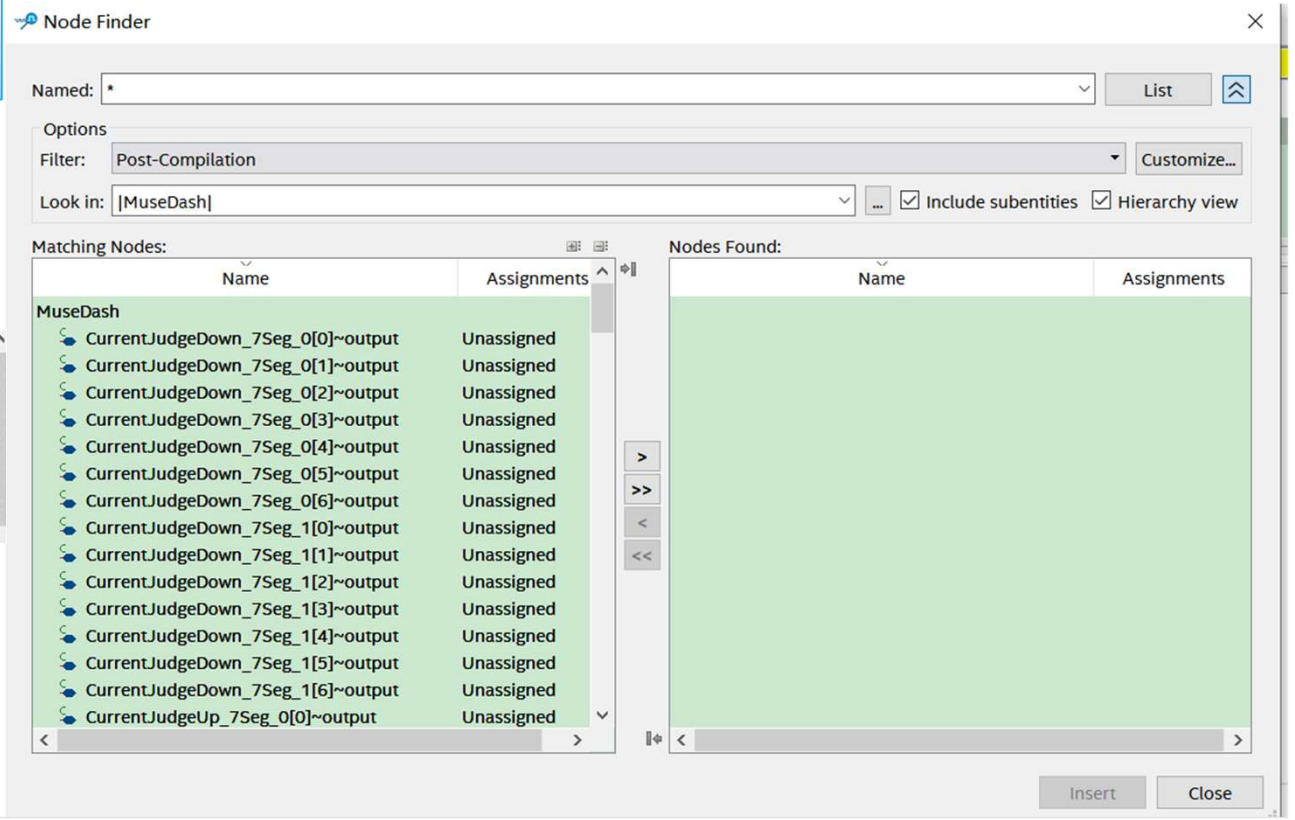
H-S Inconsistency

- Vivado Simulation ≠ On Board Verification
- Quartus Hardware Debugging?

```
if(cur_noteup == `NOTHING && cur_noteup_delay == `NOTHING && cur_noteup_delay2 == `NOTHING) begin
    next_state_up = `NO_NOTE;
```

Quartus Debugging

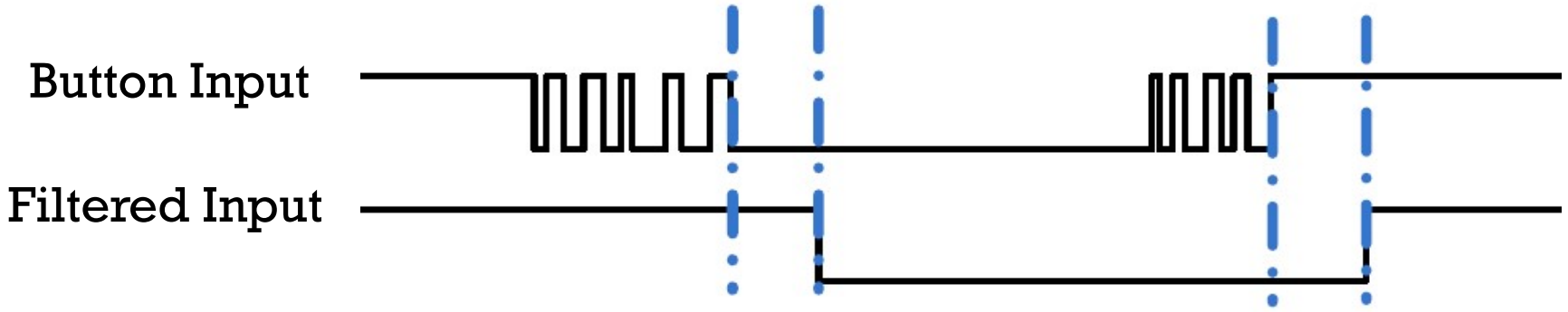




Future Works

- Support of keyboard and monitor IOs
- Support music playing
- Add music & chart selection interface
- Further expansion of embedded design
- Debouncement delay optimization

Debouncement delay



A fixed delay from debouncement!

Action	Timing range
Button jilter	5~10ms
'Manual' jilter	＞40~50ms
Filter standard	20ms

Judgement	Timing range
Critical Perfect	±25ms
Perfect	±50ms
Good	±120ms
Miss	＞120ms

Thank you