# Lab 3 COMPENG 3DQ5

Joshua Umansky, George Wahba

400234265, 400371904

McMaster University

COMPENG 3DQ5

# Lab 3 COMPENG 3DQ5

The code snippet outlines how we designed the display when pushing each of the letters between A to J. Each letter is assigned to a bit in the register letter, which is changed if PS2_code equals the corresponding value. A priority encoder was then used to output the MSK.

```
always_ff @ (posedge CLOCK_50_I or negedge resetn) begin
	if (resetn == 1'b0) begin
		PS2_code_ready_buf <= 1'b0;
		PS2_reg <= 8'd0;
		letter <= 9'd0;
	end else begin
		PS2_code_ready_buf <= PS2_code_ready;
		if (PS2_code_ready && ~PS2_code_ready_buf && PS2_make_code) begin
			// scan code detected
			PS2_reg <= PS2_code;
			if(PS2_code == 8'h29) letter <= 9'b0;
		end
		if(PS2_code == 8'h3B) letter[9] <= 1'b1; //J
		if(PS2_code == 8'h43) letter[8] <= 1'b1; //I
		if(PS2_code == 8'h33) letter[7] <= 1'b1; //H
		if(PS2_code == 8'h34) letter[6] <= 1'b1; //G
		if(PS2_code == 8'h2B) letter[5] <= 1'b1; //F
		if(PS2_code == 8'h24) letter[4] <= 1'b1; //E
		if(PS2_code == 8'h23) letter[3] <= 1'b1; //D
		if(PS2_code == 8'h21) letter[2] <= 1'b1; //C
		if(PS2_code == 8'h32) letter[1] <= 1'b1; //B
		if(PS2_code == 8'h1C) letter[0] <= 1'b1; //A
```

In the specific case of space being inserted into the system, the MSK is blanked out and the *run* value is incremented. To set up the *run* incrementation, two separate BCD case-blocks of code were created, one that would increment the ones digit of the counter, and one that would increment the tens digit of the counter, and a conditional was checked to determine that when a value of “9” was reached for either the ones or tens counter, that it would reset itself to 0, and based on whether the tens or ones was reset, either the tens counter would activate, or the *run* value would reset itself completely.

```
always_ff @(posedge CLOCK_50_I or negedge resetn) begin
	if(!resetn) begin
		run <= 7'd0;
	end else begin
	if (PS2_code_ready && ~PS2_code_ready_buf && PS2_make_code) begin
			// scan code detected
			PS2_reg <= PS2_code;
			if(PS2_code == 8'h29)begin
				if(run[3:0] == 4'd9) begin
					run[3:0] <= 4'd0;
					run[7:4] <= run[7:4] + 1'd1;
				end else begin
					run[3:0] <= run[3:0] + 1'd1;
				end
				if(run[7:4] == 4'd9 && run[3:0] == 4'd9) begin
					run<= 8'd0;
				end
		end
	end

	case (run[7:4])
		4'd0:	bcd1 = 6'o40; // space (should never show 0)
		4'd1:	bcd1 = 6'o61; // 1
		4'd2:	bcd1 = 6'o62; // 2
		4'd3:	bcd1 = 6'o63; // 3
		4'd4:	bcd1 = 6'o64; // 4
		4'd5:	bcd1 = 6'o65; // 5
		4'd6:	bcd1 = 6'o66; // 6
		4'd7:	bcd1 = 6'o67; // 7
		4'd8:	bcd1 = 6'o70; // 8
		4'd9:	bcd1 = 6'o71; // 9
		default: bcd1 = 6'o40; // 0
		endcase
```

We determined the amount of registers we used specifically in the main experiment file to be 31, which includes the following;

- Letter (10)
- Ps2_reg (8)
- Run (8)
- Delay_x_pos (3)
- Ps2_code_ready (1)
- Ps2_make_code (1)

| Entity:Instance | Logic Cells | Dedicated Logic Registers |
|---|---|---|
| Cyclone IV E: EP4CE115F29C7 | | |
| experiment4 | 221 (93) | 96 (31) |
| VGA_controller:VGA_unit | 90 (90) | 38 (38) |
| char_rom:char_rom_unit | 4 (4) | 0 (0) |
| PS2_controller:ps2_unit | 37 (37) | 27 (27) |

This was then confirmed inside of Quartus, and when summed with the register values used in the VGA Controller (38) and PS2 controller (27) for a combined total of 96 registers.