LAB01 Exercise Design Review

Lecturer : KH

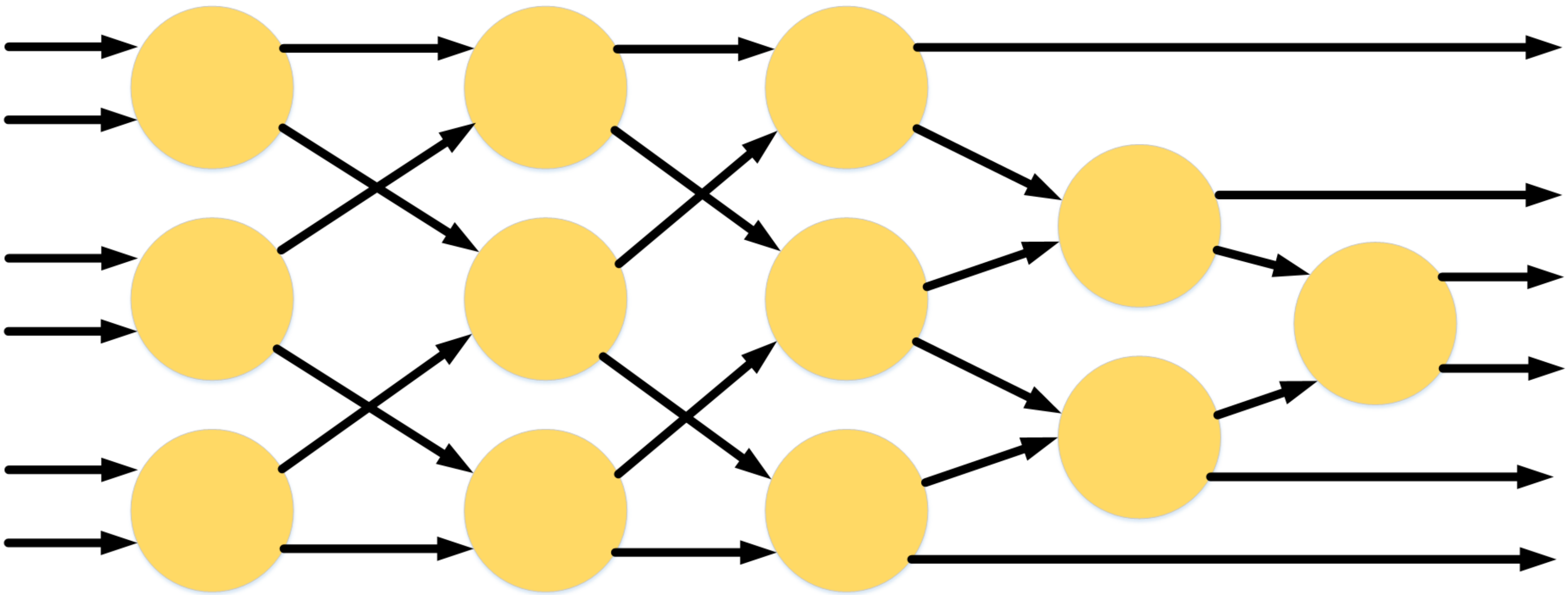
Date:2020/10/07

Outline

- Sort
- Normalization
- Equations
- Normalization + Equations

6 numbers sort architecture

- Use 12 comparators



sorter architecture(1/3)

	unsigned	signed		unsigned	signed
0000	0	0	1000	8	-8
0001	1	1	1001	9	-7
0010	2	2	1010	10	-6
0011	3	3	1011	11	-5
0100	4	4	1100	12	-4
0101	5	5	1101	13	-3
0110	6	6	1110	14	-2
0111	7	7	1111	15	-1

sorter architecture(2/3)

unsigned

In0[3]	In1[3]	
0	0	Depends on last three bits
0	1	in0 < in1
1	0	in0 > in1
1	1	Depends on last three bits

in1[3] \ in0[3]	0	1
0	1	x
1	0	1

signed

In0[3]	In1[3]	
0	0	Depends on last three bits
0	1	in0 > in1
1	0	in0 < in1
1	1	Depends on last three bits

in1[3] \ in0[3]	0	1
0	1	0
1	x	1

```
wire flg = (~s & (in0[3] | ~in1[3])) | (s & (~in0[3] | in1[3]))
wire abs_big = (~s & in0[3] & ~in1[3]) | (s & ~in0[3] & in1[3])
```

sorter architecture(3/3)

- Original -> use 5 bits 2's complement comparator
- Optimized -> use 3 bits unsigned comparator and 6inv 4and 4or gates

```
module comp(
    input           s,
    input    [3:0] in0,
    input    [3:0] in1,
    output   [3:0] out0,
    output   [3:0] out1
);

    wire abs_big = (~in0[3] & in1[3] & s) | (~s & in0[3] & ~in1[3]); ///in signed mode 0 > 1    in unsigned mode 1 > 0
    wire flg = (  (s & (~in0[3] | in1[3])) | (~s & (in0[3] | ~in1[3])) );
    wire bigger = (in0[2:0] > in1[2:0]) ;

    assign out0 = (abs_big | (flg & bigger) ) ? in0 : in1;
    assign out1 = (abs_big | (flg & bigger) ) ? in1 : in0;
endmodule
```

Convert unsigned number to signed number

- Let unsigned number and signed number use same hardware resources

```
wire signed [4:0] sort_result0 = {{s&sort_tmp0[3]},sort_tmp0};
wire signed [4:0] sort_result1 = {{s&sort_tmp1[3]},sort_tmp1};
wire signed [4:0] sort_result2 = {{s&sort_tmp2[3]},sort_tmp2};
wire signed [4:0] sort_result3 = {{s&sort_tmp3[3]},sort_tmp3};
wire signed [4:0] sort_result4 = {{s&sort_tmp4[3]},sort_tmp4};
wire signed [4:0] sort_result5 = {{s&sort_tmp5[3]},sort_tmp5};
```

Normalization

- Original -> use 6 MUX and 6 Subtractors

```
wire signed [4:0] norm_result0 = (opt[2]) ? (sort_result0 - avg) : sort_result1;
wire signed [4:0] norm_result1 = (opt[2]) ? (sort_result1 - avg) : sort_result2;
wire signed [4:0] norm_result2 = (opt[2]) ? (sort_result2 - avg) : sort_result3;
wire signed [4:0] norm_result3 = (opt[2]) ? (sort_result3 - avg) : sort_result4;
wire signed [4:0] norm_result4 = (opt[2]) ? (sort_result4 - avg) : sort_result5;
wire signed [4:0] norm_result5 = (opt[2]) ? (sort_result5 - avg) : sort_result6;
```

- Optimized -> use 1 MUX and 6 Subtractors

```
wire signed [4:0] norm_val = (opt[2]) ? avg :  0;
wire signed [4:0] norm_result0 = sort_result0 - norm_val;
wire signed [4:0] norm_result1 = sort_result1 - norm_val;
wire signed [4:0] norm_result2 = sort_result2 - norm_val;
wire signed [4:0] norm_result3 = sort_result3 - norm_val;
wire signed [4:0] norm_result4 = sort_result4 - norm_val;
wire signed [4:0] norm_result5 = sort_result5 - norm_val;
```

Equations(1/2)

- Eq0 : Use 1 Subtractor , 1 Adder , 1 Multiplier and 1 Divider
- Eq1 : Use 1 Subtractor , 1 Adder , 1 Multiplier and 1 Comparator

```
//================================================================
//    Eq0
//================================================================
    wire [8:0] eq0 = (n0-n1*n2+n5)/3;
//================================================================
//    Eq1
//================================================================
    wire signed [8:0] eq1_tmp = (n3<<1 + n3) - n0*n4;
    wire signed [8:0] eq1 = (eq1_tmp > 0) ? eq1_tmp : eq1_tmp * -1;
```

Equations(2/2)

- Eq0 / Eq1 share 1 Subtractor , 1 Adder , 1 Multiplier
- Cost : need 6 MUX to select input data

```
wire signed [5:0] add_in1 = (equ) ? n3*2  : n0;
wire signed [4:0] add_in2 = (equ) ? n3    : n5;
wire signed [4:0] mul_in1 = (equ) ? n0    : n1;
wire signed [4:0] mul_in2 = (equ) ? n4    : n2;
wire signed [8:0] tmp0 = (add_in1 + add_in2) - (mul_in1 * mul_in2);
```

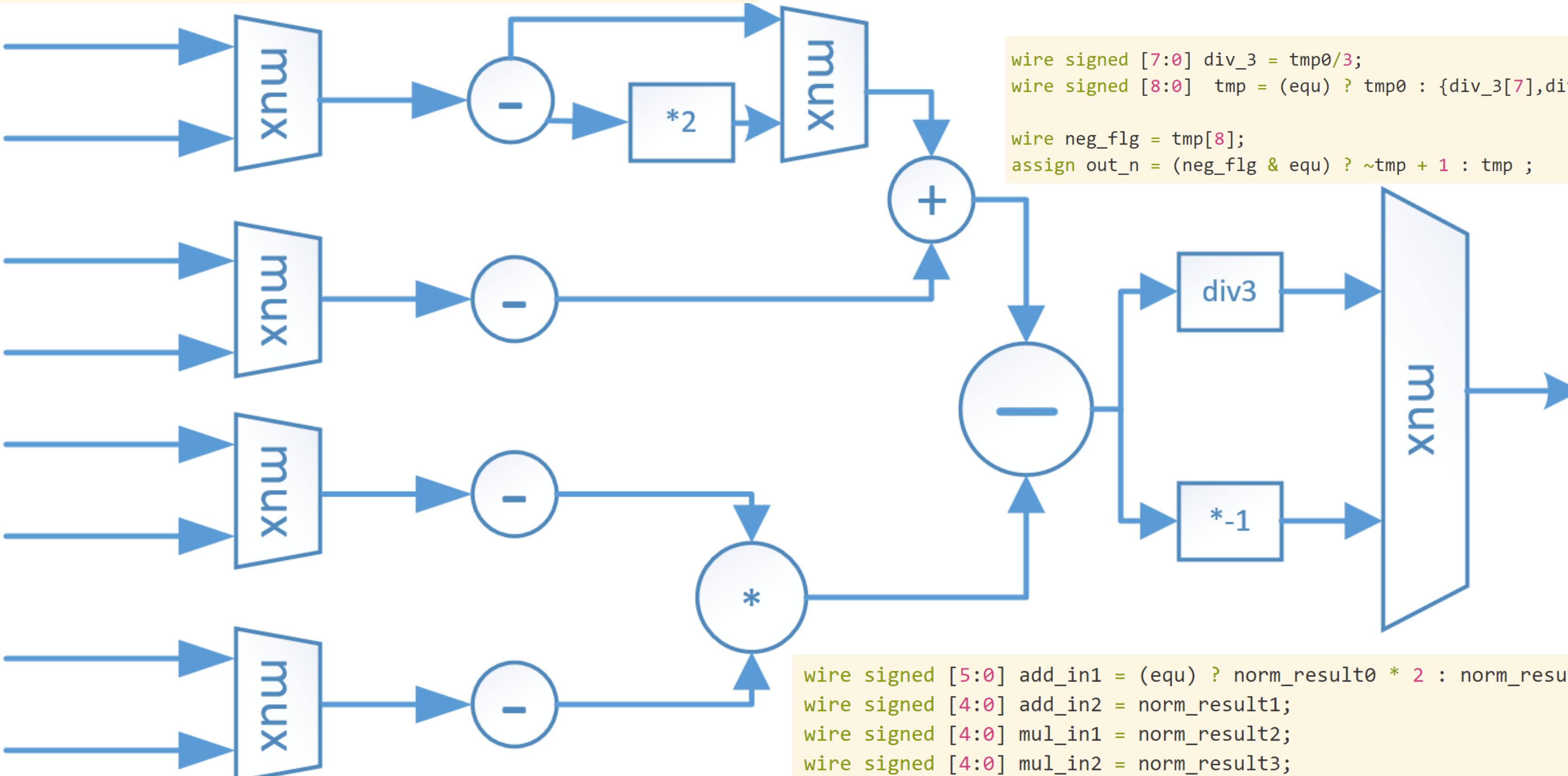
Normalization + Equations

- In fact , we just need 4 Subtractors
- Original -> use 6 Subtractors
- Optimized -> use 4 Subtractors

1.	1 2 3 4 Eq0 : ((n0 – n1 * n2 + n5) / 3) (round-down the answer if it is not integer) For example, if result is -3.75, round down to -3, if result is 5.5, round down to 5
2.	1 2 3 Eq1 : | n3 * 3 – n0 * n4 |
(Hint: Try to use behavior modeling description instead of gate level description)	

Normalization + Equations architecture

```
wire signed [4:0] norm_mux_out0 = (equ) ? sort_result3 : sort_result0;
wire signed [4:0] norm_mux_out1 = (equ) ? sort_result3 : sort_result5;
wire signed [4:0] norm_mux_out2 = (equ) ? sort_result0 : sort_result1;
wire signed [4:0] norm_mux_out3 = (equ) ? sort_result4 : sort_result2;
```




```
wire signed [7:0] div_3 = tmp0/3;
wire signed [8:0]  tmp = (equ) ? tmp0 : {div_3[7],div_3};

wire neg_flg = tmp[8];
assign out_n = (neg_flg & equ) ? ~tmp + 1 : tmp ;
```

```
wire signed [5:0] add_in1 = (equ) ? norm_result0 * 2 : norm_result0;
wire signed [4:0] add_in2 = norm_result1;
wire signed [4:0] mul_in1 = norm_result2;
wire signed [4:0] mul_in2 = norm_result3;
wire signed [8:0] tmp0 = (add_in1 + add_in2) - (mul_in1 * mul_in2);
```