

18/03/2019

EE446 LABORATORY

EXPERIMENT 2

PRELIMINARY REPORT

Muttalip Caner TOL

2031466

Tuesday Afternoon

1.2.1. Datapath Design

3. I have used Booth's algorithm which is shown in Figure 1, in order to handle with the signed multiplication. Therefore, we should carry the $Q[0]$ and $Q[-1]$ bits as the control unit inputs from the Datapath.

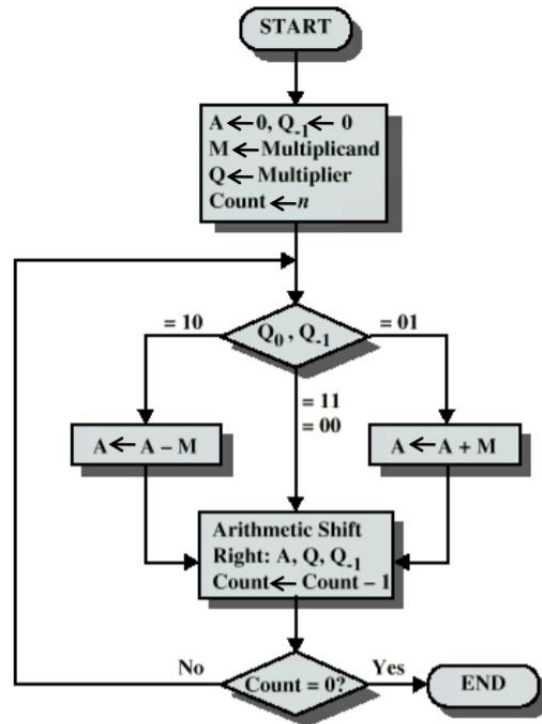


Figure 1. Booth's Signed Multiplication Algorithm

In Figure 1, A and Q are implemented as two shift register. Q_{-1} is another 1-bit register. "n" is the number of bits of the operands. At the end of the algorithm, A holds the most significant half byte and Q holds the least significant half byte of the result.

4. My signed division algorithm is as follows:

- If R1 is negative, take 2's complement of R1
- If R0 is negative, take 2's complement of R0
- Apply non-restoring division algorithm which is shown in Figure 2.
- Take 2's complement of the Quotient if R0 and R1 have different signs at the beginning.
- Take 2's complement of the Remainder if Dividend is negative at the beginning

Since we have used non-restoring division algorithm, we need to carry the sign bit of the A register to control unit.

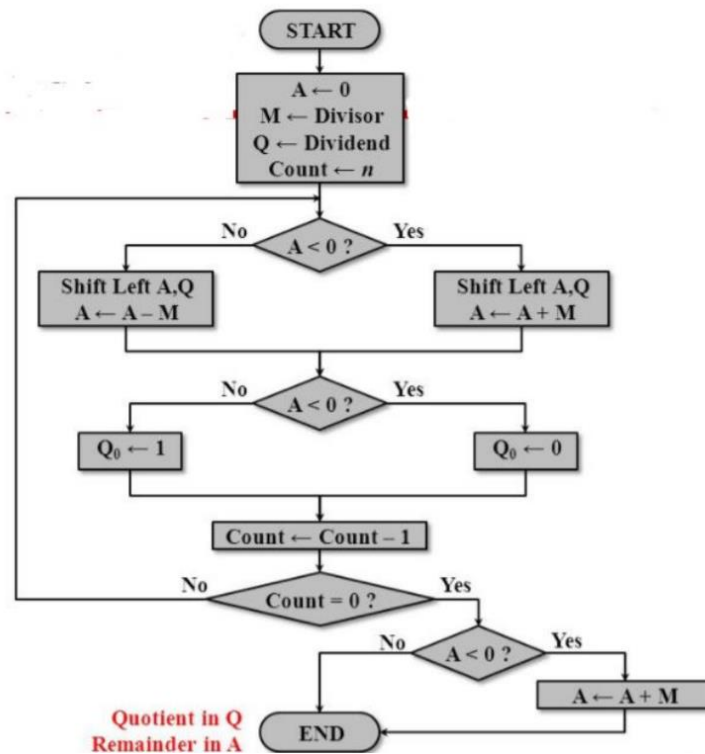


Figure 2. Non-restoring division algorithm

1.2.2. Controller Design

- Figure 3 shows the controller unit.

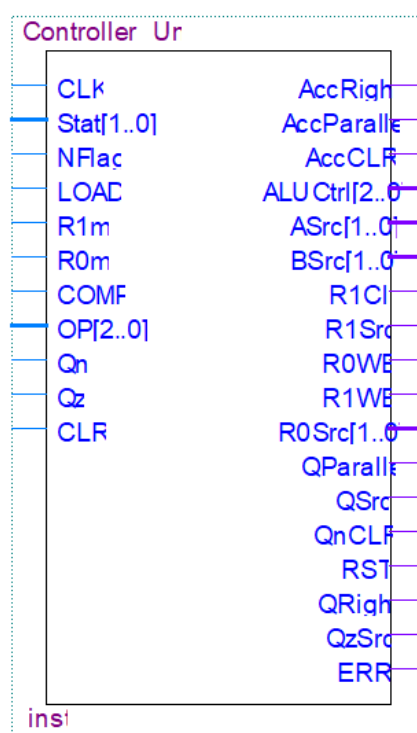


Figure 3. Controller Unit as a block box

2.

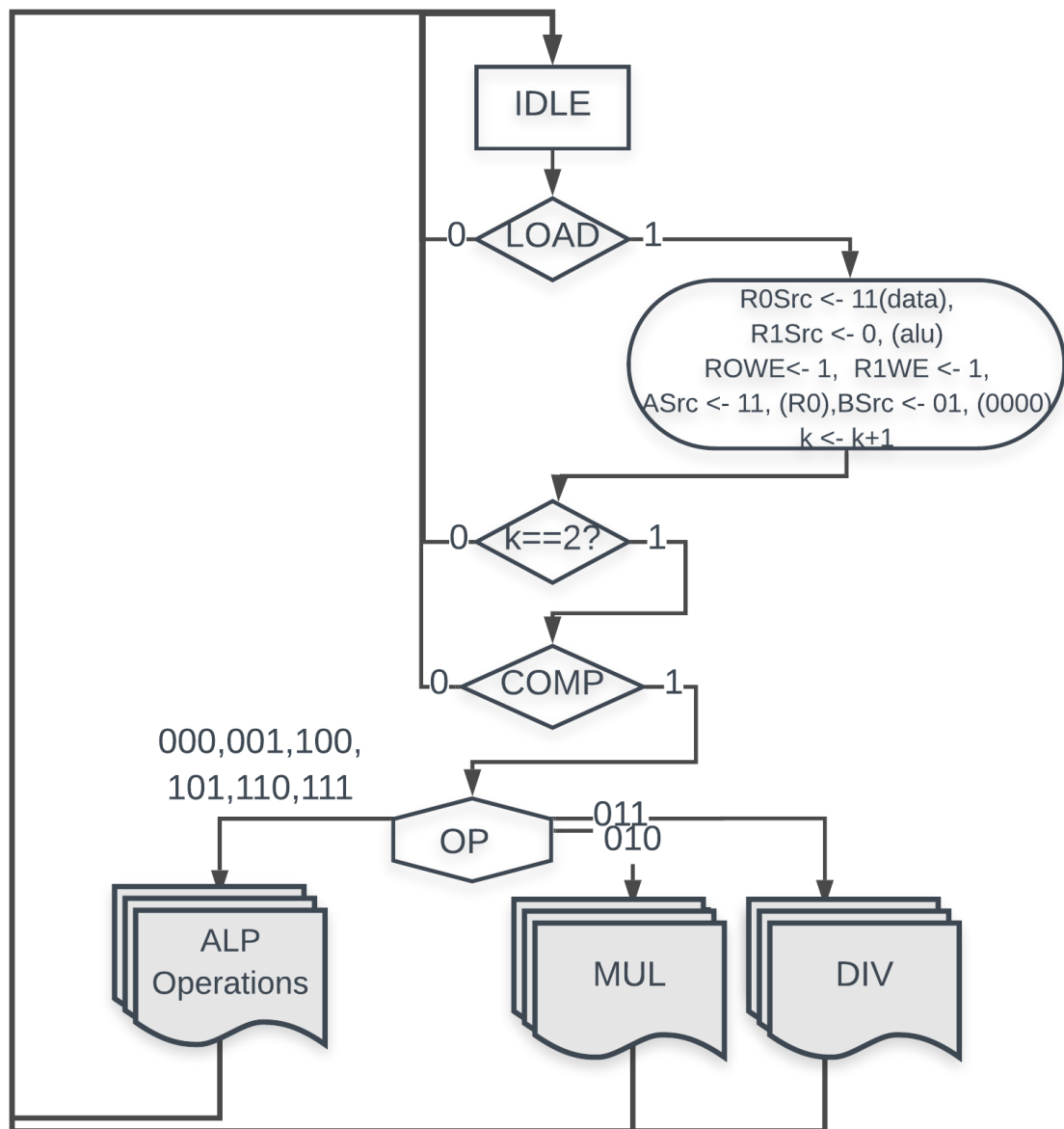


Figure 4. First black box

3.

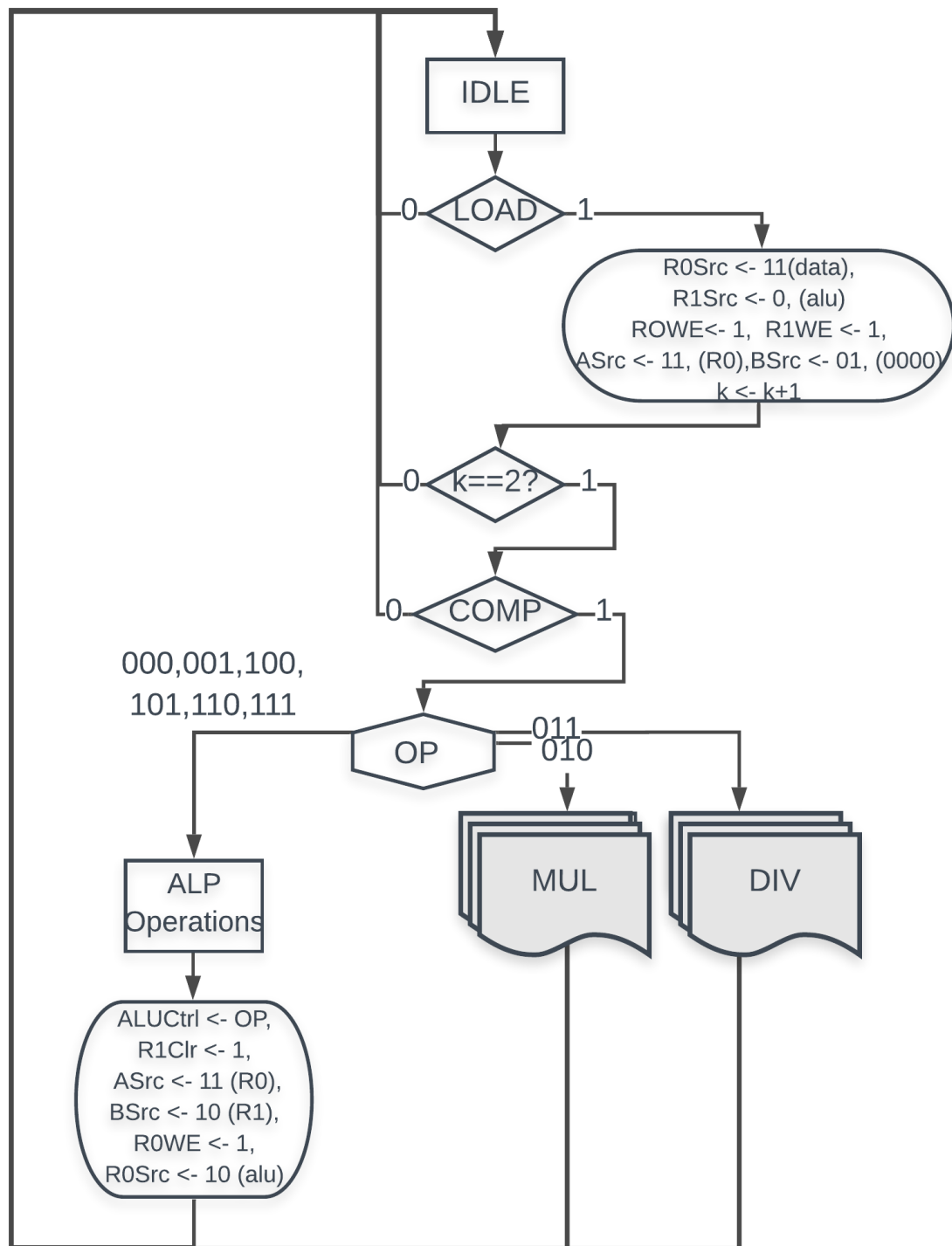


Figure 5. ASM Chart of the second black box connected to the first one

4.

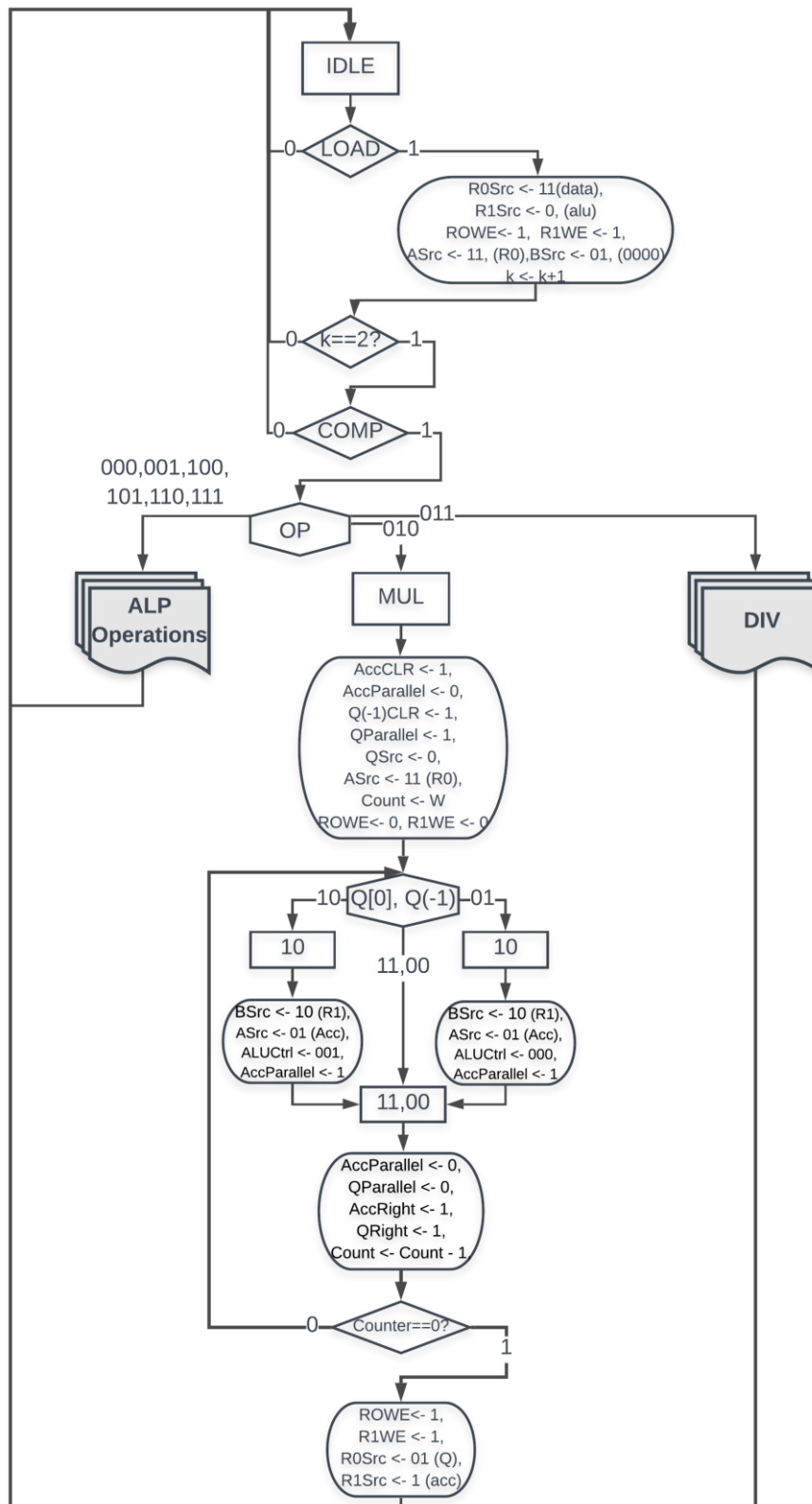


Figure 6. ASM Chart of the multiplication algorithm

Since we are using Booth's algorithm in our multiplication algorithm, it is both compatible with both signed and unsigned numbers.

5.

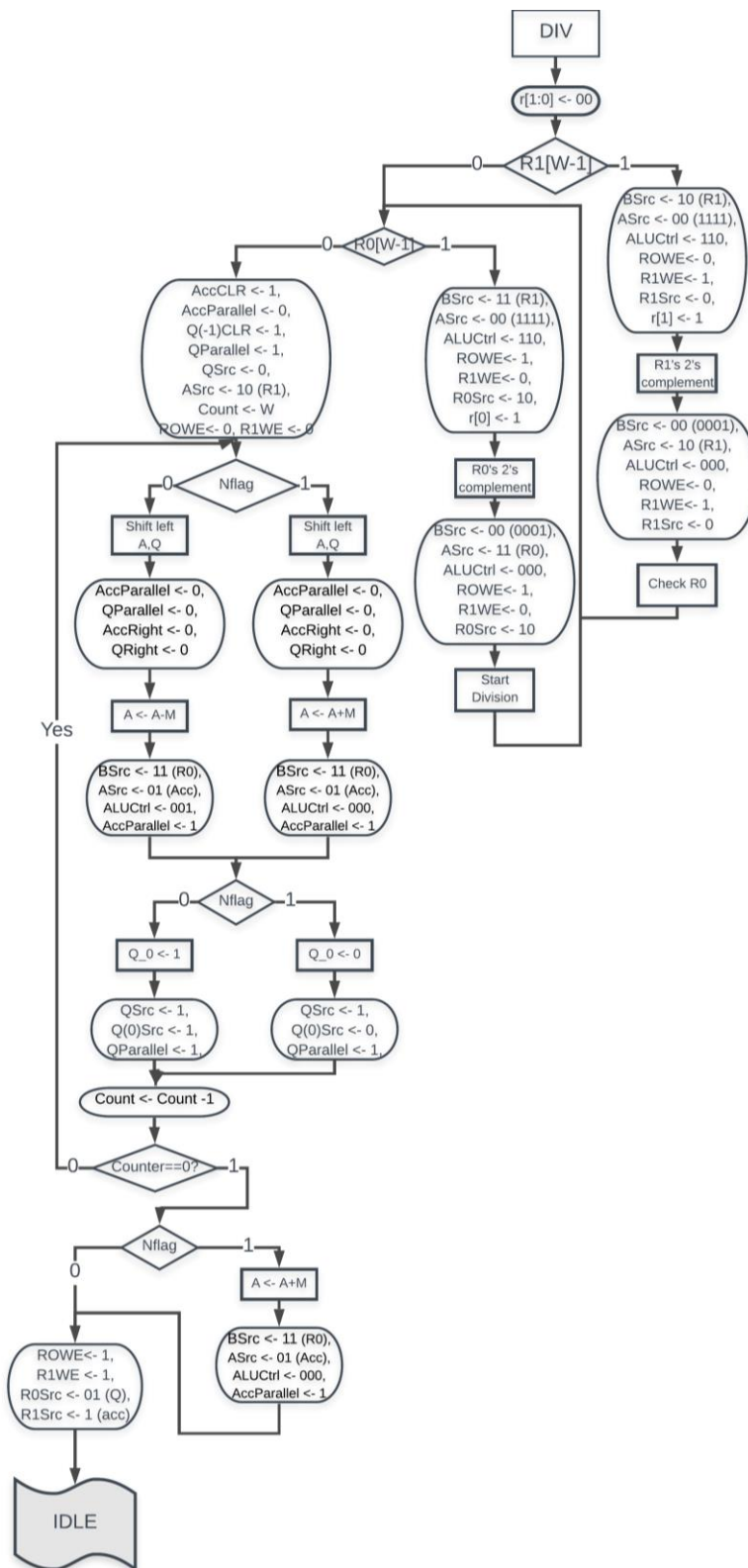


Figure 7. ASM Chart of the division algorithm

Since we check the operands and updates the quotient and remainder signs, we are able to use signed numbers for our multiplication controller.

6. For logic and arithmetic operations except the multiplication and division, they take 4 cycles to finish the the operation.

Multiplication algorithm takes 16 cycles to complete.

Division algorithm takes 24 cycles at worst case.

7. My ASM chart has 22 states. I am using the Op signal as input directly to the ALU Unit. Therefore there is no need for additional datapath objects to connect the OP with ALUCtrl.


```

1  module Controller_Unit #(parameter W=4) (CLK,
2      AccRight, AccParallel, AccCLR, // Acc register control
3      ALUCtrl, ASrc, BSrc, // ALU Controllers
4      Stat, NFlag, // Status bits
5      LOAD, R1m, R0m,
6      COMP, R1Clr, R1Src, ROWE, R1WE, R0Src,
7      OP, //ALP Operation
8      QParallel, QSrc, QnCLR, RST, QRight, QzSrc, Qn, Qz,
9      CLR, // reset registers
10     ERR // Arithmetic overflow
11
12     );
13     input CLK, CLR;
14     input LOAD, COMP;
15     input [2:0] OP;
16     input [1:0] Stat;
17     input R1m, R0m;
18     output reg R1Clr=0, R1Src, ROWE=0, R1WE=0;
19     output reg [1:0] R0Src, ASrc, BSrc;
20     output reg [2:0] ALUCtrl;
21     input NFlag, Qn, Qz;
22     output reg ERR=0;
23     output reg AccRight, AccParallel, AccCLR, QParallel, QSrc, QnCLR=0, RST=0,
24     QRight, QzSrc;
25
26     parameter [4:0] ST0 =0, ST1=1, ST2=2, ST3=3, ST4 =4, ST5=5, ST6=6,
27     ST7=7, ST8=8, ST9=9, ST10=10, ST11=11, ST12=12, ST13=13,
28     ST14=14, ST15=15, ST16=16, ST17=17, ST18=18, ST19=19,
29     ST20=20, ST21=21;
30
31     integer k=0, Count=W;
32     reg [1:0] r;
33     reg NS, CS;
34
35     initial
36     begin
37         CS = ST0;
38         NS = ST0;
39     end
40
41     always @(CLK,CLR,LOAD, COMP, OP, Stat,R1m, R0m,NFlag ,r, Count, Qn, Qz)
42     begin : COMB
43
44         case(CS)
45         ST0: begin
46
47             if(k>=2 && COMP)
48                 case(OP)
49                     3'b000 : NS = ST1;
50                     3'b001 : NS = ST1;
51                     3'b010 : NS = ST2;
52                     3'b011 : NS = ST3;
53                     3'b100 : NS = ST1;
54                     3'b101 : NS = ST1;
55                     3'b110 : NS = ST1;
56                     3'b111 : NS = ST1;
57                 endcase
58             else
59                 NS = ST0;
60             end
61
62         ST1: begin
63             NS = ST0;
64             end
65
66         ST2: begin
67             case({Qz,Qn})
68                 2'b00 : NS = ST6;
69
70                 2'b01 : NS = ST4;
71
72                 2'b10 : NS = ST5;

```

```

72
73         2'b11 : NS = ST6;
74
75     endcase
76 end
77
78 ST3: begin
79     if(R1m)
80         NS = ST7;
81     else if(R0m)
82         NS = ST9;
83     else if(NFlag)
84         NS = ST12;
85     else
86         NS = ST11;
87     end
88
89 ST4: begin
90     NS = ST5;
91     end
92
93 ST5: begin
94     NS = ST0;
95     end
96
97 ST6: begin
98     if(Count==0)
99         NS = ST0;
100    else
101        case({Qz,Qn})
102            2'b00 : NS = ST6;
103
104            2'b01 : NS = ST4;
105
106            2'b10 : NS = ST5;
107
108            2'b11 : NS = ST6;
109
110        endcase
111    end
112
113 ST7: begin
114     NS = ST8;
115     end
116
117 ST8: begin
118     if(R0m)
119         NS = ST9;
120     else if(NFlag)
121         NS = ST12;
122     else
123         NS = ST11;
124     end
125
126 ST9: begin
127     NS = ST10;
128     end
129
130 ST10: begin
131     if(NFlag)
132         NS = ST12;
133     else
134         NS = ST11;
135     end
136
137 ST11: begin
138     NS = ST13;
139     end
140
141 ST12: begin
142     NS = ST14;
143     end
144

```

```

145 ST13: begin
146     if(NFlag)
147         NS = ST16;
148     else
149         NS = ST15;
150     end
151
152 ST14: begin
153     if(NFlag)
154         NS = ST16;
155     else
156         NS = ST15;
157     end
158
159 ST15: begin
160     if(Count==0)
161         begin
162             if(NFlag)
163                 NS = ST17;
164             else
165                 NS = ST0;
166             end
167         else
168             if(NFlag)
169                 NS = ST12;
170             else
171                 NS = ST11;
172             end
173         end
174
175 ST16: begin
176     if(Count==0)
177         begin
178             if(NFlag)
179                 NS = ST17;
180             else
181                 NS = ST0;
182             end
183         else
184             if(NFlag)
185                 NS = ST12;
186             else
187                 NS = ST11;
188             end
189         end
190
191 ST17: begin
192     NS = ST18;
193     end
194
195 ST18: begin
196     if(r[1])
197         NS = ST19;
198     else if(r[0] ^ r[1])
199         NS = ST21;
200     else
201         NS = ST0;
202     end
203
204 ST19: begin
205     NS = ST20;
206     end
207
208 ST20: begin
209     if(r[0] ^ r[1])
210         NS = ST21;
211     else
212         NS = ST0;
213     end
214
215 ST21: begin
216     NS = ST0;
217     end
218
219 endcase
220 end

```

```

always @(posedge CLK or posedge CLR)
begin : SEQ
    if(CLR)
        CS <= ST0;
    else
        CS <= NS;
    end

always @(CLK, CLR, LOAD, COMP, OP, Stat, R1m, R0m, NFlag, r, Count, Qn, Qz)
begin: OUT
    ERR = Stat[1];
    RST = CLR;
    AccCLR = CLR;
    QnCLR = CLR;
    R1Clr = CLR;

    if (CLR!=1)

        begin

            case(CS)
                ST0 :
                    if(LOAD==1)
                        begin
                            R0Src = 2'b11;
                            R1Src = 1'b0;
                            R0WE = 1'b1;
                            R1WE = 1'b1;
                            ASrc = 2'b11;
                            BSrc = 2'b01;
                            k = k + 1;
                        end

                ST1 : begin
                            ALUCtrl = OP;
                            R1Clr = 1'b1;
                            ASrc = 2'b11;
                            BSrc = 2'b10;
                            R0WE = 1'b1;
                            R1WE = 1'b0;
                            R0Src = 2'b10;
                        end

                ST2 : begin
                            AccCLR = 1'b1;
                            AccParallel = 1'b0;
                            QnCLR = 1'b1;
                            QParallel = 1'b1;
                            QSrc = 1'b0;
                            ASrc = 2'b11;
                            Count = W;
                            R0WE = 1'b0;
                            R1WE = 1'b0;
                        end

                ST3 : begin
                            r = 2'b00;
                            if(R1m)
                                begin
                                    BSrc = 2'b10;
                                    ASrc = 2'b00;
                                    ALUCtrl = 3'b110;
                                    R0WE = 1'b0;
                                    R1WE = 1'b1;
                                    R1Src = 1'b0;
                                    r[1] = 1'b1;
                                end
                            else if(R0m)
                                begin
                                    BSrc = 2'b11;
                                    ASrc = 2'b00;
                                    ALUCtrl = 3'b110;
                                end
                        end
            endcase
        end
    end
end

```

```

291         ROWE = 1'b1;
292         R1WE = 1'b0;
293         R0Src = 2'b10;
294         r[0] = 1'b1;
295         end
296     else
297         begin
298             AccCLR = 1'b1;
299             AccParallel = 1'b0;
300             QnCLR = 1'b1;
301             QParallel = 1'b1;
302             QSrc = 1'b0;
303             ASrc = 2'b10;
304             Count = W;
305             ROWE = 1'b0;
306             R1WE = 1'b0;
307             end
308         end
309
310     ST4 : begin
311         BSrc = 2'b10;
312         ASrc = 2'b01;
313         ALUCtrl = 3'b000;
314         AccParallel = 1'b1;
315         end
316
317     ST5 : begin
318         BSrc = 2'b10;
319         ASrc = 2'b01;
320         ALUCtrl = 3'b001;
321         AccParallel = 1'b1;
322         end
323
324     ST6 : begin
325         AccParallel = 1'b0;
326         QParallel = 1'b0;
327         AccRight = 1'b1;
328         QRight = 1'b1;
329         Count = Count -1;
330         if(Count==0)
331             begin
332                 ROWE = 1'b1;
333                 R1WE = 1'b1;
334                 R0Src = 2'b01;
335                 R1Src = 1'b1;
336                 end
337             end
338
339     ST7 : begin
340         BSrc = 2'b00;
341         ASrc = 2'b10;
342         ALUCtrl = 3'b000;
343         ROWE = 1'b0;
344         R1WE = 1'b1;
345         R1Src = 1'b0;
346         end
347
348     ST8 : begin
349         if(R0m)
350             begin
351                 BSrc = 2'b11;
352                 ASrc = 2'b00;
353                 ALUCtrl = 3'b110;
354                 ROWE = 1'b1;
355                 R1WE = 1'b0;
356                 R0Src = 2'b10;
357                 r[0] = 1'b1;
358                 end
359             else
360                 begin
361                     AccCLR = 1'b1;
362                     AccParallel = 1'b0;
363                     QnCLR = 1'b1;

```

```

364         QParallel = 1'b1;
365         QSrc = 1'b0;
366         ASrc = 2'b10;
367         Count = W;
368         ROWE = 1'b0;
369         R1WE = 1'b0;
370         end
371     end
372
373 ST9 : begin
374     BSrc = 2'b00;
375     ASrc = 2'b11;
376     ALUCtrl = 3'b000;
377     ROWE = 1'b1;
378     R1WE = 1'b0;
379     R0Src = 2'b10;
380     end
381
382 ST10 : begin
383     AccCLR = 1'b1;
384     AccParallel = 1'b0;
385     QnCLR = 1'b1;
386     QParallel = 1'b1;
387     QSrc = 1'b0;
388     ASrc = 2'b10;
389     Count = W;
390     ROWE = 1'b0;
391     R1WE = 1'b0;
392     end
393
394 ST11 : begin
395     AccParallel = 1'b0;
396     QParallel = 1'b0;
397     AccRight = 1'b0;
398     QRight = 1'b0;
399     end
400
401 ST12 : begin
402     AccParallel = 1'b0;
403     QParallel = 1'b0;
404     AccRight = 1'b0;
405     QRight = 1'b0;
406     end
407
408 ST13 : begin
409     BSrc = 2'b11;
410     ASrc = 2'b01;
411     ALUCtrl = 3'b001;
412     AccParallel = 1'b1;
413     end
414
415 ST14 : begin
416     BSrc = 2'b11;
417     ASrc = 2'b01;
418     ALUCtrl = 3'b000;
419     AccParallel = 1'b1;
420     end
421
422 ST15 : begin
423     QSrc = 1'b1;
424     QzSrc = 1'b1;
425     QParallel = 1'b1;
426     Count = Count - 1;
427     if((Count == 0) && NFlag)
428         begin
429             ROWE = 1'b1;
430             R1WE = 1'b1;
431             R0Src = 2'b01;
432             R1Src = 1'b1;
433             end
434     end
435
436 ST16 : begin

```

```
437         QSrc = 1'b1;
438         QzSrc = 1'b0;
439         QParallel = 1'b1;
440         Count = Count - 1;
441         if((Count == 0) && NFlag)
442             begin
443                 ROWE = 1'b1;
444                 R1WE = 1'b1;
445                 R0Src = 2'b01;
446                 R1Src = 1'b1;
447             end
448         end
449
450     ST17 : begin
451         BSrc = 2'b11;
452         ASrc = 2'b01;
453         ALUCtrl = 3'b000;
454         AccParallel = 1'b1;
455
456         ROWE = 1'b1;
457         R1WE = 1'b1;
458         R0Src = 2'b01;
459         R1Src = 1'b1;
460     end
461 endcase
462 end
463 end
464
465 endmodule
466
```