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.

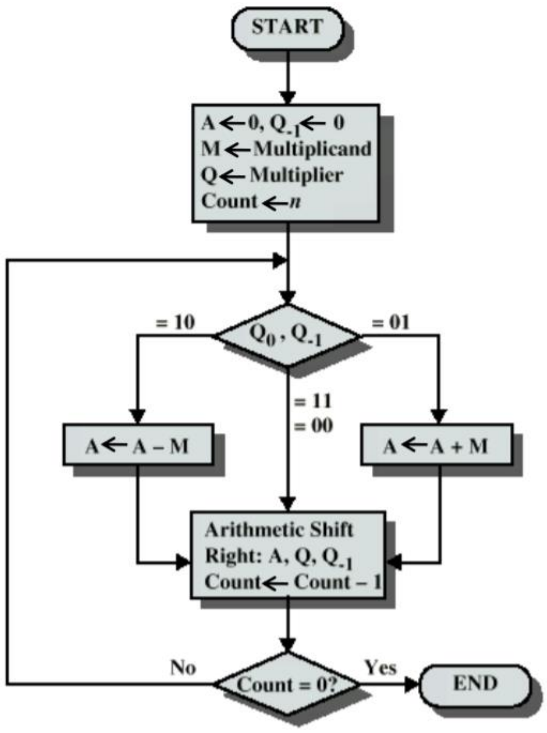


Figure 1. Booth's Signed Multiplication Algorithm

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 algoritm, 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.

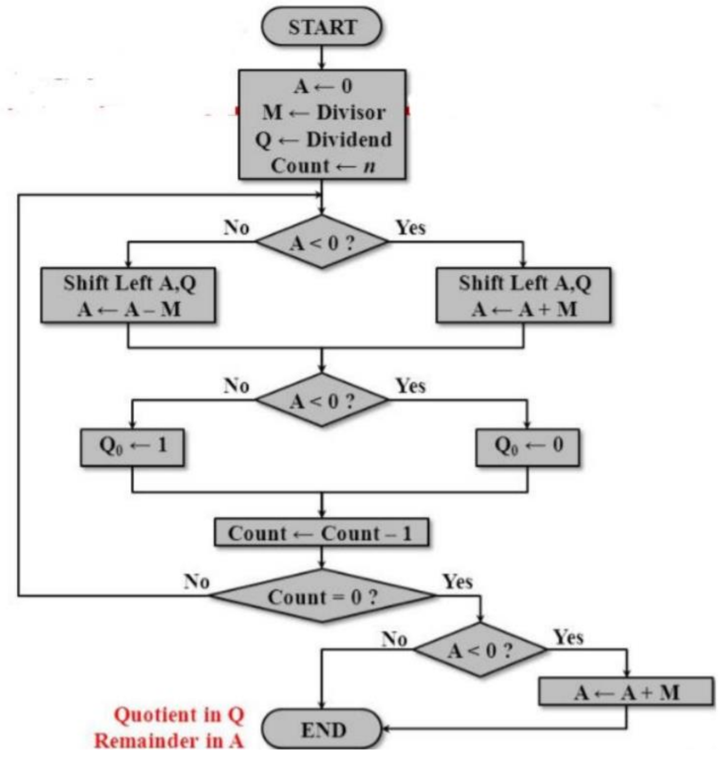


Figure 2. Non-restoring division algorithm

1.2.2. Controller Design

1. Figure 3 shows the controller unit.

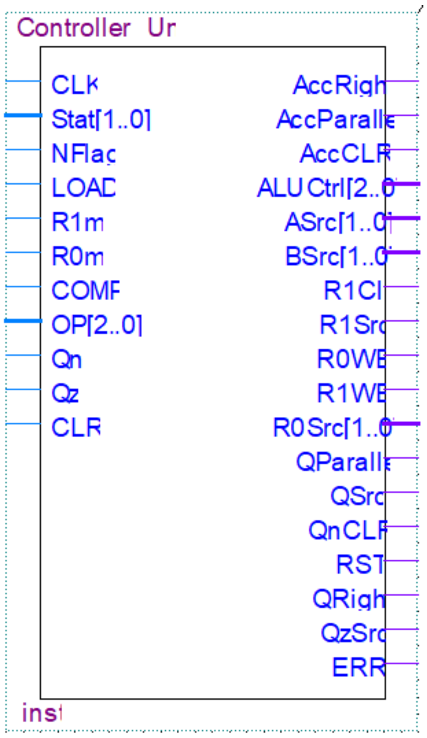


Figure 3. Controller Unit as a block box

2.

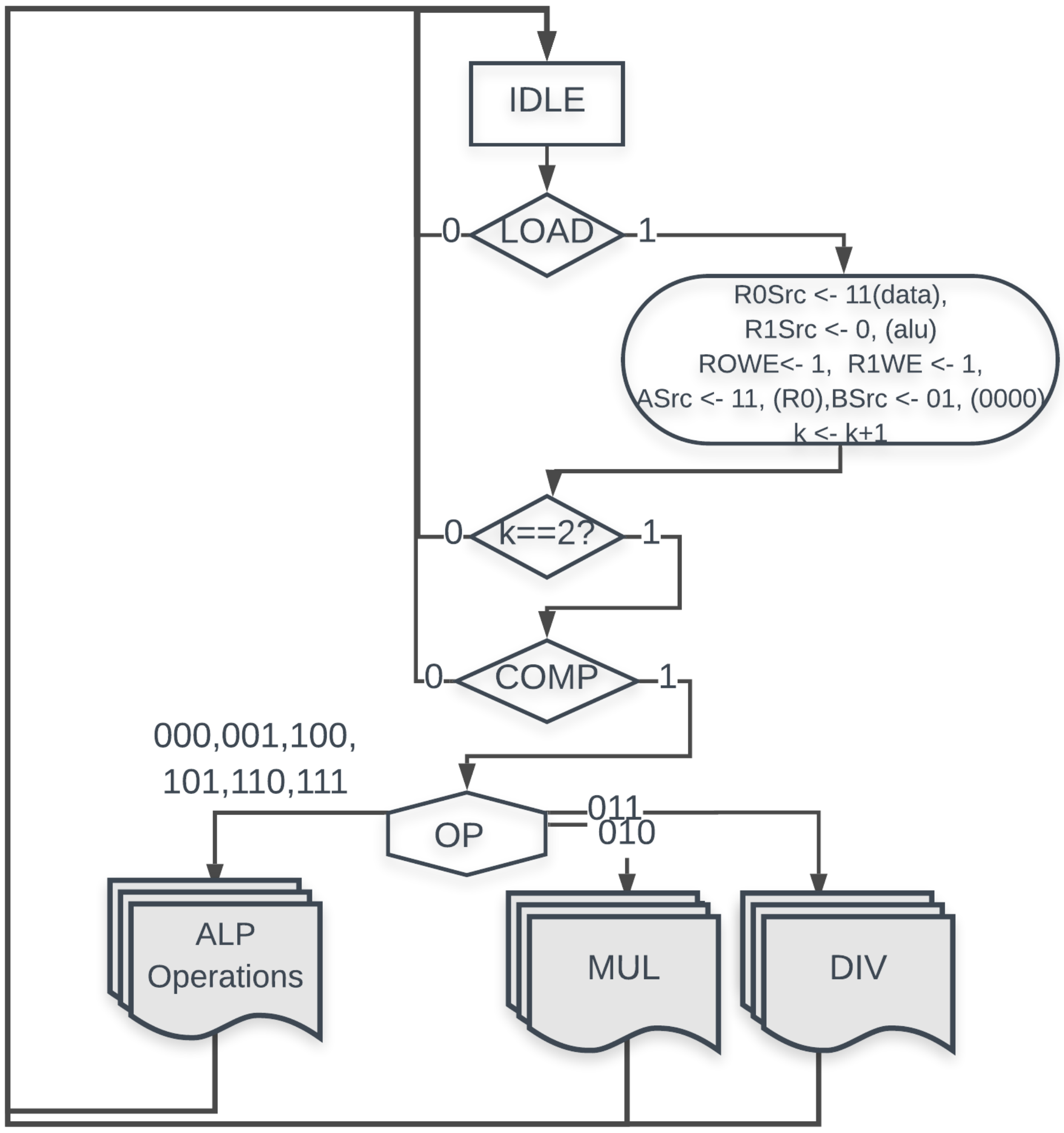


Figure 4. First black box

s

3.

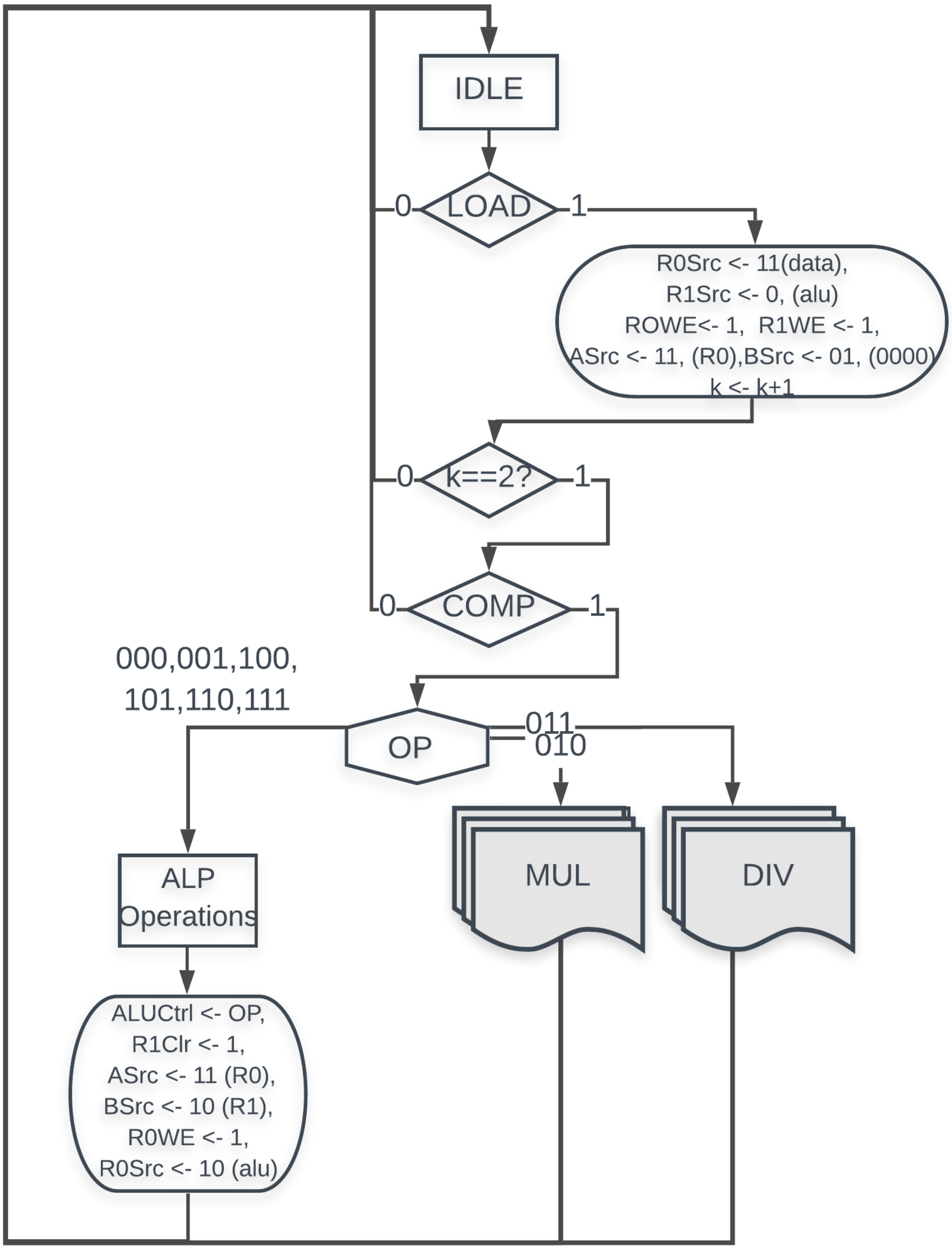


Figure 5. ASM Chart of the second black box connected to the first one

4.

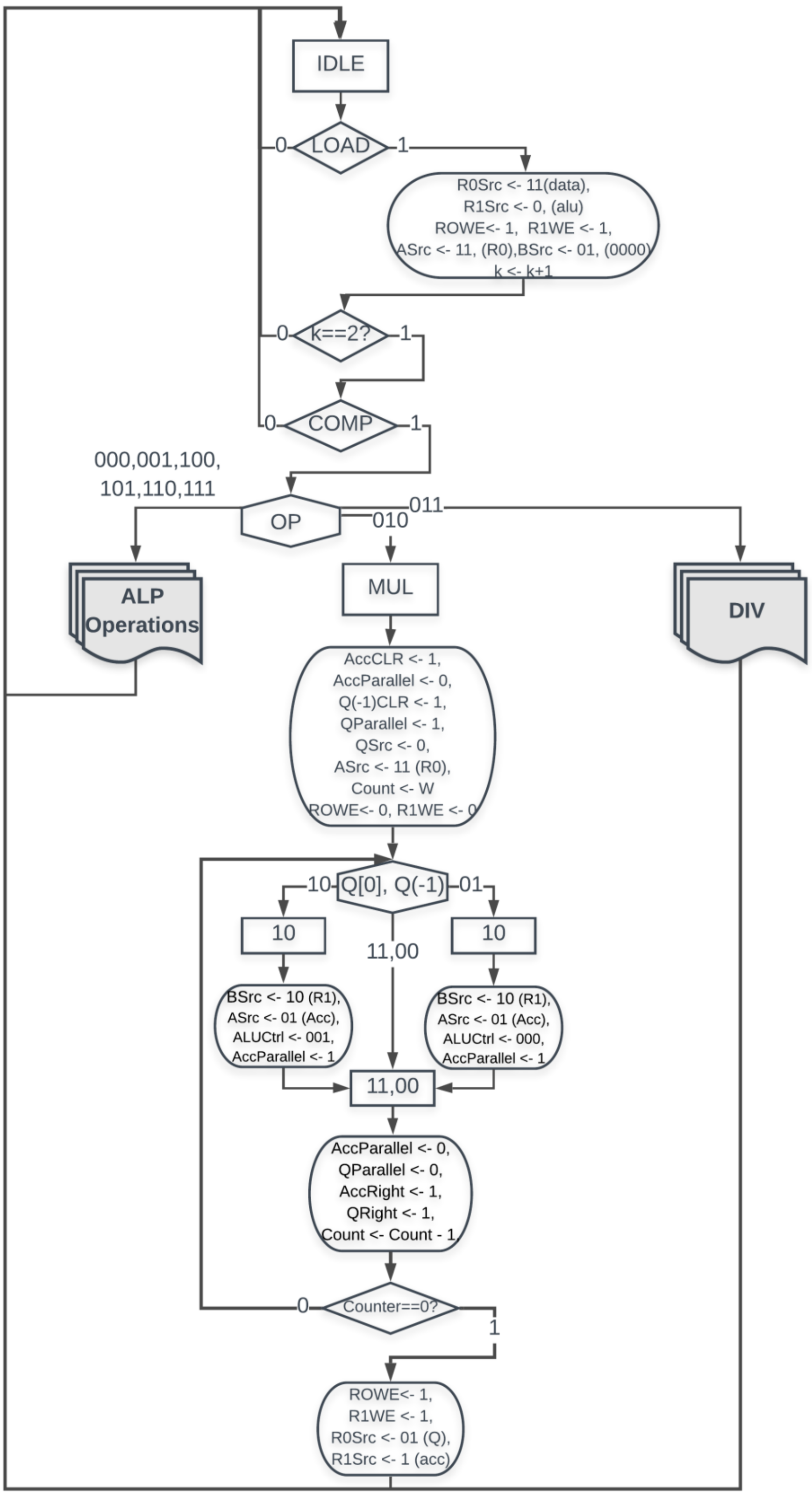


Figure 6. ASM Chart of the multiplication algorithm

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

5.

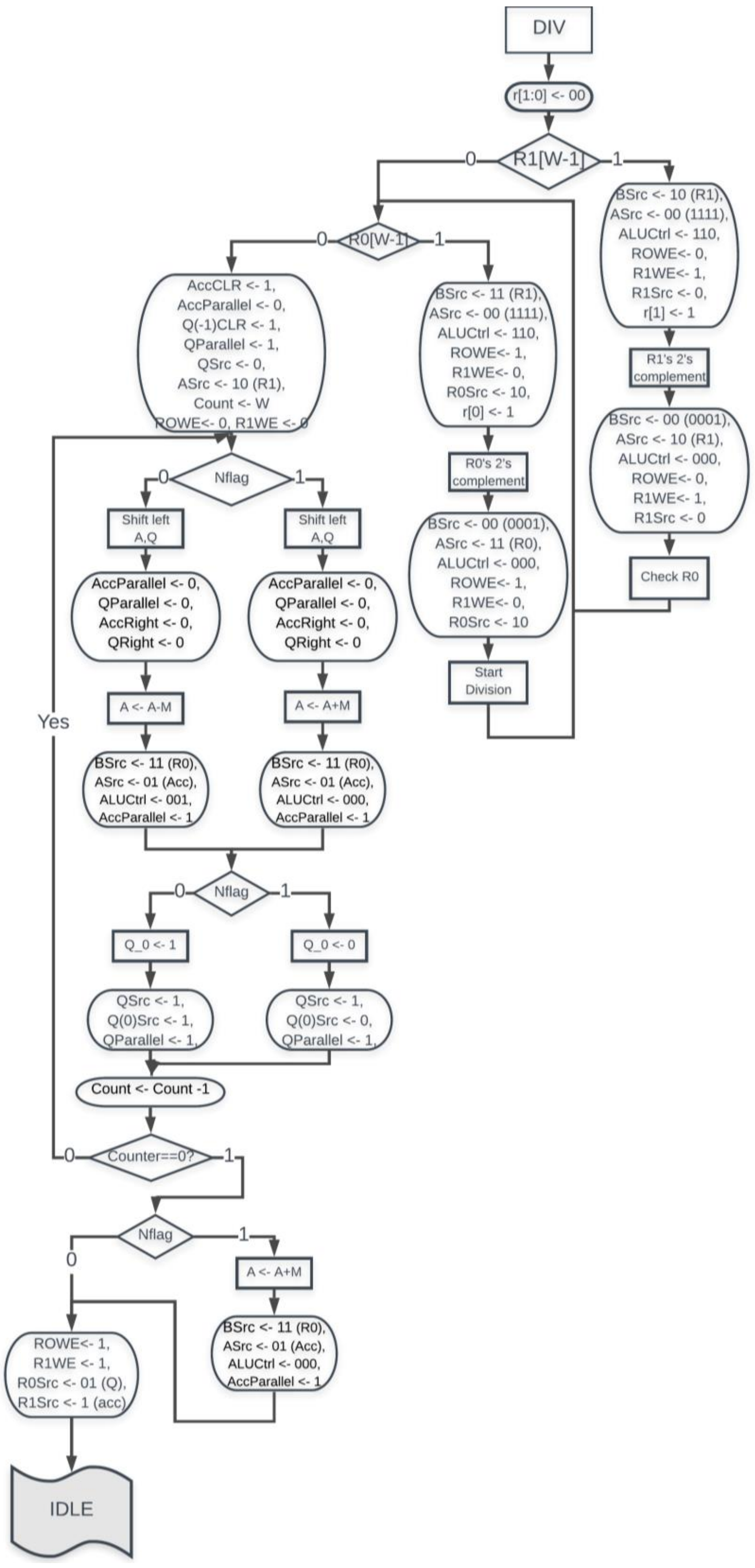

DIV
r[1:0] <- 00
R1[W-1]
0
1
BSrc <- 10 (R1), ASrc <- 00 (1111), ALUCtrl <- 110, ROWE<- 0, R1WE<- 1, R1Src <- 0, r[1] <- 1
R1's 2's complement
BSrc <- 00 (0001), ASrc <- 10 (R1), ALUCtrl <- 000, ROWE<- 0, R1WE<- 1, R1Src <- 0
Check R0
R0[W-1]
BSrc <- 11 (R1), ASrc <- 00 (1111), ALUCtrl <- 110, ROWE<- 1, R1WE<- 0, R0Src <- 10, r[0] <- 1
R0's 2's complement
BSrc <- 00 (0001), ASrc <- 11 (R0), ALUCtrl <- 000, ROWE<- 1, R1WE<- 0, R0Src <- 10
Start Division
AccCLR <- 1, AccParallel <- 0, Q(-1)CLR <- 1, QParallel <- 1, QSrc <- 0, ASrc <- 10 (R1), Count <- W ROWE<- 0, R1WE <- 0
Nflag
Shift left A,Q
AccParallel <- 0, QParallel <- 0, AccRight <- 0, QRight <- 0
A <- A-M
A <- A+M
BSrc <- 11 (R0), ASrc <- 01 (Acc), ALUCtrl <- 001, AccParallel <- 1
BSrc <- 11 (R0), ASrc <- 01 (Acc), ALUCtrl <- 000, AccParallel <- 1
Yes
Q_0 <- 1
Q_0 <- 0
QSrc <- 1, Q(0)Src <- 1, QParallel <- 1,
QSrc <- 1, Q(0)Src <- 0, QParallel <- 1,
Count <- Count -1
Counter==0?
ROWE<- 1, R1WE <- 1, R0Src <- 01 (Q), R1Src <- 1 (acc)
IDLE

Figure 7. ASM Chart of the division algorithm

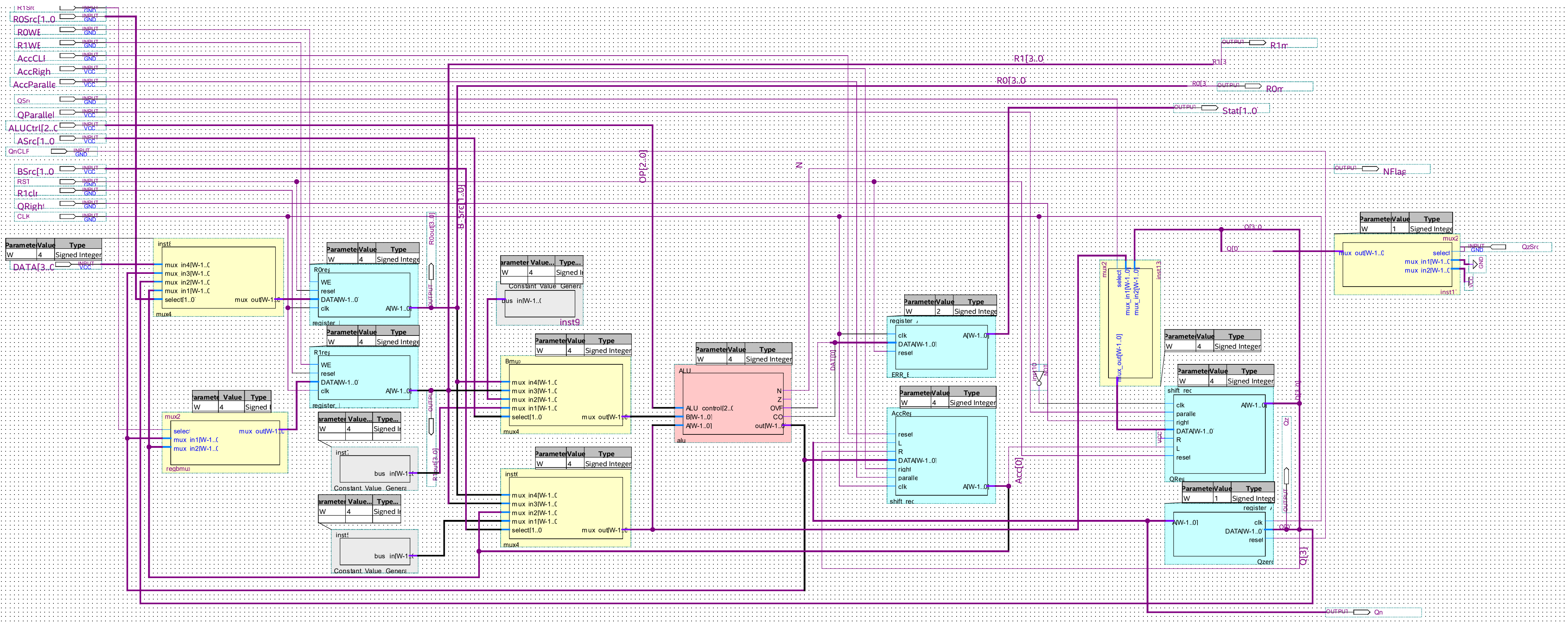
Since we check the operands and updates the quotient and remainder signes, we are able to use signed numbers for our multiplication controller.

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.


R1Src
R0Src[1..0
R0WE
R1WE
AccCLF
AccRigh
AccParalle
QSh
QParallel
ALUCtrl[2..0
ASrc[1..0
QnCLF
BSrc[1..0
RST
R1clr
QRight
CLK
DATA[3..0
inst6
mux4
R0reg
R1reg
regbmux
mux2
inst9
Bmux
ALU
alu
ERR_F
AccReg
shift_rec
QReg
Qzero
inst13
inst11
inst10
R1[3..0]
R0[3..0]
R1m
R0m
Stat[1..0]
NFlag
QzSrc
Qz
Qn
OP[2..0]
B_Src[1..0]
N
DAT[0]
Acc[0]
Q[0]
Q[3..0]
Q[3]
R0out[3..0]
R1out[3..0]
Constant Value Generator

```verilog
module Controller_Unit #(parameter W=4)(CLK,
            AccRight, AccParallel, AccCLR, // Acc register control
            ALUCtrl, ASrc, BSrc,  // ALU Controllers
            Stat, NFlag,    // Status bits
            LOAD,R1m, R0m,
            COMP, R1Clr, R1Src, R0WE, R1WE, R0Src,
            OP,  //ALP Operation
                QParallel, QSrc, QnCLR, RST, QRight, QzSrc, Qn, Qz,
            CLR, // reset registers
            ERR // Arithmetic overflow

              );
     input CLK,  CLR;
     input LOAD, COMP;
     input [2:0] OP;
     input [1:0] Stat;
      input R1m, R0m;
     output reg R1Clr=0, R1Src, R0WE=0, R1WE=0;
      output reg [1:0] R0Src, ASrc, BSrc;
      output reg [2:0] ALUCtrl;
     input NFlag , Qn, Qz;
     output reg ERR=0;
     output reg AccRight, AccParallel, AccCLR, QParallel, QSrc, QnCLR=0, RST=0,
     QRight, QzSrc;


      parameter [4:0] ST0 =0, ST1=1, ST2=2, ST3=3, ST4 =4, ST5=5, ST6=6,
                            ST7=7, ST8=8, ST9=9, ST10=10, ST11=11, ST12=12, ST13=13,
                            ST14=14, ST15=15, ST16=16, ST17=17, ST18=18, ST19=19,
                            ST20=20, ST21=21;

      integer k=0, Count=W;
      reg [1:0]r;
      reg NS, CS;

      initial
      begin
      CS = ST0;
      NS = ST0;
      end

      always @(CLK,CLR,LOAD, COMP, OP, Stat,R1m, R0m,NFlag ,r, Count, Qn, Qz)
            begin : COMB

            case(CS)
            ST0: begin

                    if(k>=2 && COMP)
                        case(OP)
                            3'b000 : NS = ST1;
                            3'b001 : NS = ST1;
                            3'b010 : NS = ST2;
                            3'b011 : NS = ST3;
                            3'b100 : NS = ST1;
                            3'b101 : NS = ST1;
                            3'b110 : NS = ST1;
                            3'b111 : NS = ST1;
                        endcase
                    else
                        NS = ST0;
                  end

            ST1: begin
                    NS = ST0;
                  end

            ST2: begin
                    case({Qz,Qn})
                        2'b00 : NS = ST6;

                        2'b01 : NS = ST4;

                        2'b10 : NS = ST5;

                        2'b11 : NS = ST6;

                    endcase
                end

            ST3: begin
                        if(R1m)
                            NS = ST7;
                        else if(R0m)
                            NS = ST9;
                        else if(NFlag)
                            NS = ST12;
                        else
                            NS = ST11;
                end

            ST4: begin
                    NS = ST5;
                end

            ST5: begin
                    NS = ST0;
                end

            ST6: begin
                        if(Count==0)
                            NS = ST0;
                        else
                            case({Qz,Qn})
                                2'b00 : NS = ST6;

                                2'b01 : NS = ST4;

                                2'b10 : NS = ST5;

                                2'b11 : NS = ST6;

                            endcase
                end

            ST7: begin
                    NS = ST8;
                end

            ST8: begin
                        if(R0m)
                            NS = ST9;
                        else if(NFlag)
                            NS = ST12;
                        else
                            NS = ST11;
                end

            ST9: begin
                    NS = ST10;
                end

            ST10: begin
                        if(NFlag)
                            NS = ST12;
                        else
                            NS = ST11;
                 end

            ST11: begin
                    NS = ST13;
                 end

            ST12: begin
                    NS = ST14;
                 end

```

```
        ST13: begin
                    if(NFlag)
                        NS = ST16;
                    else
                        NS = ST15;
               end

        ST14: begin
                    if(NFlag)
                        NS = ST16;
                    else
                        NS = ST15;
               end

        ST15: begin
                    if(Count==0)
                         begin
                            if(NFlag)
                                NS = ST17;
                            else
                                NS = ST0;
                         end
                    else
                            if(NFlag)
                                NS = ST12;
                            else
                                NS = ST11;
               end

        ST16: begin
                    if(Count==0)
                         begin
                            if(NFlag)
                                NS = ST17;
                            else
                                NS = ST0;
                         end
                    else
                            if(NFlag)
                                NS = ST12;
                            else
                                NS = ST11;
               end

        ST17: begin
                    NS = ST18;
                end

        ST18: begin
                    if(r[1])
                        NS = ST19;
                    else if(r[0] ^ r[1])
                        NS = ST21;
                    else
                        NS = ST0;
                end

        ST19: begin
                    NS = ST20;
                end

        ST20: begin
                    if(r[0] ^ r[1])
                        NS = ST21;
                    else
                        NS = ST0;
                end
        ST21: begin
                    NS = ST0;
                end
        endcase
        end
```

```verilog

        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;
```

```
                                R0WE = 1'b1;
                                R1WE = 1'b0;
                                R0Src = 2'b10;
                                r[0] = 1'b1;
                                end
                            else
                                begin
                                AccCLR = 1'b1;
                                AccParallel = 1'b0;
                                QnCLR = 1'b1;
                                QParallel = 1'b1;
                                QSrc = 1'b0;
                                ASrc = 2'b10;
                                Count = W;
                                R0WE = 1'b0;
                                R1WE = 1'b0;
                                end
                        end

                ST4 : begin
                            BSrc = 2'b10;
                            ASrc = 2'b01;
                            ALUCtrl = 3'b000;
                            AccParallel = 1'b1;
                        end

                ST5 : begin
                            BSrc = 2'b10;
                            ASrc = 2'b01;
                            ALUCtrl = 3'b001;
                            AccParallel = 1'b1;
                        end

                ST6 : begin
                            AccParallel = 1'b0;
                            QParallel = 1'b0;
                            AccRight = 1'b1;
                            QRight = 1'b1;
                            Count = Count -1;
                            if(Count==0)
                                begin
                                R0WE = 1'b1;
                                R1WE = 1'b1;
                                R0Src = 2'b01;
                                R1Src = 1'b1;
                                end
                        end

                ST7 : begin
                            BSrc = 2'b00;
                            ASrc = 2'b10;
                            ALUCtrl = 3'b000;
                            R0WE = 1'b0;
                            R1WE = 1'b1;
                            R1Src = 1'b0;
                        end

                ST8 : begin
                            if(R0m)
                                begin
                                BSrc = 2'b11;
                                ASrc = 2'b00;
                                ALUCtrl = 3'b110;
                                R0WE = 1'b1;
                                R1WE = 1'b0;
                                R0Src = 2'b10;
                                r[0] = 1'b1;
                                end
                            else
                                begin
                                AccCLR = 1'b1;
                                AccParallel = 1'b0;
                                QnCLR = 1'b1;
```

```verilog
                QParallel = 1'b1;
                QSrc = 1'b0;
                ASrc = 2'b10;
                Count = W;
                R0WE = 1'b0;
                R1WE = 1'b0;
                end
        end

ST9 : begin
            BSrc = 2'b00;
            ASrc = 2'b11;
            ALUCtrl = 3'b000;
            R0WE = 1'b1;
            R1WE = 1'b0;
            R0Src = 2'b10;
        end

ST10 : begin
            AccCLR = 1'b1;
            AccParallel = 1'b0;
            QnCLR = 1'b1;
            QParallel = 1'b1;
            QSrc = 1'b0;
            ASrc = 2'b10;
            Count = W;
            R0WE = 1'b0;
            R1WE = 1'b0;
         end

ST11 : begin
            AccParallel = 1'b0;
            QParallel = 1'b0;
            AccRight = 1'b0;
            QRight = 1'b0;
         end

ST12 : begin
            AccParallel = 1'b0;
            QParallel = 1'b0;
            AccRight = 1'b0;
            QRight = 1'b0;
         end

ST13 : begin
            BSrc = 2'b11;
            ASrc = 2'b01;
            ALUCtrl = 3'b001;
            AccParallel = 1'b1;
         end

ST14 : begin
            BSrc = 2'b11;
            ASrc = 2'b01;
            ALUCtrl = 3'b000;
            AccParallel = 1'b1;
         end

ST15 : begin
            QSrc = 1'b1;
            QzSrc = 1'b1;
            QParallel = 1'b1;
            Count = Count - 1;
            if((Count == 0) && NFlag)
                begin
                R0WE = 1'b1;
                R1WE = 1'b1;
                R0Src = 2'b01;
                R1Src = 1'b1;
                end
         end

ST16 : begin
```

```
                                    QSrc = 1'b1;
                                    QzSrc = 1'b0;
                                    QParallel = 1'b1;
                                    Count = Count - 1;
                                    if((Count == 0) && NFlag)
                                        begin
                                        R0WE = 1'b1;
                                        R1WE = 1'b1;
                                        R0Src = 2'b01;
                                        R1Src = 1'b1;
                                        end
                                 end

                        ST17 : begin
                                    BSrc = 2'b11;
                                    ASrc = 2'b01;
                                    ALUCtrl = 3'b000;
                                    AccParallel = 1'b1;

                                    R0WE = 1'b1;
                                    R1WE = 1'b1;
                                    R0Src = 2'b01;
                                    R1Src = 1'b1;
                                 end
                    endcase
                    end
                    end

endmodule
```