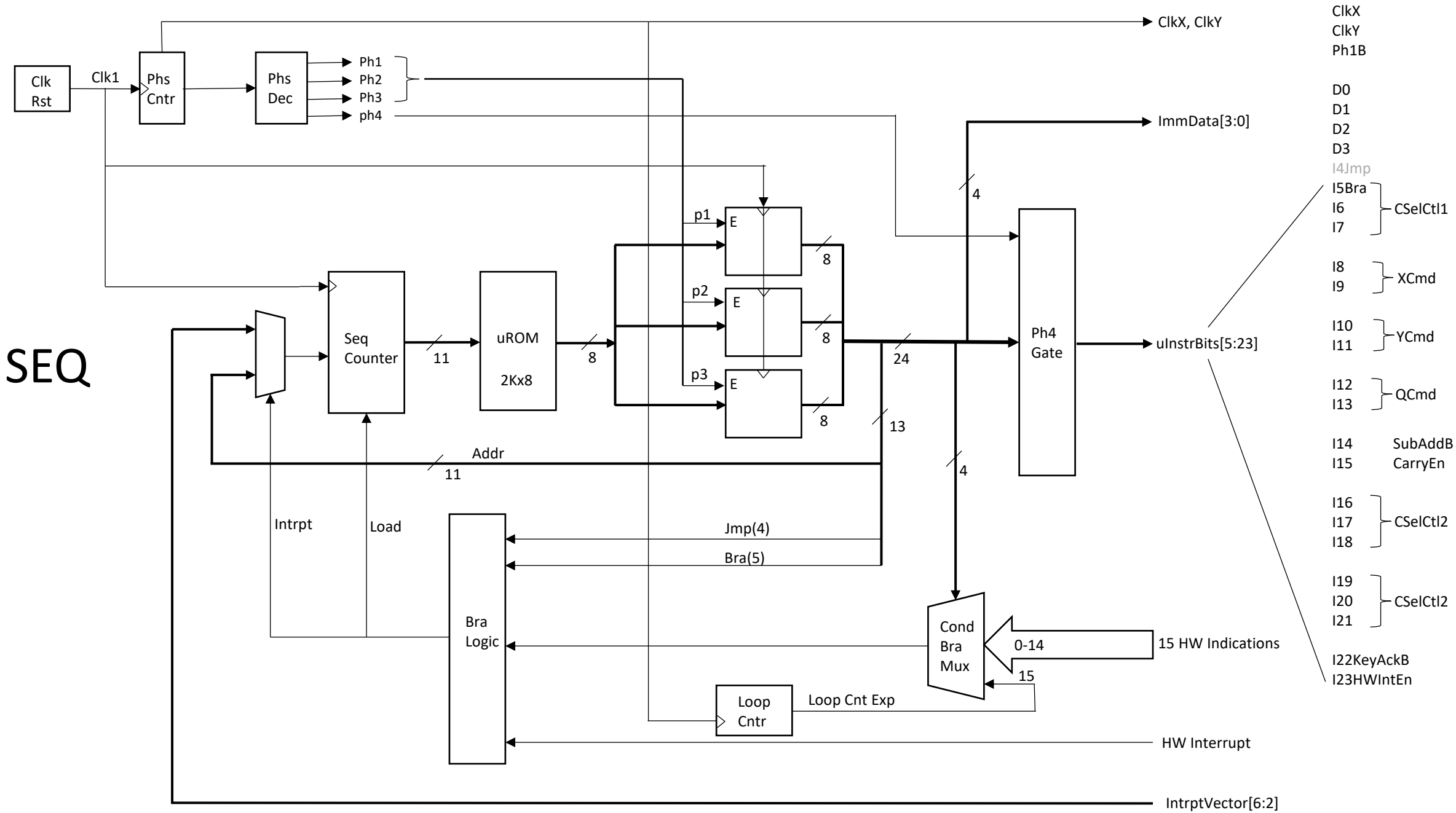
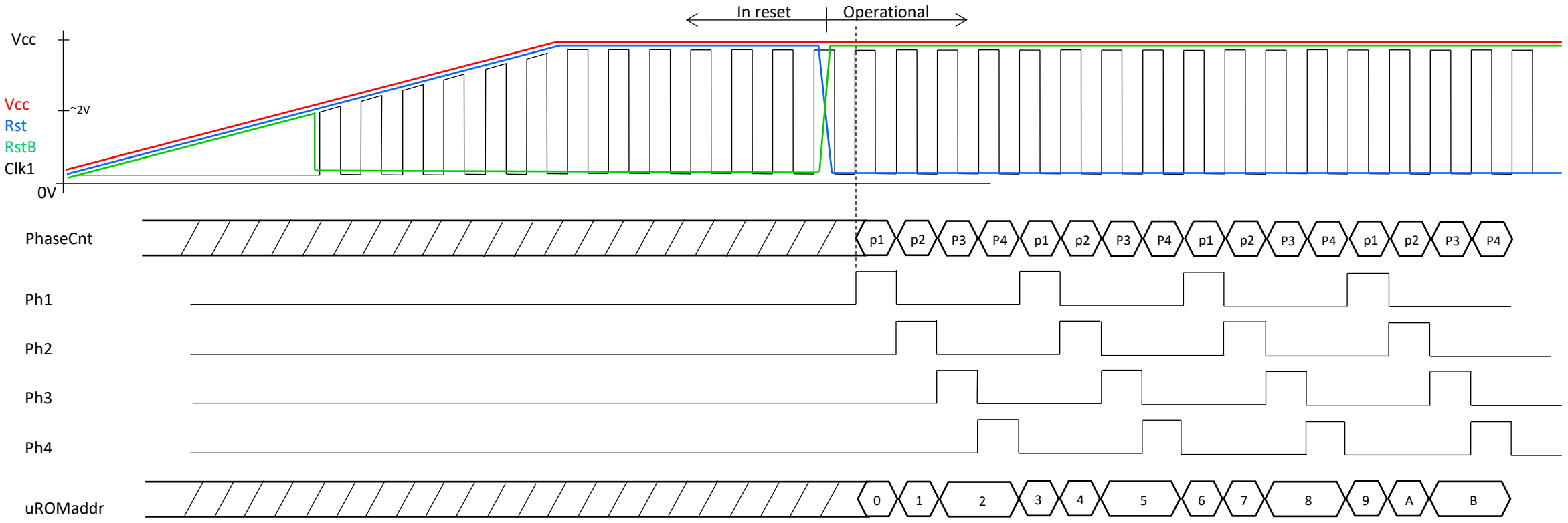
TTL 4-Banger Calculator

- Add, subtract, multiply, and divide
- 8 numeric digits plus an Error and sign digit
  - Max num: +99,999,999 to -99,999,999
  - Min Num: +0.0000001 to -0.0000001
- Floating decimal point
- Signals error on overflow for all operations, keyboard input overflow, and divide by 0
- Leading zero blanking
- All 74hcXX logic implementation (plus 2 4XXX) chips
- 3 major logic blocks:
  1. SEQ: Micro-sequencer engine with a 2816 EEPROM microprogram store
     - 24-bit micro-instruction word
     - Unconditional branch, 16 conditional branches, 1 NMI-type interrupt
     - Single level loop counter (no nesting)
     - Clock and reset generation
  2. DSKY: Keyboard and Display unit
     - 9-digit LED display sequencing, 20-key keyboard encoding (3 keys not used)
  3. REG: Register-Arithmetic unit
     - 1 digit BCD adder/subtractor
     - 3, 8-digit registers (4-bits X 8)
     - Exponent calculator (counters), 8 status and control register bits, operation progress tracking state machine

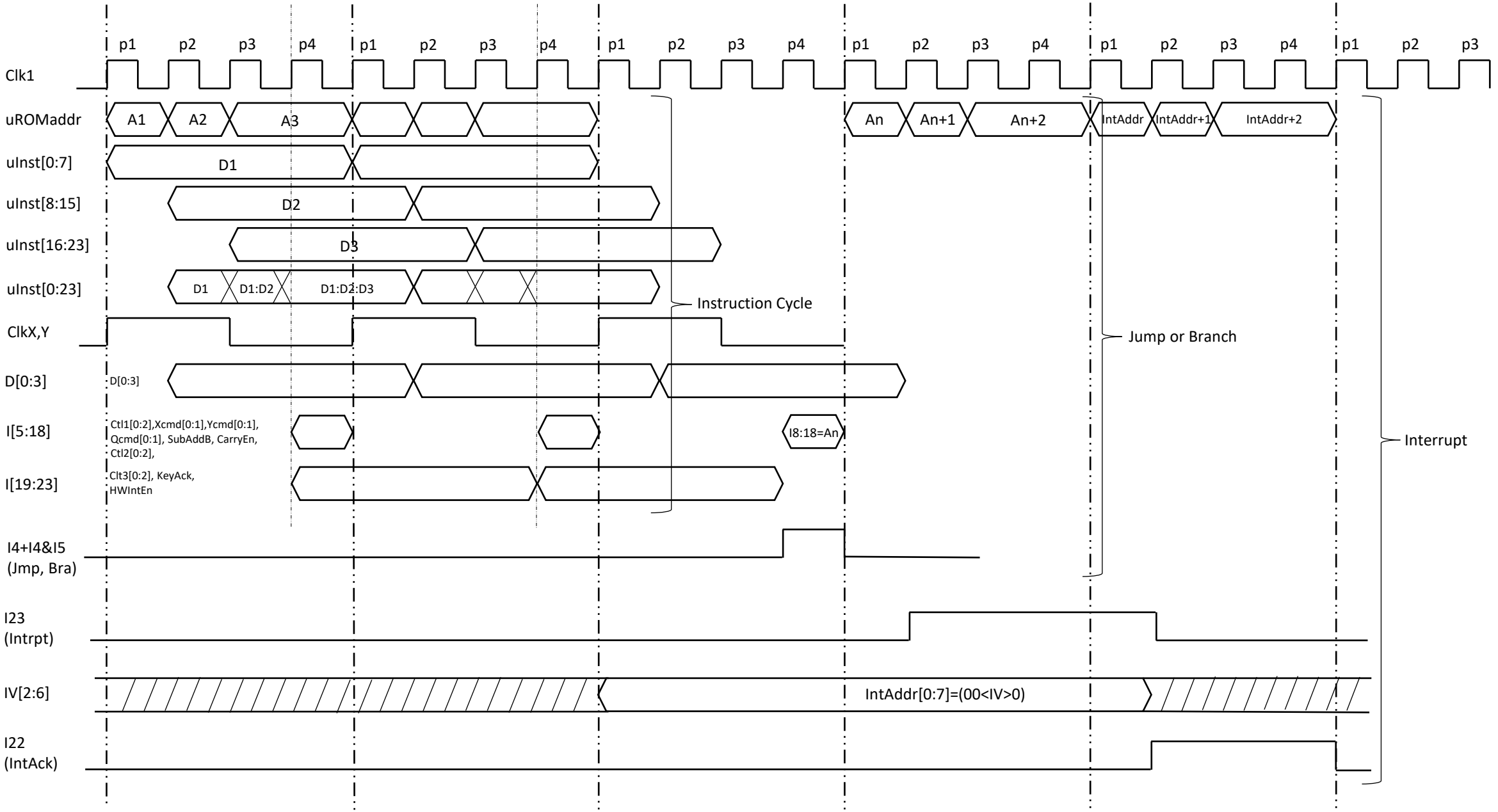
- Architectural Deficits (intentional)
  - No rounding logic
  - No guard digits
- Architectural Bugs (unintentional)
  - Accuracy loss dividing 2 8-digit integers
  - Underflow condition in addition not accounted for


SEQ
Clk
Rst
Clk1
Phs
Cntr
Phs
Dec
Ph1
Ph2
Ph3
ph4
ClkX, ClkY
ImmData[3:0]
4
p1
p2
p3
E
E
E
8
8
8
Seq
Counter
11
uROM
2Kx8
8
24
13
Ph4
Gate
uInstrBits[5:23]
Addr
11
4
Intrpt
Load
Jmp(4)
Bra(5)
Bra
Logic
Cond
Bra
Mux
0-14
15
15 HW Indications
Loop
Cntr
Loop Cnt Exp
HW Interrupt
IntrptVector[6:2]
ClkX
ClkY
Ph1B
D0
D1
D2
D3
I4Jmp
I5Bra
I6
I7
CSelCtl1
I8
I9
XCmd
I10
I11
YCmd
I12
I13
QCmd
I14
I15
SubAddB
CarryEn
I16
I17
I18
CSelCtl2
I19
I20
I21
CSelCtl2
I22KeyAckB
I23HWIntEn

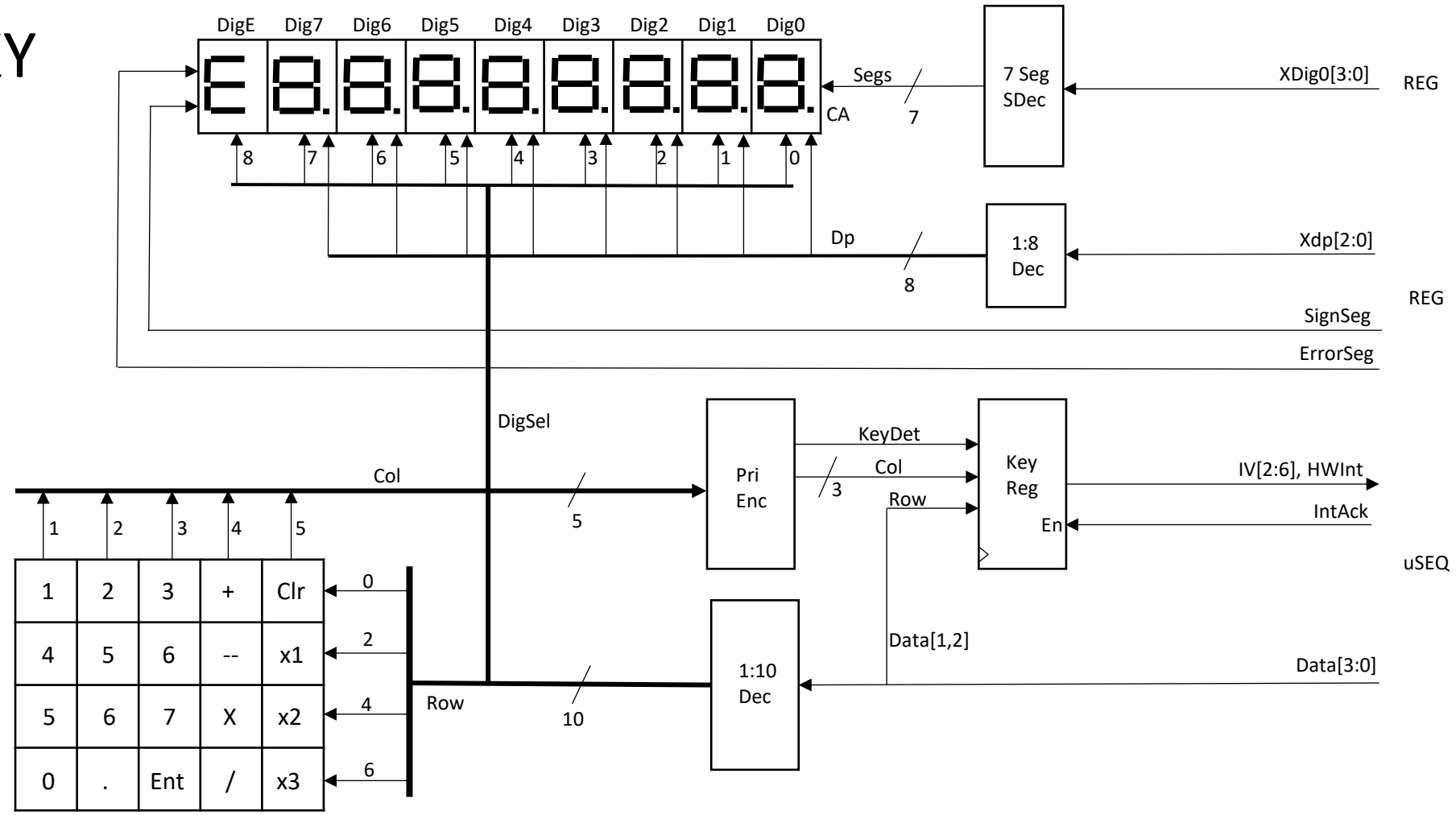
Power-On Reset Startup



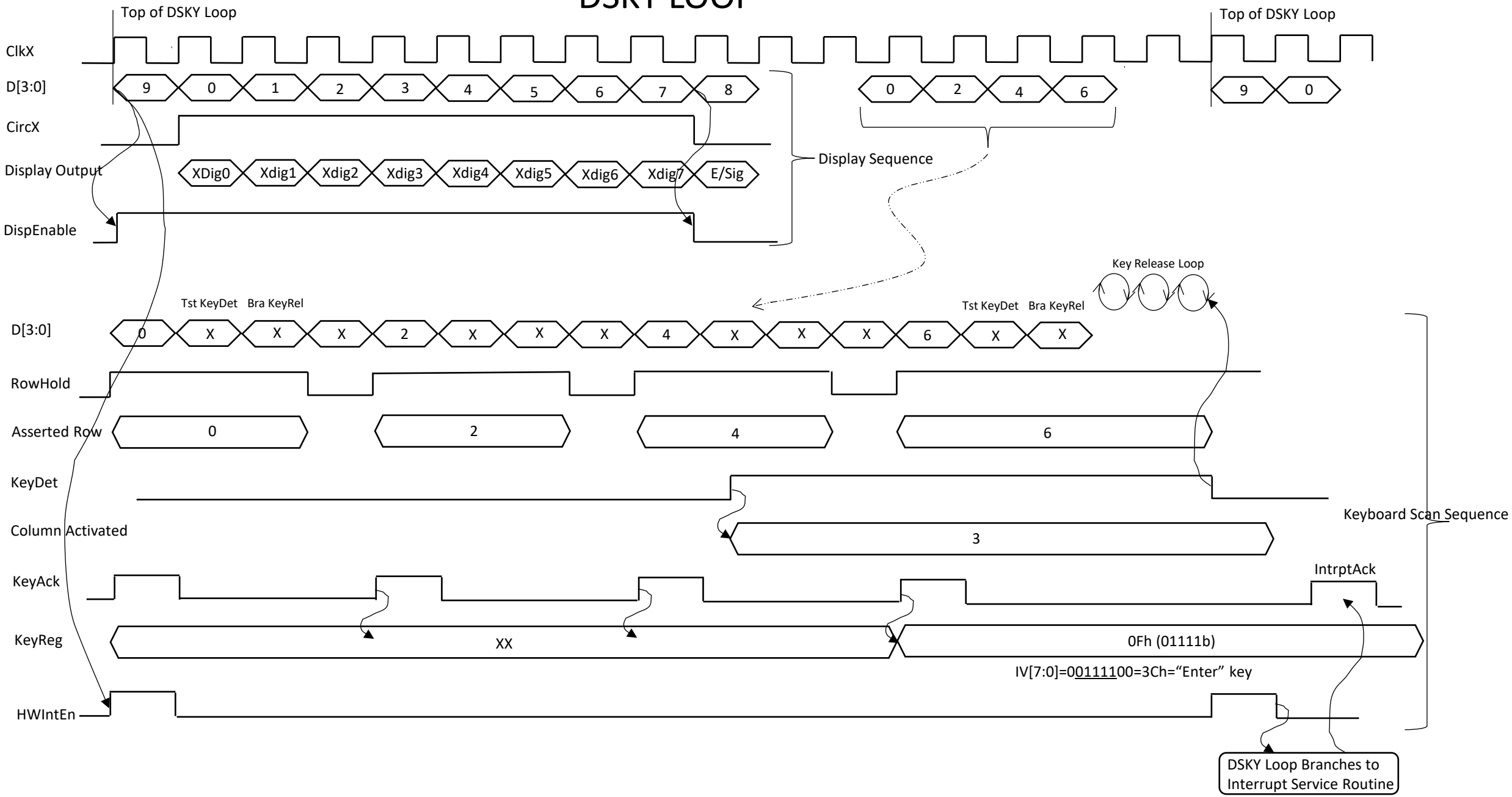
Sequencer Instruction Timing

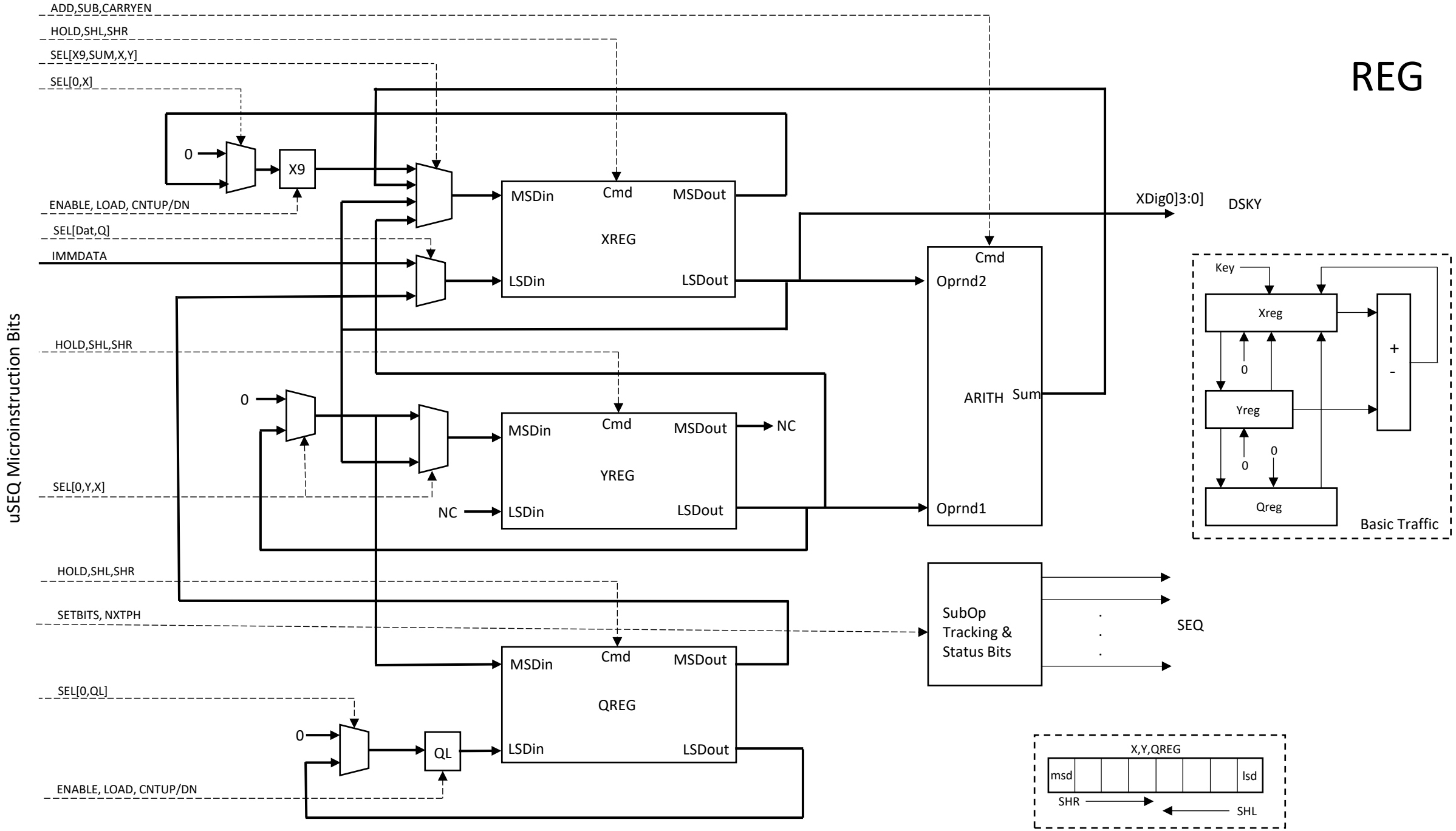


DSKY

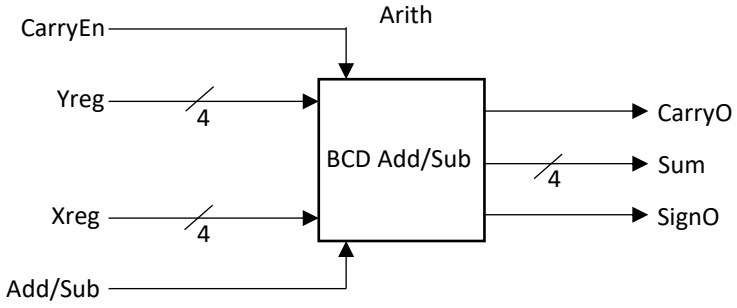

DigE
Dig7
Dig6
Dig5
Dig4
Dig3
Dig2
Dig1
Dig0
Segs
7
CA
7 Seg
SDec
XDig0[3:0]
REG
8
7
6
5
4
3
2
1
0
Dp
8
1:8
Dec
Xdp[2:0]
REG
SignSeg
ErrorSeg
DigSel
KeyDet
Col
3
Pri
Enc
Key
Reg
IV[2:6], HWInt
IntAck
En
Row
Col
5
1
2
3
4
5
uSEQ
1
2
3
+
Clr
4
5
6
--
x1
5
6
7
X
x2
0
.
Ent
/
x3
0
2
4
6
Row
10
1:10
Dec
Data[1,2]
Data[3:0]

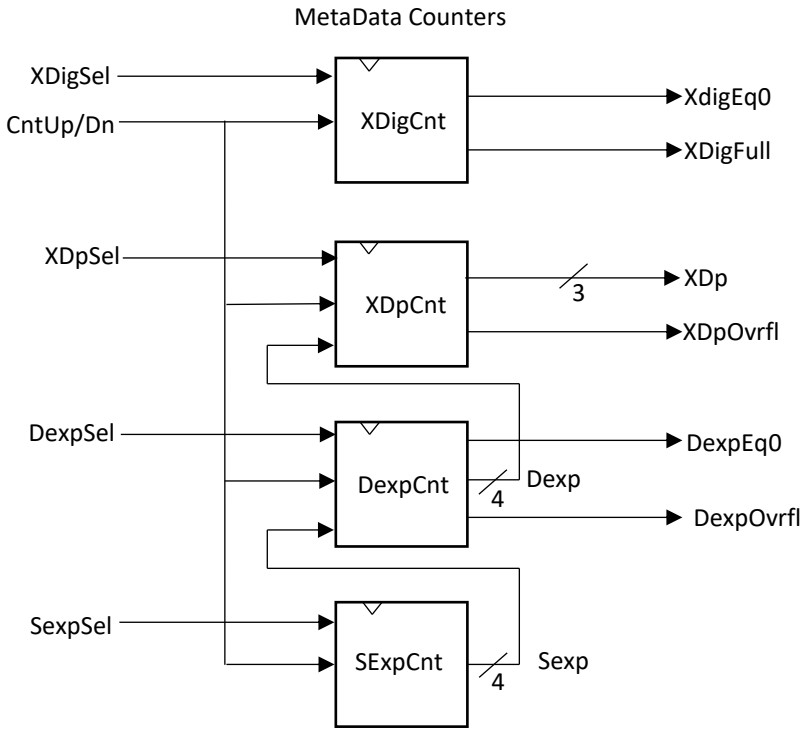
DSKY LOOP

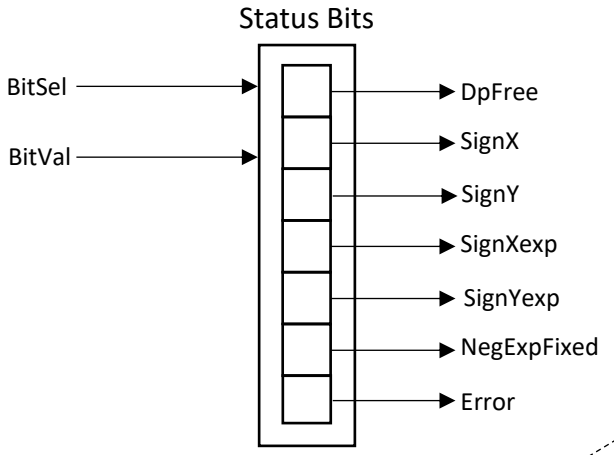

Top of DSKY Loop
Top of DSKY Loop
ClkX
D[3:0]
9 0 1 2 3 4 5 6 7 8
0 2 4 6
9 0
CircX
Display Output
XDig0 Xdig1 Xdig2 Xdig3 Xdig4 Xdig5 Xdig6 Xdig7 E/Sig
Display Sequence
DispEnable
Key Release Loop
Tst KeyDet Bra KeyRel
Tst KeyDet Bra KeyRel
D[3:0]
0 X X X 2 X X X 4 X X X 6 X X
RowHold
Asserted Row
0 2 4 6
KeyDet
Column Activated
3
Keyboard Scan Sequence
KeyAck
IntrptAck
KeyReg
XX
0Fh (01111b)
IV[7:0]=00111100=3Ch="Enter" key
HWIntEn
DSKY Loop Branches to Interrupt Service Routine

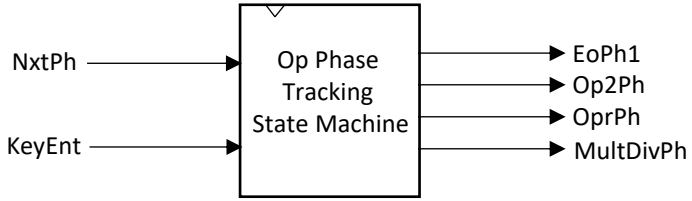

REG
ADD,SUB,CARRYEN
HOLD,SHL,SHR
SEL[X9,SUM,X,Y]
SEL[0,X]
0
X9
ENABLE, LOAD, CNTUP/DN
SEL[Dat,Q]
IMMDATA
MSDin
Cmd
MSDout
XREG
LSDin
LSDout
XDig0]3:0]
DSKY
Cmd
Oprnd2
ARITH
Sum
Oprnd1
HOLD,SHL,SHR
0
MSDin
Cmd
MSDout
NC
YREG
NC
LSDin
LSDout
SEL[0,Y,X]
uSEQ Microinstruction Bits
HOLD,SHL,SHR
SETBITS, NXTPH
SubOp
Tracking &
Status Bits
SEQ
SEL[0,QL]
0
QL
MSDin
Cmd
MSDout
QREG
LSDin
LSDout
ENABLE, LOAD, CNTUP/DN
Key
Xreg
0
Yreg
0
0
Qreg
+
-
Basic Traffic
X,Y,QREG
msd
lsd
SHR
SHL

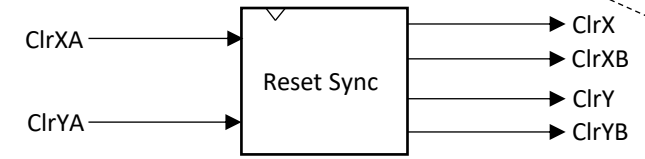
ArithStatPhs

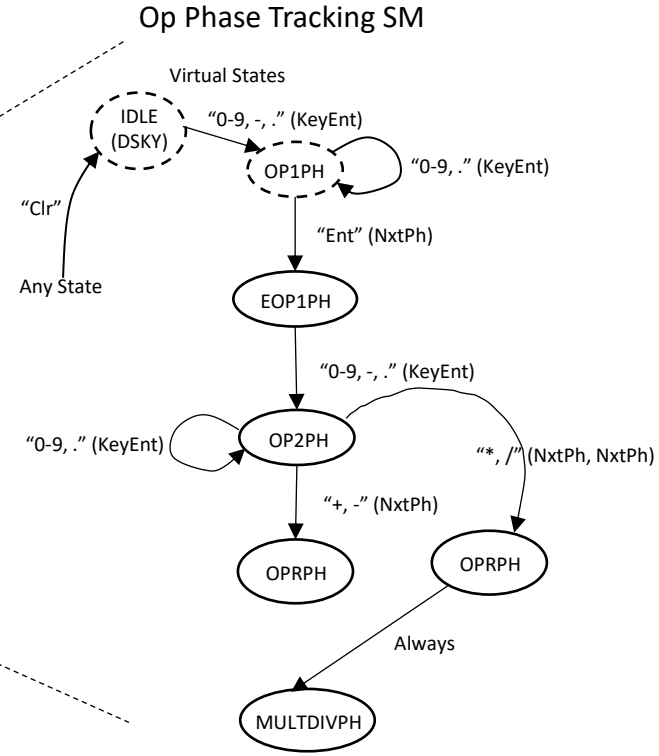












BCD Arithmetic

A, B={0..9] BCD/Hex
S=[0..18]BCD, [0..12]Hex

```
  7
+ 5
-----
0xC       If S>9 then
+ 6           Add 6 to S
-----
12 bcd
```

	Co		Ci	Co		Ci	
46		0100			0110		addend
+55		0101			0101		addend
101		1001			1011		sum
		0000			0110		bdc correction
	1	0000	1	1	0001		

For a decimal digit Z:
The 9's complement (9-Z) on the decimal number wheel is the same as comp(Z_{bcd})+0xA on the hex number wheel

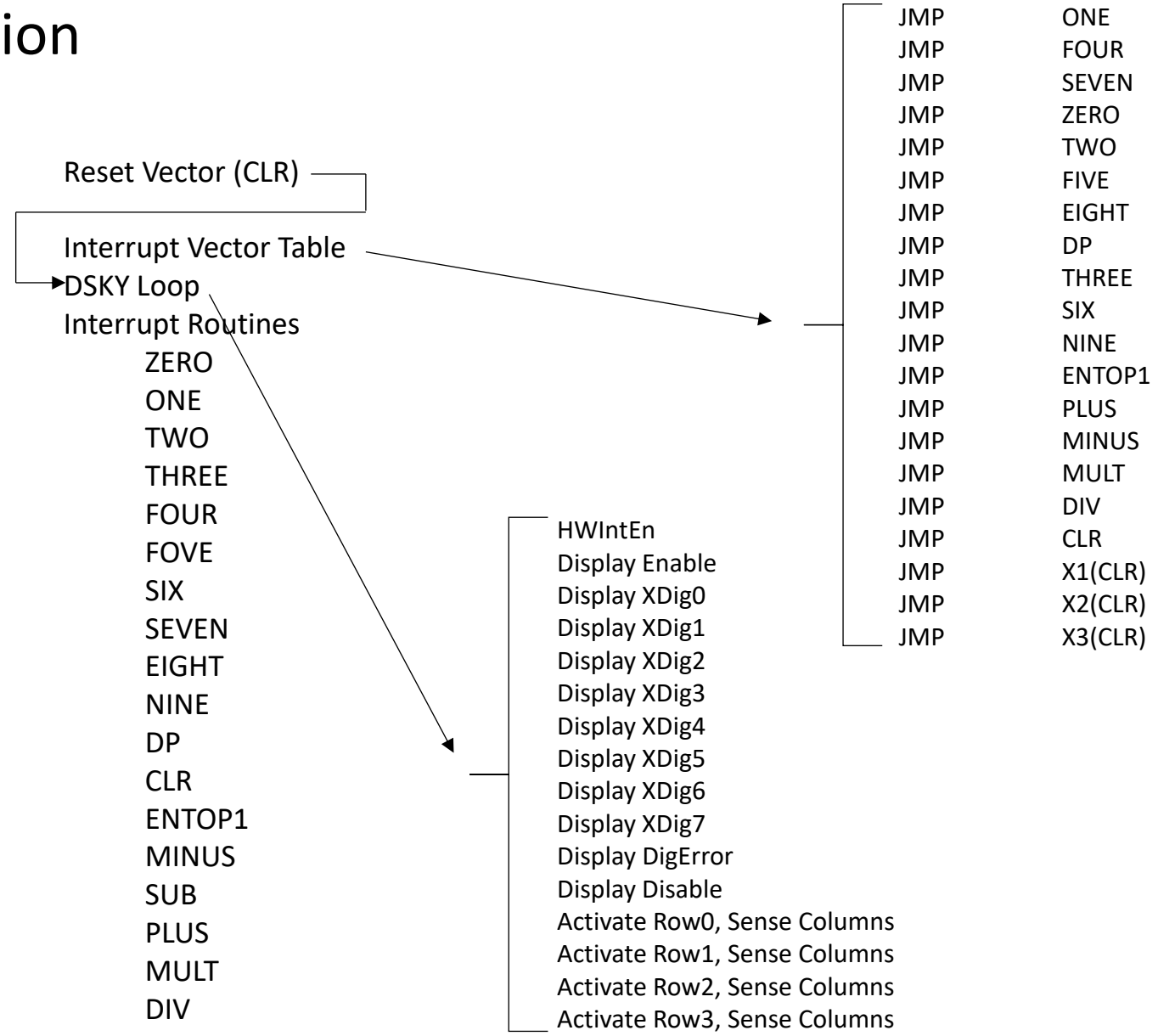
The 10's complement of a multi-digit number is the 9's complement of each digit with 1 added-with-carry to the multi-digit result

A-B
Compute 10's complement of B {
Take 1's complement of each digit of B
Add b'1010 to each digit of B
Add 1 B (add 1 to digit 1 and propagate carry)
}
Add A
If final carry out then difference is positive
Discard carry out
Else difference is negative
Take 10's complement

	Co		Ci	Co		Ci	
46		0100			0110		minuend
-28		0010			1000		subtrahend
18							
		1101			0111		1's complement of subtrahend
		1010			1010		add 10
	1	0111		1	0001		9's complement of subtrahend
		0100			0110	1	add minuend + 1
							if result digit > 9
		1011			1000		then
		0110			0000		add 6 else add 0
	1	0001			1000		Final carry=1 sum is pos

	Co		Ci	Co		Ci	
57		0101			0111		
-78		0111			1000		
-21							
		1000			0111		1's complement of subtrahend
		1010			1010		add 10
	1	0010		1	0001		9's complement
		0101			0111	1	add minuend + 1
							Final carry is 0 so difference is
	0	0111			1001		neg
		1000			0110		1's complement
		1010			1010		add 10
	1	0010			0000		9's complement
		0000			0000	1	add 1
		0010			0001		

MicroCode Organization



MicroCode Structure and Syntax Example

			I[5:0], I23		I[3:0]	I[22]	I[13:8]	I[7:5]	I[18:16]	I[21:19]	I[3:0]	I[14]	I[15]	Logisim uROM Image			
			1[5:0], 3[7]	2[7:0], 3[2:0]	1[3:0]	3[6]	2[5:0]	1[7:5]	3[2:0]	3[5:3]	1[3:0]	2[6]	2[7]	D[7:0]	I[15:8]	I[23:16]	
Addr	**Hex**	**Label**	**SEQ**		**DSKY**		**REG**					**ALU**		**1[7:0]**	**2[7:0]**	**3[7:0]**	**4**
			Bra	**Target**	**Sel[3:0]**	**KeyAck**	**RegCmd**	**SelCtl1**	**SelCtl2**	**SelCtl3**	**KeyMCntSel**	**Sub_AddB**	**CarryEn**	v2.0 raw			
507	1FB	**ENTOP1**	BRSignXexp	SETOP1DPFIR										3E	01	02	If Op1 has neg exponent (DPFirst) then save in S
510	1FE		JMP	TSTDIGEMPTY										10	04	02	otherwise goto test for empty X reg
513	201	SETOP1DPFIR							StatBitEn	StatBit1	SignYexp			05	00	1B	
516	204	TSTDIGEMPTY	BRDigEmpty	ENTERZERO										32	0D	02	If no digit hit yet, Jump to enter0
519	207		BRDPFree	ADJEXPFIXDP										37	16	02	If DP was never hit then goto adjust exp and fix
522	20A		JMP	LEFTJUST										10	1F	02	Otherwise go to left justify
525	20D	ENTERZERO							CntrEn	CntrIncr	XDigCnt			01	00	36	Incr Xreg Dig count (enter a 0)
528	210								StatBitEn	StatBit1	FixDP			00	00	1B	and fix the DP
531	213		JMP	LEFTJUST										10	1F	02	then left justify
534	216	ADJEXPFIXDP							CntrEn		XYSexpCnt			04	00	06	If X is whole integer, fix DP and adjust exponen
537	219								CntrEn		XYDexpCnt			03	00	06	
540	21C								StatBitEn	StatBit1	FixDP			00	00	1B	
543	21F	LEFTJUST	BRDigFull	COPYXY										31	2E	02	Left justify X until Xdig is full
546	222						LeftJustX	SelXData			0			00	02	00	Shift left with 0's in at msd
549	225								CntrEn	CntrIncr	XDigCnt			01	00	36	and adjust Xdp and Xdig count to match
552	228								CntrEn	CntrIncr	XDPCnt			02	00	36	
555	22B		JMP	LEFTJUST										10	1F	02	
558	22E	COPYXY					CopyXY	SelXReg	SelXXlsd	SelYXlsd				20	05	28	Copy Xreg to Yreg
561	231						CopyXY	SelXReg	SelXXlsd	SelYXlsd				20	05	28	
564	234						CopyXY	SelXReg	SelXXlsd	SelYXlsd				20	05	28	
567	237						CopyXY	SelXReg	SelXXlsd	SelYXlsd				20	05	28	
570	23A						CopyXY	SelXReg	SelXXlsd	SelYXlsd				20	05	28	
573	23D						CopyXY	SelXReg	SelXXlsd	SelYXlsd				20	05	28	
576	240						CopyXY	SelXReg	SelXXlsd	SelYXlsd				20	05	28	
579	243						CopyXY	SelXReg	SelXXlsd	SelYXlsd				20	05	28	
582	246		BRSignX	CPSIGXSIGY										34	4C	02	Copy Xsign to Ysign
585	249		JMP	RMLDZEROS										10	4F	02	
588	24C	CPSIGXSIGY							StatBitEn	StatBit1	SignY			04	00	1B	

MicroInstruction Fields

SelCtl1 (I5br:I7)

SelXReg	001	Select X reg MSD source from X reg or Y reg group
LoadX9	010	Load X9 source into X9 reg
SelXQmsd	011	Select X reg source from Q Dig7
LoadQL	100	Load QL source into QL reg
SHDexpXDP	101	Load Dexp counter value into XDP counter
Reserved1	110	Currently unassigned
Reserved2	111	Currently unassigned
SelXData	000	Select X LSD reg source from D[3:0]
SelXSumX9	000	Select X reg MSD source from Sum or X9 group

SelCtl2 (I16:I18)

SelQLQlsd	001	Select QL reg source from Q reg LSD
SelYQYlsd	010	Select Y and Q reg MSD sources from Y reg LSD
StatBit1	011	Set selected status bit to 1
SelX9Xmsd	100	Select X9 reg source from X reg MSD
SelYXlsd	101	Select Y reg source from X reg LSD
CntrIncr	110	Increment selected and enabled counter
RowHold	111	DSKY row latch disable (hold)
SelYZero	000	Select Y reg MSD source from 0
SelQZero	000	Select Q reg MSD source from 0
SelX9Zero	000	Select X9 reg source from 0
SelQLZero	000	Select QL reg source from 0

SelCtl3 (I19:I21)

ClrXPh1	001	Clear X reg domain devices
ClrYPh2	010	Clear Y reg domain devices
StatBitEn	011	Enable status latches for write
SelXYlsd	100	Select X reg MSD source from Y reg LSD
SelXSum	100	Select X reg MSD source from SUM
NxtPh	101	Advance tracking state machine state
CntrEn	110	Enable selected counter for up/down advance
SHSexpDexp	111	Load Sexp counter value into Dexp
SelXX9	000	Select X reg MSD source from X9 reg
SelXXlsd	000	Select X reg MSD source from X reg LSD

RegCmd (I8:I13)

CircX	000001	Shift X reg from MSD toward LSD
DPAdjX	000001	
ShrX	000001	
EntNumX	000010	Shift X reg from LSD toward MSD
LeftJustX	000010	
ApendDivdn	000010	
RmLdZeroX	000010	
DPAdjY	000100	Shift X reg from MSD toward LSD
CopyXY	000101	Shift X and Y reg from MSD toward LSD
ExchXY	000101	
AddSub	000101	
DivXY10	000101	
ShrX9X8QDiv10	010001	Shift X and Q reg from MSD toward LSD
CopyYQ	010100	Shift Y and Q reg from MSD toward LSD
CopyQ0QL	100000	Shift Q reg from LSD toward MSD
CopyQX	100010	Shift X and Q reg from LSD toward MSD

Branches (D3:D0)

JMP	NA	Unconditional Branch
BRDecEven	0000	Branch if DexpCnt=0
BRDigFull	0001	Branch of DigCnt=8
BRDigEmpty	0010	Branch if DigCnt=0
BRDintOvrfl	0011	Branch if Dexp>=8
BRSignX	0100	Branch if SignX status bit set
BRSignY	0101	Branch if SignY status bit set
BRInOp2Ph	0110	Branch if in or beyond Op2Ph tracking state
BRDPFree	0111	Branch of DP status bit has not been set
BRQLZero	1000	Branch if Qlreg=0
BRX9Zero	1001	Branch if X9reg=0
BRDpOvrfl	1010	Branch if XDpCnt>=8
BRNoKeyDet	1011	Branch if no key is currently pressed
BRIsNeg	1100	Branch if arithmetic (SUM) sign bit is set
BRCarryO	1101	Branch if arithmetic carry out bit is set
BRSignXexp	1110	Branch if Op2's exponent sign status bit is set
BRLoop8NZ	1111	Branch if LoopCnt>0
BRSignYexp	0110	Branch if Op1's exponent sign status bit is set
BrNegExpTerm	0101	Branch if current negative exponent calculated

Immediate Data Labels (D3:D0)

0	0000	Immedate data 0
1	0001	Immedate data 1
2	0010	Immedate data 2
3	0011	Immedate data 3
4	0100	Immedate data 4
5	0101	Immedate data 5
6	0110	Immedate data 6
7	0111	Immedate data 7
8	1000	Immedate data 8
9	1001	Immedate data 9
XDigCnt	0001	Select the XDigit counter
XDPCnt	0010	Select the XDP counter
XYDexpCnt	0011	Select the Dexp counter
XYSexpCnt	0100	Select the Sexp counter
X9Cnt	0101	Select the X9 counter
QLCnt	0110	Select the QL counter
FixDP	0000	Select the DP status bit
SignXexp	0001	Select the Op2 exponent sign status bit
NegExpTerm	0010	Select the negative exponent accumulation terminated status bit
SignX	0011	Select the Op2 sign status bit
SignY	0100	Select the Op1 sign status bit
SignYexp	0101	Select the Op1 exponent sign status bit
AssertError	0110	Select the Error status bit

Add/Sub (I14:I15)

SubAddB	0	Add Digit 0 of Y register with Digit 0 of X register
	1	Subtract Digit 0 of X register from Digit 0 of Y register
CarryEn	0	Disable Carry In to Adder/Subtractor for one instruction
	1	Enable Carry In to Adder/Subtractor for one instruction

SEQ/DSKY (I22:I23)

KeyAck	0	NA
	1	Write state of keyboard into keyboard register, reset sequencer loop counter
HWIntEn	0	NA
	1	Enable branch to interrupt service routine if HWInt is active for one instruction

1-9 Service Routine

```
SHL <num> into Xreg
Incr XdigCnt
If Dp_Not_Fixed then
   Incr SexpCnt
   If Op1Ph then
      Incr DexpCnt
   Else
      Decr DexpCnt
Else          // Dp Fixed.  Key hits no longer accumulate exponents
   Incr XDpCnt
   Set NegExpTerm  // Indicate negative exponent accumulation terminated
```

Dp Service Routine

```
Set FixDp
If Dp is first key hit then
   Set SignXexp
   SHL "0" into Xreg
   Incr XDigCnt
Decr SexpCnt
If Op1Ph then
   Decr DexpCnt
Else
   Incr DexpCnt
JMP DSKY
```

Zero Service Routine

```
JMP RESET
```

Zero Service Routine

```
SHL "0" int Xreg
If Not SignXexp then    // If Xreg is > 1
    If XDigEmpty then    // And Xreg is empty
       JMP DSKY          // Then key hit can be ignored
   Else                 // Building operand > 1
      Incr XdigCnt
      If DpFixed Then    // Building the fractional part
         Incr XDp Cnt
         JMP DSKY
      Else               // Building the integer part
         Incr SexpCnt
         If Op1Ph Then
            Incr DexpCnt
         Else
            Decr DexpCnt
         JMP DSKY
Else                     // Building operand < 1
   Incr XdigCnt
   Incr XDpCnt
   If NegExpTerm Then    //  If no more leading 0's then done
      JMP DSKY
   Else                  // Account for the larger negative exponent
     Decr SexpCnt
   If Op1Ph Then
      Decr DexpCnt
   Else
      Incr DexpCnt
      Incr QL      // QL used to mirror DexpCnt count of leading 0's (used by Div)
  JMP DSKY
```

Enter Op1 Service Routine

```
If Dp was first key hit then  // Xreg < than 1
   Set SignYexp bit
If no numeric key was hit then
   Incr XDigCnt;  Set FixDp
Else
   If Dp was not hit then    // Xreg is pure integer
      Ser FixDp;  Decr Dexp, Sexp
Left_Justify_Xreg
Copy_Xreg_Into_Yreg
Copy SignX into SignY
Remove_Leading_Zeros   // Does nothing unless operand is < 1
                  // The operand in X is mantissa
                  // The operand in Y is displayable version
Exchange X<>Y
NxtPh          // Move to EOp1Ph
Clear X9
JMP DSKY
```

Add and Subtract Service Routine

MINUS

```
If Not Op2Ph Then     // If not in Op2 then "-" means sign bit
   Set SignX
   JMP DSKY
Else                  // If in Op2 then "-" means subtract
   Invert_SignX
```

PLUS

```
If DigEmpty Then      // If no key hit for Op2 then enter "0"
   Incr XDigCnt
   Set FixDp
Else
   If DpFree Then     // Op2 is pure integer, fix Dp and adj exp
      Set FixDp
      Decr SexpCnt
      Incr DexpCnt
Left_Justify_Xreg
While X8 == 0         //  Remove leading 0's from Op2<1
   SHL Xreg, 0
   Incr XDpCnt
While DexpCnt != 0    //  Get operands exponent-aligned (DpAdj)
   If DintOvrfl Then  // Shift Yreg right, count Dexp up
      SHR Yreg, 0
      Incr DexpCnt
   Else               // Shift Xreg right, count Dexp down
      SHR Xreg, 0
      Decr DexpCnt
While DpOvrfl         //  Restore leading 0's to Op2, if any
   SHR Xreg; SHR Yreg
   Decr XDpCnt
If (SignX XOR SignY) == 0 Then   // If Op signs same, then ADD
   Xreg = Xreg + Yreg
   If CarryO Then     // If final carry out then shift in X9, adj XDp
      Incr X9
      SHR Xreg, X9
      Decr XDpCnt
Else                  // Op signs are different
   If (Sign X == 1 AND SignY == 0) Then   // Yreg must be the negative op for ALU
      Exch Xreg<>Yreg
      Exch SignX<>SignY
   Xreg = Xreg – Yreg
If DpOvrfl Then
   Set Error
JMP DSKY
```

OPADD

```
SHR Xreg, Sum; SHR Yreg, 0; ADD            // Shift Xreg and Yreg right,
SHR Xreg, Sum; SHR Yreg, 0; ADD, CarryEn   // with Lsd's going through ALU
SHR Xreg, Sum; SHR Yreg, 0; ADD, CarryEn   // and the Sum shifting into
SHR Xreg, Sum; SHR Yreg, 0; ADD, CarryEn   // Xreg msd, and 0's shifting
SHR Xreg, Sum; SHR Yreg, 0; ADD, CarryEn   // into Yreg msd
SHR Xreg, Sum; SHR Yreg, 0; ADD, CarryEn
SHR Xreg, Sum; SHR Yreg, 0; ADD, CarryEn
SHR Xreg, Sum; SHR Yreg, 0; ADD, CarryEn
```

OPSUB

```
SHR Xreg, Sum; SHR Yreg, 0; SUB
SHR Xreg, Sum; SHR Yreg, 0; SUB, CarryEn
SHR Xreg, Sum; SHR Yreg, 0; SUB, CarryEn
SHR Xreg, Sum; SHR Yreg, 0; SUB, CarryEn
SHR Xreg, Sum; SHR Yreg, 0; SUB, CarryEn
SHR Xreg, Sum; SHR Yreg, 0; SUB, CarryEn
SHR Xreg, Sum; SHR Yreg, 0; SUB, CarryEn
SHR Xreg, Sum; SHR Yreg, 0; SUB, CarryEn
If IsNeg Then         //  Get 10's complement and BCD adjust
   Exch Xreg<>Yreg
   SHR Xreg, Sum; SHR Yreg, 0; SUB
   SHR Xreg, Sum; SHR Yreg, 0; SUB, CarryEn
   SHR Xreg, Sum; SHR Yreg, 0; SUB, CarryEn
   SHR Xreg, Sum; SHR Yreg, 0; SUB, CarryEn
   SHR Xreg, Sum; SHR Yreg, 0; SUB, CarryEn
   SHR Xreg, Sum; SHR Yreg, 0; SUB, CarryEn
   SHR Xreg, Sum; SHR Yreg, 0; SUB, CarryEn
   SHR Xreg, Sum; SHR Yreg, 0; SUB, CarryEn
   Set SignX          // Since result is negative
While X8 == 0 AND NOT DpOvrfl   // Remove excess leading 0's, if any
   SHL Xreg, 0
   Incr XDpCnt
```

```
Ex:   150.006
    - 150.04
--------------
```

Mult Service Routine

```
If DigEmpty Then      //  If no key hit for Op2 then enter "0"
   Incr XDigCnt
   Set FixDp
Else
   If DPFree Then    // Else if Dp not hit then FixDp and adj exps
      Set FixDp
      Decr SexpCnt
      Incr DexpCnt
Left_Justify_Xreg
While X8 == 0       // Remove any leading 0's in Op2
   SHL Xreg, 0; Incr XDpCnt
NxtPh                  // Move to OprPh
If (SignX XOR SignY) == 1 Then
   Set SignX
Else
  Clear SignX
Copy Yreg to Qreg   // Get regs set up for Mult loop
Copy Xreg to Yreg, clearing Xreg
Mult_Loop
If (SignXexp == SignYexp == 0 AND DpOvrfl) Then   // Overflow handler
   Set Error
   JMP DSKY
If (SignXexp == SignYexp == 1 AND DpOvrfl) Then   // Underflow handler
   ClrXPh
   JMP DSKY
If DpOvrfl Then     // Product is <1
   While DpOvrfl    // Insert leading 0's to product <1
      SHR Xreg, 0    //   leaving XDP at 0 (7 after inv)
      Incr XDpCnt
Else                   // Product is >1
   While X8 == 0    //  Remove leading 0's to product >1, if any
      SHL Xreg, 0
      Decr XDpCnt   // Move Dp to the left
JMP DSKY
```

```
Copy Q0 into QL; SHR Qreg
NxtPh                          // Move to MultDivPh
Shift SexpCnt into XDpCnt      // SexpCnt has the DP placement for product
For 7 Loops
   While QL != 0
      SHR Xreg, Sum; SHR Yreg, Y0; ADD              // Shift Xreg and Yreg right,
      SHR Xreg, Sum; SHR Yreg, Y0; ADD, CarryEn     //  with Xmsd getting getting the Sum
      SHR Xreg, Sum; SHR Yreg, Y0; ADD, CarryEn     //  and Ymsd circulating Ylsd
      SHR Xreg, Sum; SHR Yreg, Y0; ADD, CarryEn
      SHR Xreg, Sum; SHR Yreg, Y0; ADD, CarryEn
      SHR Xreg, Sum; SHR Yreg, Y0; ADD, CarryEn
      SHR Xreg, Sum; SHR Yreg, Y0; ADD, CarryEn
      SHR Xreg, Sum; SHR Yreg, Y0; ADD, CarryEn
      If CarryO Then
         Incr X9
      Decr QL
   SHR Xreg, X9; SHR Qreg, 0
   Clr X9
   Copy Q0 into QL
   While QL != 0
      SHR Xreg, Sum; SHR Yreg, Y0; ADD              // Shift Xreg and Yreg right,
      SHR Xreg, Sum; SHR Yreg, Y0; ADD, CarryEn     //  with Xmsd getting getting the Sum
      SHR Xreg, Sum; SHR Yreg, Y0; ADD, CarryEn     //  and Ymsd circulating Ylsd
      SHR Xreg, Sum; SHR Yreg, Y0; ADD, CarryEn
      SHR Xreg, Sum; SHR Yreg, Y0; ADD, CarryEn
      SHR Xreg, Sum; SHR Yreg, Y0; ADD, CarryEn
      SHR Xreg, Sum; SHR Yreg, Y0; ADD, CarryEn
      SHR Xreg, Sum; SHR Yreg, Y0; ADD, CarryEn
      If CarryO Then
         Incr X9
      Decr QL
   If X9 != 0 Then
      SHR Xreg, X9
      Incr XDpCnt
```

Div Service Routine

```
   For 8  Loops                 // Check for Op2==0
      Load X8 into X9
      If X9 == 0 Then
         SHR Xreg, X1
      Else Continue A
   Set Error
   JMP DSKY
A:
   If NOT DpFixed Then      // If Dp never entered
      Set FixDp                 //   Fix Dp and exp adj
      Decr SexpCnt
      Incr DexpCnt
      NxtPh                     // Move to OprPh
   If (SignX XOR SignY) == 0 Then    // Compute sign of quotient
      Clear SignX
   Else
     Set SignX
   If Op2 < 1 Then           // Divide both ops by 10 if Xreg is full
      If DigFull Then
         SHR Xreg, 0; SHR Y, 0
   Copy Yreg to Qreg
   Copy Xreg to Yreg, clearing Xreg
   Move to MultDivPh
   Copy DexpCnt to XDpCnt   // Exponent for quotient is DexpCnt
   If SignXexp Then       // Remove the extra "0" in front of Dp
      Decr XDigCnt
      While QL != 0
         Decr QL        // QL is a copy of DexpCnt to use for removing leading 0's
         Decr XDigCnt
   While NOT DigEmpty   // Init dividend with XDigCnt digits of Qreg
      SHL Xreg, Q8; SHL Qreg, 0
      Decr XDigCnt
   X = X-Y                   // Begin test if one more dividend digit is needed
   If IsNeg Then
      X = X+Y    // Restore dividend
      SHL Xreg, Q8; SHL Qreg, 0     // Add one more digit to dividend
      Decr XDpCnt
   Else
     X = X+Y    // Restore dividend
   While (X = X-Y) >= 0              // First of 8 outer loop iterations
      Incr QL   // Accumulate quotient
   X = X+Y     // Restore dividend to positive
   For 7 Loops                       // Remaining 7 outer loop iterations
      SHL Xreg, Q8             // Append dividend with additional digit
      SHL Qreg, 0; Copy QL to Q0     // Accumulate quotent
      Clr QL
      While (X = X-Y) >= 0       // Inner loop
         Incr QL
      X = X+Y
   SHL Xreg, Q8
   SHL Qreg, 0; Copy QL to Q0
   Clr QL
   Copy Qreg to Xreg          // Copy quotient to Xreg
   If (SignYexp == 1 AND SignXexp == 0) Then   // Underflow handler
      If DPOvrfl Then
         Clr Xreg
         JMP DSKY
   If (SignXexp == 1 AND SignYexp) == 0 Then  // Overflow Handler
      If DPOvrfl Then
         Set Error
         JMP DSKY
   If NOT DPOvrfl Then   // Quotient >=1;  Done
      JMP DSKY
   Repeat
      SHR Xreg, 0
      Incr XDpCnt
   Until DPOvrfl
   JMP DSKY
```

9+0.0000001=9.0000001
99999999+.1=99999999
99+0.0000001=99

Dexp Counter values for All Op1 and Op2 Combinations

DPFirst=1

Dintovrfl=1

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

Xexp(+), Yexp=(+)

OP2(X)		10000000	1000000	100000	10000	1000	100	10	1
OP1 (Y)	exp	7	6	5	4	3	2	1	0
10000000	7	0	1	2	3	4	5	6	7
1000000	6	-1	0	1	2	3	4	5	6
100000	5	-2	-1	0	1	2	3	4	5
10000	4	-3	-2	-1	0	1	2	3	4
1000	3	-4	-3	-2	-1	0	1	2	3
100	2	-5	-4	-3	-2	-1	0	1	2
10	1	-6	-5	-4	-3	-2	-1	0	1
1	0	-7	-6	-5	-4	-3	-2	-1	0
0.1	-1	-8	-7	-6	-5	-4	-3	-2	-1
0.01	-2	-9	-8	-7	-6	-5	-4	-3	-2
0.001	-3	-10	-9	-8	-7	-6	-5	-4	-3
0.0001	-4	-11	-10	-9	-8	-7	-6	-5	-4
0.00001	-5	-12	-11	-10	-9	-8	-7	-6	-5
0.000001	-6	-13	-12	-11	-10	-9	-8	-7	-6
0.0000001	-7	-14	-13	-12	-11	-10	-9	-8	-7

Xexp(+), Yexp=(-)

Xexp(-), Yexp=(+)

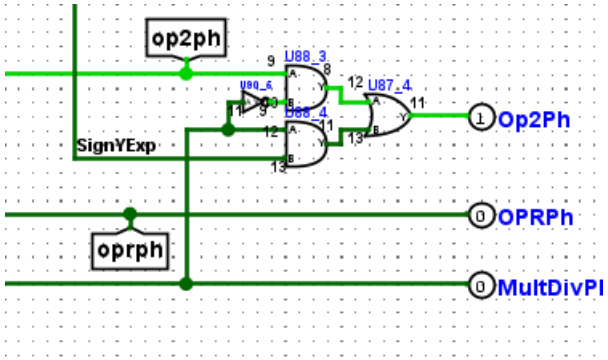
OP2(X)	0.1	0.01	0.001	0.0001	0.00001	0.000001	0.0000001
OP1 (Y)	-1	-2	-3	-4	-5	-6	-7
10000000	8	9	10	11	12	13	14
1000000	7	8	9	10	11	12	13
100000	6	7	8	9	10	11	12
10000	5	6	7	8	9	10	11
1000	4	5	6	7	8	9	10
100	3	4	5	6	7	8	9
10	2	3	4	5	6	7	8
1	1	2	3	4	5	6	7
0.1	0	1	2	3	4	5	6
0.01	-1	0	1	2	3	4	5
0.001	-2	-1	0	1	2	3	4
0.0001	-3	-2	-1	0	1	2	3
0.00001	-4	-3	-2	-1	0	1	2
0.000001	-5	-4	-3	-2	-1	0	1
0.0000001	-6	-5	-4	-3	-2	-1	0

Xexp(-), Yexp=(-)

Add Overflow

(DintOvrfl=1)(SignYexp=0)(SignXexp=0)
OR
(DintOvrfl=1)(SignYexp=1)(SignXexp=1)

Overflow happens when Op1 and Op2 are both 8-digit integers and the sum results in a final carry out digit 8.



Op2(X) is DPFirst AND DintOVrfl

Add Underflow

(DintOvrfl=1)(SignYexp=0)(SignXexp=1)
Use Yreg

Add Underflow

(DintOvrfl=1)(SignYexp=1)(SignXexp=0)
Use Xreg

Op2(X) is NOT DintOvrfl AND NOT DPFirst

Because SignYexp is only available to Mult and Div (due to arch oversight) I can't Use these expressions to detect underflow.

Dexp and Sexp Counter Values From All Combinations of Op1 and Op2

OP2

A div C...		0.0000001 -7		0.0000001 -7		0.000001 -6		0.000001 -6		0.00001 -5		0.00001 -5		0.0001 -4		0.0001 -4		0.001 -3		0.001 -3		0.01 -2		0.01 -2		0.1 -1		0.1 -1		0 0		1 0		1 0		10 1		10 1		100 2		100 2		1000 3		1000 3		10000 4		10000 4		100000 5		100000 5		1000000 6		1000000 6		10000000 7		10000000 7
		Dexp		Sexp		Dexp		Sexp		Dexp		Sexp		Dexp		Sexp		Dexp		Sexp		Dexp		Sexp		Dexp		Sexp				Dexp		Sexp		Dexp		Sexp		Dexp		Sexp		Dexp		Sexp		Dexp		Sexp		Dexp		Sexp		Dexp		Sexp	
0.0000001	-7	0		-14		-1		-13		-2		-12		-3		-11		-4		-10		-5		-9		-6		-8				-7		-7		-8		-6		-9		-5		-10		-4		-11		-3		-12		-2		-13		-1		-14		0
0.000001	-6	1		-13		0		-12		-1		-11		-2		-10		-3		-9		-4		-8		-5		-7				-6		-6		-7		-5		-8		-4		-9		-3		-10		-2		-11		-1		-12		0		-13		1
0.00001	-5	2		-12		1		-11		0		-10		-1		-9		-2		-8		-3		-7		-4		-6				-5		-5		-6		-4		-7		-3		-8		-2		-9		-1		-10		0		-11		1		-12		2
0.0001	-4	3		-11		2		-10		1		-9		0		-8		-1		-7		-2		-6		-3		-5				-4		-4		-5		-3		-6		-2		-7		-1		-8		0		-9		1		-10		2		-11		3
0.001	-3	4		-10		3		-9		2		-8		1		-7		0		-6		-1		-5		-2		-4				-3		-3		-4		-2		-5		-1		-6		0		-7		1		-8		2		-9		3		-10		4
0.01	-2	5		-9		4		-8		3		-7		2		-6		1		-5		0		-4		-1		-3				-2		-2		-3		-1		-4		0		-5		1		-6		2		-7		3		-8		4		-9		5
0.1	-1	6		-8		5		-7		4		-6		3		-5		2		-4		1		-3		0		-2				-1		-1		-2		0		-3		1		-4		2		-5		3		-6		4		-7		5		-8		6
OP1	0																																																											
	1	0	7		-7		6		-6		5		-5		4		-4		3		-3		2		-2		1		-1				0		0		-1		1		-2		2		-3		3		-4		4		-5		5		-6		6		-7		7
	10	1	8		-6		7		-5		6		-4		5		-3		4		-2		3		-1		2		0				1		1		0		2		-1		3		-2		4		-3		5		-4		6		-5		7		-6		8
	100	2	9		-5		8		-4		7		-3		6		-2		5		-1		4		0		3		1				2		2		1		3		0		4		-1		5		-2		6		-3		7		-4		8		-5		9
	1000	3	10		-4		9		-3		8		-2		7		-1		6		0		5		1		4		2				3		3		2		4		1		5		0		6		-1		7		-2		8		-3		9		-4		10
	10000	4	11		-3		10		-2		9		-1		8		0		7		1		6		2		5		3				4		4		3		5		2		6		1		7		0		8		-1		9		-2		10		-3		11
	100000	5	12		-2		11		-1		10		0		9		1		8		2		7		3		6		4				5		5		4		6		3		7		2		8		1		9		0		10		-1		11		-2		12
	1000000	6	13		-1		12		0		11		1		10		2		9		3		8		4		7		5				6		6		5		7		4		8		3		9		2		10		1		11		0		12		-1		13
	10000000	7	14		0		13		1		12		2		11		3		10		4		9		5		8		6				7		7		6		8		5		9		4		10		3		11		2		12		1		13		0		14

Division Overflows
Anytime Dexp is positive number greater than 7
Within the positive exp1, negative exp2 quadrant

(DPOvrfl=1)(SignYexp=0)(SignXexp=1)

Mult Overflows
Anytime Sexp is positive number greater than 7
Within the positive exp1, positive exp2 quadrant

(DPOvrfl=1)(SignYexp=SignXexp=0)

Div Underflows

(DPOvrfl=1)(SignYexp=1)(SignXexp=0)

Mult Underflows

(DPOvrfl=1)(SignYexp=SignXexp=1)