The following pseudo-code illustrates the behavior of the bit difference calculator. If there are X more 1s than 0s in the input, then the output is X. If there are Y more 0s than 1s, then the output is –Y.

One possible FSMD (used by the fsmd\_1p architecture):

Inputs: go, data (WIDTH bits)

Outputs: result (clog2(2\*WIDTH+1) bits), done

Reset values: done = 0, result = 0, diff = 0, data\_r = 0

while(1) {

while (*go* == 0);

done = 0;

*data\_r* = *data*; // Store input in a register.

// This ensures that the code will still

// work if the input changes during the loop.

*diff* = 0;

for *width* iterations {

if data\_r[0] == 1

*diff* ++;

else

*diff* --;

*data\_r* = data\_r >> 1; // Shift right by 1

}

*result* = *diff*;

*done* = 1;

}

*go=0*

*go=1*

*go=0*

*count\_r < WIDTH*

*count\_r == WIDTH*

**START**

**COMPUTE**

**RESTART**

*go=1 / diff\_r = 0; done\_r = 0;*

*go=1*

*go=0*

*count\_r < WIDTH*

*count == WIDTH / result = diff\_r, done\_r = 1*

**START**

**COMPUTE**

**RESTART**

*go=1 / diff\_r = 0; done\_r = 0;*

*go=0*

A third possible FSMD:

*count\_r == WIDTH / result = diff\_r; done\_r = 1;*

*go=1 / done\_r = 0;*

*go=0*

*count\_r < WIDTH*

**START**

**COMPUTE**

An alternative FSMD with 2 states (used by the fsmd\_1p\_2 architecture):



