```
idle: if (data_ready=0) goto idle; // wait until data_ready=1

load_c1:
   r4 = src1; // c1
load_c2:
   r5 = src1; // c2
load_c3:
   r7 = src1; // c3
load_c4:
   r8 = src1; // c4

store: if (data_ready=0) goto eidle;
reg[3] = data; // Store data in a register
err = 0; // reset error
zero: reg[0] = 0; // zero out accumulator

sort1: reg[1] = reg[2]; // Reorder registers
sort2: reg[2] = reg[9]; // Reorder registers
sort3: reg[9] = reg[10]; // Reorder registers
sort4: reg[10] = reg[3]; // Reorder registers

mul1: reg[6] = reg[1] * reg[4]; // 1st sample * 1st coef
add: reg[0] = reg[0] + reg[6]; // add Large pos. coefficient
if (V) goto eidle; // On overflow, err condition
mul2: reg[6] = reg[2] * reg[5]; // 2nd sample * 2nd coef
sub: reg[0] = reg[0] - reg[6]; // sub Large neg. coefficient

mul3: reg[6] = reg[9] * reg[7]; // 3rd sample * 3rd coef
add2: reg[0] = reg[0] + reg[6]; // add Large pos. coefficient
```

if (V) goto eidle; // On overflow, err condition

mul4: reg[6] = reg[10] * reg[8]; // 4th sample * 4th coef

sub2: reg[0] = reg[0] - reg[6]; // sub Large neg. coefficient

if (V) goto eidle; // On overflow, err condition

else goto idle;

eidle: err = 1;

if (data_ready=1) goto store; // wait until data_ready=1

if (data_ready=0) goto eidle;