Effective Exploitation of a *Zero Overhead Loop Buffer*

Gang-Ryung Uh
Yuhong Wang
David Whalley
Sanjay Jinturkar
Chris Burns
Vincent Cao

HARDWARE LOOPING SUPPORT (cont.)

```
for (i = 0; i < 10000; i++)
   a[i] = 0;
```

Source Code of Loop

```
    r0 = _a          # r[0]=ADDR(_a);
    a2 = 0           # a[2]=0;
    a1 = -9999       # a[1]=-9999;
L5: *r0++ = a2       # {M[r[0]]=a[2];
                     #  r[0]=r[0]+2;}
    a1 = a1 + 1      # {a[1]=a[1]+1;
                     #  IC=a[1]+1?0;}
    if le goto L5    # PC=IC<=0?L5:PC;
```

DSP16000 Assembly and Corresponding RTLs without Using the ZOLB

HARDWARE LOOPING SUPPORT (cont.)

```
cloop = 10000
r0 = _a
a2 = 0
do cloop {
  *r0++ = a2
}
```

After Using the ZOLB

LOOP SPLITTING

```
for (i = 0; i < 10000; i++) {
   a[i] += a[i]*x;
   b[i] += b[i]*y;
   c[i] += c[i]*x;
   d[i] += d[i]*y;
   x = x+1;
   y = y+2;
}
```

Source Code before Loop Splitting

```
for (i = 0; i < 10000; i++) {
   a[i] += a[i]*x;
   c[i] += c[i]*x;
   x = x+1;
}
for (i = 0; i < 10000; i++) {
   b[i] += b[i]*y;
   d[i] += d[i]*y;
   y = y+2;
}
```

Source Code after Loop Splitting

LOOP COLLAPSING

```
int a[50][100];

for (i = 0; i < 50; i++)
   for (j = 0; j < 100; j++)
      a[i][j] = 0;
```

Original Nested Loops

```
int a[5000];

for (i = 0; i < 5000; i++)
   a[i] = 0;
```

After Loop Collapsing

LOOP INTERCHANGE

```
extern int a[200][100];

for (i=0; i<200; i++)
   for (j=0; j<50; j++)
      a[i][j]=0;
```

Source Code of Nested Loops

```
extern int a[200][100];

for (j=0; j<50; j++)
   for (i=0; i<200; i++)
      a[i][j]=0;
```

Source Code after Loop Interchange

RESULTS and CONCLUSIONS

(1) Description of Test Programs

(2) Contrasting Loop Unrolling and Using the ZOLB

(3) Impact of Improving Transformations on Using the ZOLB

TEST PROGRAMS

Program	Description
add8	add two 8-bit integers
copy8	copy one 8-bit image to another
fir	finite impulse response filter
fire	fire encoder
inverse8	invert an 8-bit image
lms	lms adaptive filter
sumabsd	sum of absolute diff of images
vec_mpy	simple vector multiply
conv	convolution code
fft	128 point complex fft
fir_no	fir filter with no redundant load
iir	iir filter
jpegdct	jpeg discrete cosine transformation
scale8	scale an 8-bit image
trellis	trellis convolutional encoder

PERFORMANCE RESULTS

Program	Unrolling(4)	ZOLB
add8	−23.11%	−36.33%
conv	−47.56%	−47.84%
copy8	−42.32%	−62.44%
fft	−10.56%	−8.69%
fir	−35.25%	−48.42%
fir_no	−7.07%	−31.35%
fire	−4.22%	−26.88%
iir	−15.43%	−19.61%
inverse8	−37.34%	−55.50%
jpegdct	−8.44%	0.00%
lms	−10.52%	−8.33%
scale8	−9.37%	−14.28%
sumabsd	−19.57%	−58.83%
trellis	−19.10%	−20.16%
vec_mpy	−28.49%	−38.16%
Average	**−24.22%**	**−31.79%**

CODE SIZE RESULTS

Program	Unrolling(4)	ZOLB
add8	+62.75%	−3.92%
conv	+29.03%	−3.23%
copy8	+12.50%	−4.17%
fft	+92.86%	−3.57%
fir	+147.37%	−10.53%
fir_no	+109.30%	−4.65%
fire	+110.78%	−21.57%
iir	+51.04%	−4.17%
inverse8	+18.37%	−4.08%
jpegdct	+59.54%	0.00%
lms	+1.78%	−0.04%
scale8	+93.85%	−1.54%
sumabsd	+25.71%	−8.57%
trellis	+0.33%	−0.17%
vec_mpy	+336.84%	−15.79%
Average	**+76.80%**	**−5.73%**