Automatic Source-to-Source Code Generation for Vector Hardware Accelerators

Serge Guelton, serge.guelton@telecom-bretagne.eu
GDR SOC-SIP – 9-10-11 June 2010

From C or FORTRAN legacy code...
- Widely spread in scientific community
- High level languages
- Easy to read and to maintain

...to hardware-accelerated assembly
- Parallel vector processing
- Real-time throughput
- Lower energy consumption

One Challenging Target: the Ter@pix Processor
- Limited Memory Size
- Fixed-size transfers
- 2D memory layout
- Integers only
- No division
- Hard-coded loops
- Limited control flow
- Vector loops
- No masked execution

Source-to-source code transformations

1 Derived Host Call

```
# PIPS generated variable
unsigned int i, j;
for (i = 1; i <= 128; i += 1)
launchers_0_microcode(i_0, CST0, &result0[-1+i],
&src00[-1+i], &src10[-1+i]);
```

2 Derived Vector Loop Emulator

```
void launchers_0(int i_0, int CST0[4], short result0[128][8],
short src00[128][8], short src10[128][8])
{
  // PIPS generated variable
  unsigned int i, j;
  for (i = 0; i < 64; i += 1)
  for (j = 0; j < 4; j += 1)
  launcher_0(i, CST0, result0, src00, src10);
}
```

3 Derived Accelerator Code

```
void launcher_0_microcode(int i_0, int CST0[4], short result0,
short src00, short src10)
{
  // PIPS generated variable
  unsigned int j;
  for(j = 0; j < 4; j += 1)
  (*result0 +j-1) = (CST0[2]*(*src00+j-1)+
CST0[3]*(*src10+j-1))>>(CST0[0]<CST0[1]);
}
```

4 Derived Assembly

```
sub launchers_0_micocode_0
    im, im7=FIP02 || || ||
    im, im5=FIP01 || ||
    ... || ||
    P, re(21)=re(21)+re(30) || ||
    endsub
```

Implemented in the PIPS Infrastructure

python compiler assembler
inter-procedural Analysis
array region analysis

Open Source Software from CRI/MINES ParisTech
memory effect computations
array privatization

Funded by the ANR