Code-generator of parallel assembly code for digital signal processor

Bukharenko Nikita
Problems

• The necessity of developing programs on low-level language (assembler)
• Getting an inefficient code after translating program from C/C++ to assembly language
• The necessity of manual parallelization of assembly code
Approach to the solution

The scheme - is the natural representation of the digital signal processing problem
• There is no need to represent it in text form
• The scheme naturally shows the parallelism
The purpose

Code-generator of parallel assembly code according to the scheme of objects for microprocessor’s DSP-core
Tasks

• Targeting to different instruction sets
• Generating parallel and sequential code (single-threaded cores only)
• Optimization of registers and memory work
The scheme of the code-generator’s environment
The scheme of the code-generator’s environment

- **Scheme**
- **Intermediate representation**
- **Scheduler**

Code generator supports the scheme with such operators:

- Computing operators
- Data objects
  - (variables, data)
- Cycles
- Conditions
The scheme of the code-generator’s environment

Scheduler

The schedule of objects implementation

(1) Spl  (2) *  (3) +  (4)  (5) -  (6) Out

Assembly code for DSP-core
The scheme of the code-generator’s environment

An example of **multiplication’s** template for DSP-core:

```vhdl
entity llmul [dsp2] is
mpss &regin1,&regin2,&regout1
entity llmul end.
```

An example of **subtraction’s** template for DSP-core:

```vhdl
entity llsub [dsp1] is
sub &reginx1,&reginy1,&regout1
entity llsub end.
```
The scheme of the code-generator’s environment

- Scheme
  - Intermediate representation
  - Scheduler
  - Templates of processor’s instructions
  - Code-generator
    - The schedule of objects implementation
    - Assembly code for DSP-core
Optimization

The main directions in the optimization of the output code:

• Optimization of memory usage (to minimize memory access)

• Optimization of general purpose registers usage (economical usage of a limited set of registers)
An example of code-generator’s work for DSP-core

Scheme

Generated code

move 0x1,R2
move 0x4,R4
move 0x00000008,R6
move 0x2,R8
move 0x5,R10
add R2,R8,R12
move 0x1,R2
do R12,label_cycle_1003
mpss R10,R2,R8 add R2,R4,R14
move R8,R10
move R14,R4
label_cycle_1003:
inc R2,R2
sub R4,R10,R8
move R6,A0
move R8,(A0)

Result: 0x14
An example of code-generator’s work for RISC-core

Scheme

Generated code

li $2,0x1
li $3,0x4
li $4,0x00000008
li $5,0x2
li $6,0x5
add $7,$2,$5
li $2,0x1
add $7,$7,0x1
label_cycle_1003:
beq $7,$2,label_finish_1003
nop
mul $5,$6,$2
add $8,$2,$3
add $6,$5,0x0
add $3,$8,0x0
add $2,$2,0x1
j label_cycle_1003
nop
label_finish_1003:
sub $5,$6,$3
sw $5,0($4)

Result: 0x14
Resume

• Visual scheme instead of text code
• Explicit parallelism at the level of the scheme
• An efficient parallel assembly code
• Targeting to different processors and instruction sets
Thank you for your attention!