POLITECNICO DI MILANO

Tuning and Customization of Generated Accelerators
Compiler Based Optimizations

Marco Lattuada

Politecnico di Milano
Dipartimento di Elettronica, Informazione e Bioingegneria
marco.lattuada@polimi.it

Activity 2.1 – Write C Interface

- Synthesize a module which takes as input an array of integer with arbitrary size and returns the minimum and maximum value
  - Do not use structure

Activity 2.1 – Solution Hint

Example: Code returning the maximum of array of 10 elements

```
void max(int input[10], int * max)
{
   int local_max = input[0];
   int i = 0;
   for(i = 0; i < 10; i++)
   {
      if(input[i] > local_max)
      {
         local_max = input[i];
      }
   }
   *max = local_max;
}
```

Activity 2.1 – Possible Solution

```
void min_max(int * input, int num_elements, int * max, int * min)
{
   int local_max = input[0];
   int local_min = input[0];
   int i = 0;
   for(i = 0; i < num_elements; i++)
   {
      if(input[i] > local_max)
      {
         local_max = input[i];
      }
      else if(input[i] < local_min)
      {
         local_min = input[i];
      }
   }
   *min = local_min;
   *max = local_max;
}
```

Activity 2.2 – Write testbench

- Write testbench for the module designed in the previous activity
    - Test arrays with different elements and different sizes

Activity 2.2 – Solution Hint

```
--generate-tb=<xml_file>
```

```
<?xml version="1.0"?>
<function>
   <testbench input="0,1,2,3,4" num_elements="5"/>
</function>
```

Activity 2.2 – Possible Solution

```
<?xml version="1.0"?>
<function>
   <testbench input="0,1,2,3,4" num_elements="5"/>
   <testbench input="0,1,2,3,4,5,6,7,8,9" num_elements="10"/>
   <testbench input="0,0,0,0,0,0,0,0,0,0" num_elements="10"/>
   <testbench input="0" num_elements="1"/>
</function>
```

Activity 2.3 – Select target device and clock target period

- Compare the number of cycles required by a function executing 64 bit multiplication on the following target
  - xc4vlx100-10ff1513 – 66MHz
  - 5SGXEA7N2F45C1 – 200MHz
  - xc7vx690t-3ffg1930-VVD – 100MHz
  - xc7vx690t-3ffg1930-VVD – 333MHz
  - xc7vx690t-3ffg1930-VVD – 400MHz

Activity 2.3 – Solution Hint

```
bambu module.c --device-name=xc4vlx100-10ff1513
--clock-period=15 --simulate
```

Activity 2.3 - Solution

```
bambu module.c --device-name=xc4vlx100-10ff1513
--clock-period=15 --simulate
```

```
bambu module.c --device-name=5SGXEA7N2F45C1 --clock-period=5
--simulate
```

```
bambu module.c --device-name=xc7vx690t-3ffg1930-VVD
--clock-period=10 --simulate
```

```
bambu module.c --device-name=xc7vx690t-3ffg1930-VVD
--clock-period=3.3 --simulate
```

```
bambu module.c --device-name=xc7vx690t-3ffg1930-VVD
--clock-period=2.5 --simulate
```

Activity 2.4 – Evaluate GCC optimizations

- Evaluate the effects of GCC optimizations on the number of cycles of adpcm benchmark
  - Different level of optimizations
  - Vectorization
  - Different inlining

Activity 2.4 – Solution Hint

```
bambu adpcm.c -O0 --simulate
```

Activity 2.4 Solution

```
bambu adpcm.c -O0 --simulate
```

```
bambu adpcm.c -O1 --simulate
```

```
bambu adpcm.c -O2 --simulate
```

```
bambu adpcm.c -O3 --simulate
```

```
bambu adpcm.c -O3 --simulate
-finline-limit=1000000
```

```
bambu adpcm.c -O3 --simulate -ftree-vectorize
```

Activity 2.5 – Use SDC Scheduling

- Check if SDC scheduling based optimizations further improve the best results obtained in the previous activity

Activity 2.5 – Solution Hint

```
--speculative-sdc-scheduling
```

Activity 2.5 – Solution

```
bambu adpcm.c -O3 --simulate
--speculative-sdc-scheduling
-finline-limit=1000000
```

Activity 2.6 – Integer Division

- Evaluate the effects on the number of cycles in using different integer division implementations on the dfdiv algorithm targeting Zynq and 66MHz

Activity 2.6 – Solution Hint

```
--hls-div=<type>
```

Activity 2.6 – Solution

Activity 2.7 – Softfloat and libm

- Generate the module implementing the following formula (single precision and double precision):

$$\gamma = \operatorname{acos}\frac{a^2 + b^2 - c^2}{2ab}$$

- Identify the combination of softfloat ops and libm which produces the best performances

Activity 2.7 - Hint

Use the following parameters:

```
--libm-std-rounding
--soft-float
…
```

Activity 2.7 – Solution

- ❑ `--softfloat` is the best choice
- ❑ Use of standard or faithful rounding only impacts on area
- ❑ Replace `pow(x, 2.0)` with `x * x` does not improve performances
  - ▶ Replacement is already performed by GCC