Workshop Agenda

- A|RT Designer Review
- FIR Filter Case Study
- Hands-on Exercise
- Questions and Answers
- Homework Assignment
Design Flow

Abstraction Levels

- Algorithm
- Behavioral Synthesis
- Architecture
- RT-level Synthesis
- Gates
- Layout Generation
- Layout

Design Flow Diagram

- Algorithm
- Behavioral Synthesis
- Architecture
- RT-level Synthesis
- Gates
- Layout Generation
- Layout
Design Flow

System Specification
Embedded Software

Datapath Resources
(arithmetic, memory)

ANSI
Library

HW
Resource
Library

HW
Resource
Library

Edit/Compile
Create Architecture

Map to Architecture

Schedule Operations

Source Code
Tuning

Architecture
Optimization

Performance Analysis

Build RTL code

Logic Synthesis

FPGA

ASIC

Target VLIW Processor Architecture

branch
logic

ALU

MULT

OUT

IN

RAM

ROM
Structure of a Cluster

Internal Design Flow

<table>
<thead>
<tr>
<th>1.</th>
<th>2.</th>
<th>3.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compilation</td>
<td>Architecture Creation</td>
<td>Mapping</td>
</tr>
</tbody>
</table>

- Pragmas
- Libraries

1. **Compilation**
2. **Architecture Creation**
3. **Mapping**
4. **Scheduling**
5. **Building**

- C
- VHDL
- HDL

```
1. 2. 3. 4. 5.
```
**Project Organization**

```
void calc_addr(const T_AD i, const T_AD j, T_AD& address)
{
    address = const1*i + const2*j;
}

void mydesign(...)
{
    ...
    for (i=0; i<16; i++)
    {
        for (j=0; j<16; j++)
        {
            calc_addr(i,j,address);
            a = A[address];
            .... //calculation of b
            A[address] = b;
        }
    }
}
```

**Data Flow Analysis**

DFA reconstructs parallelism from sequential C description.

DFA will check whether or not address is different for every iteration! This will determine how much loop folding can be performed.
Fixed-point Classes (2)

C++ Template Class Hierarchy

Abstract Base Class:
Number<S,W,P>

Fix<8,5>

Int<8>

Ufix<8,5>

Uint<8>

FIR Filter Case Study

http://www.frontierd.com
Case Study

FIR: The Characteristics

Band-pass filter with 71 symmetric filter coefficients

Frequency response

impulse response

Case Study

FIR: Algorithm and Architecture

void FIR(const T_IN IN, T_OUT OUT)
{  #pragma OUT OUT
   /* Definition of the summation of the tap multipliers: */
   static T_AB baseptr = 1;
   T_AD index = T_addr_getsum(baseptr);
   T_AR ACC = 1N * BASE(j);
   loop: for IT_AD j=1: NUM_TAP: ++j)
   {  RCO = T_ADDRBASE(index) * BASE[i];
      index = T_addr_next tweaking(index, mod);
   }
   /* Definition of the output: */
   baseptr = T_addr_scacum(baseptr.mod);  
   DELAY(baseptr) = RRO;
   OUT = ACC;
}
Case Study

Default Run

215 cycles for a 71 taps filter is too much!

Solution: Improved Address Computation

2 additional ACU’s
Case Study

Improved Addressing Computation

76 cycles instead of 215

Algorithm Improvement

- 2 different delay-lines are used (stored on different RAM’s)
- due to symmetric coefficients, it is possible to first add two delayed values with the same coefficient before performing a multiplication

# multiplications is reduced with factor 2
Case Study

**Architecture Changes**

Results

Only 44 cycles are needed for 71 taps filter
## Summary of Results

<table>
<thead>
<tr>
<th>Description</th>
<th>Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>215</td>
</tr>
<tr>
<td>improved address computation</td>
<td>76</td>
</tr>
<tr>
<td>extra resources</td>
<td>53</td>
</tr>
<tr>
<td>using symmetry</td>
<td>44</td>
</tr>
</tbody>
</table>