Integration of reduction operators in a compiler for FPGAs

Manjukumar Harthikote Matha
Department of Electrical and Computer Engineering
Colorado State University
Outline

- Introduction to RC system and FPGA
- The Cameron project
- SA-C compiler
- Histogramming – new reduction operator
  - AHAHA graph
  - VHDL designs
  - Experimental results
- Median filter – new reduction operators
  - AHAHA graph – using Stones perfect shuffle network
  - Optimizations at AHAHA level
  - Experimental results
- Summary
Introduction

Reconfigurable Computing (RC)

- RC systems consist of a host coprocessor and a FPGA board, memories and a PCI communication device.
- FPGAs can be reconfigured (reprogrammed) to perform a specific function.

[Diagram showing CPU, MEMORY, FPGA, and MEMORY components connected via PCI.]
FPGA

- FPGAs consist of an array of interconnected Configurable Logic Blocks (CLBs)
- CLBs consist of two 4 input LUTs, carry logic and 2 storage elements (FFs)
- Global Routing Matrices (GRMs) are used for interconnecting CLBs
- BRAM – on chip memory for FPGAs
- DLL – Eliminate clock skew and provide advanced clocking mechanism
- ADM XRC–II board is used in this study
The Cameron Project

Objective

- Provide a software approach for the development of RC systems
- Using SA-C (Single Assignment C) – a “extended subset” of C language
- SA-C* (Single Assignment C with streams) language – an extension of SA-C
- Support for image processing

Approach

- One step compilation from SA-C code to FPGA configuration code
- Automatic generation of host-board interface
- Optimizations done by compiler at various levels for better hardware performance (space, time, memory traffic)
- User controlled optimizations through pragmas
SA-C compiler

- SA-C* code is translated in following steps before generating the hardware configuration bit-stream
  - DDCF – Data Dependency and Control Flow Graph
  - DFG – Data Flow Graph
  - AHAHA – Aggregated Hierarchical Abstract Hardware Architecture

- Introducing a new operator involves integrating it in all phases of the compiler
- Aggregated Hierarchical Abstract Hardware Architecture expands complex nodes in DFG to simpler nodes

- Handshaking signals are used for synchronization of nodes
  - _In signals are used for synchronization with the producer nodes
  - _Out signals are used for synchronization with the consumer nodes

Handshaking signals in AHAHA
New operator: Histogramming

- Histogram refers to the distribution of image pixel values over the available gray scale.

- In SA-C* histogram is a “hybrid” reduction operator involving both sum reduction and array creation. None such hybrids existed before.

  ```
  histogram ( input pixel, size of histogram, mask value)
  ```

- The translation to AHAHA level and VHDL implementations of histogram is done as part of this study.

  ```
  uint8 [:] main ( uint8 I [:,:], bool M [:,:]) {
      hardware() {
          uint8 R [:] =
              for i in I dot m in M return ( histogram (i, 10, m ) );
      }
      return (R);
  }
  ```
Write tile node is introduced before converting to AHAHA level (the only node in SA-C* to go such transformation before AHAHA level)
AHAHA graph
AHAHA graph
AHAHA graph
AHAHA graph
Histogramming on FPGA

- Histogram operation is done using on-chip memory BRAM

- New pixel value at every clock cycle

- The accumulation takes place till the done signal arrives

- BRAM has “one clock latency” for either read or write operation
Memory conflict

AHA_HIST

BRAM

PORT A

Data read from 0x16

Pixels

0x16

Memory Conflict

PORT B

Write data to 0x16

Accumulator
Avoiding memory conflicts

- Three different designs
  - Handshaking – Usage of handshaking signals to avoid conflicts
  - Counter Based – Usage of counter to avoid conflicts
  - Clock-2x – Double the clock using DCM

State machine of handshaking design
Handshaking Approach

- Get_Data state is the initial state waiting for valid data
- Update_Hist state starts histogramming
- Delay state is invoked in the case of memory conflict
Counter design

- A explicit counter maintained
- BRAM updated with the counter value only when current value is not same as the previous
Counter Design
Clock-2x design

- Double the clock rate using DCM (Digital Clock Management)
- New clock is provided to BRAM – read and write operation in a single cycle
Clock-2x design

CLK

CLK2X

INPUT VALUE

A1  A2  A3  A3  A4

ADDR

A1  A2  A3  A3  A4

DOA

D1  D2  D3  D3  D4

DIA

D1+1  D2+1  D3+1  D3+1  D4+1

CURRENT_STATE

Read  Write  Read  Write  Read  Write  Read  Write
## Experimental Results

### Input image
![Input image](image.png)

### Histogram of the image
![Histogram](histogram.png)

<table>
<thead>
<tr>
<th>Design</th>
<th>Number of slices</th>
<th>Maximum frequency (MHz)</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handshaking</td>
<td>879</td>
<td>84.991</td>
<td>4.522 ms</td>
</tr>
<tr>
<td>Counter</td>
<td>912</td>
<td>87.635</td>
<td>4.146 ms</td>
</tr>
<tr>
<td>Clock-2x</td>
<td>806</td>
<td>87.589</td>
<td>4.144 ms</td>
</tr>
</tbody>
</table>
Median Filter

- Median filter is used to remove noises in an image.
- Operates on a window of pixels.
- SA-C* syntax “array_median (window of pixels)”
- DFG creates a REDUCE_Umedian_MACRO node.

```c
uint8 [::] main ( uint8 I [::] ) {
    hardware() {
        uint8 R [::] =
            for window W[3,3] in I return ( array_median (W) );
    }
} return (R);
```
AHAHA

- Stones shuffle network \([1]\) is used for sorting the pixels
- For \(n\) elements, requires \(\log n\) stages and are sorted in \(\Theta (\log^2 n)\)
- Network requires \(2^{k-1}[k(k-1) + 1]\) comparators for a list of \(n = 2^k\)

\[\text{[1] Stone H.S, Parallel processing with the perfect shuffle. In IEEE transaction on Computing, C-20, pp153-161, Feb. 1971}\]
Optimizations at AHAHA level

- Constant folding optimization
- Remove the blank nodes
- Remove the nodes with input zeroes
- Removal of unnecessary nodes in the last stage

```
x
0
MAXCOMPARE
(-)
0
x
```

```
x
0
MINCOMPARE
(+)
x
0
```
AHAHA Optimizations
AHAHA Optimizations

![Diagram with symbols and variables: c, d, e, f, +, -]
Experimental Results

- Partial unrolling to prevent re-read of pixels

![Diagram showing the difference between without and with loop unrolling](image-url)
Experimental Results

Input noisy image  3x3 Window median filter  5x5 Window median filter
Experimental Results

Real Time Execution of Median Filter

- 5x5 Window median filter
- 3x3 Window median filter
Summary

- Histogramming using Counter and Clock-2x provided better performance.
- Median filter is optimized at AHAHA level in SA-C*.

<table>
<thead>
<tr>
<th>Window Size</th>
<th>$n=2^k$</th>
<th>Total nodes needed</th>
<th>Nodes needed after optimization in SA_C*</th>
</tr>
</thead>
<tbody>
<tr>
<td>3x3</td>
<td>16</td>
<td>104</td>
<td>32</td>
</tr>
<tr>
<td>5x5</td>
<td>32</td>
<td>336</td>
<td>141</td>
</tr>
<tr>
<td>7x7</td>
<td>64</td>
<td>992</td>
<td>401</td>
</tr>
</tbody>
</table>
Questions ?