Automatic Exploration of Hardware/Software Partitioning

Syed Abbas Ali Shah, Sven Alexander Horsinka, Basitan Farkas, Rolf Meyer, Malden Berekovic

E.I.S., TU Braunschweig
Designer Experience

Design Space

Right Design

Source: binoculas.net
DSE Strategies

Exhaustive Search:
- Simulate all the possible designs then choose the best one among them (Minimum Energy Delay Product)

Design Parameters → Optimal solution

Less number of simulation => Less time

$ES_{total\_simulations} = \prod_{i=1}^{N} n_i$

Time consuming
Can take days!

Give the Designer more freedom; being involved as less as possible in the MPSoC design Space Exploration

Source: ayehu.com
SoCRocket - The building blocks
SoCRocket is more than a model library
Included SystemC Models (core)
All models where developed with RTL equivalents as blueprint
SoCRocket - DSE Flow
Runtime re-configuration

Design Idea / Reference Software

Software

Params/Defaults (JSON)

Exploration Prototype (HW) (SystemC design)

Compile SW

SW ELF (config independent)

Compile Simulator (HW)

Platform Simulator EXE (config independent)

GreenControl/CCI

SW Runtime Config. (mkprom)

Configuration (JSON)

HW Runtime Configuration (Read JSON, CLI opts)

Platform Simulation/Analysis

(From SoCRocket Design Flow Report)
Baseline SoC architecture based on SoCRocket IPs
Accelerator architecture and data flow

CPU 0
MMU
D/I Cache

Accelerator
TLM wrapper computation
out buf in buf
Slave IF

Bus
1. read in
2. write in
3. read out
4. write out
Memory

Software

CPU 0
MMU
D/I Cache

Accelerator
TLM wrapper computation
out buf DMA in buf
Master/Slave IF

Bus
1. configuration
2. data access
4. interrupt
3. IRQ
IRQMP
Memory

DMA
SW2TLM automated tool chain and user interaction

Characterized Software Section

Software (baseline)

User Input (JSON)

SW2TLM

SW to be accelerated

SW Wrapper (drivers)

HLS/Synthesis (Vivado/DC)

TLM Wrapper

Exploration Results

Software (modified)

Virtual Platform (SoCRocket)

Hardware (modified)
void dct(signed char pixels[8][8], int color, signed short dctresult[8][8])

**conf**: {
  "file_name": "dct.c",
  "function_name": "dct",
  "defines": "header.h",
  "irq_en": true,
  "dma_en": false
}

**input**: {
  "pram1": {
    "vname": "pixels",
    "typ": "signed char",
    "len": "8*8"
  },
  "pram2": {
    "vname": "color",
    "typ": "int",
    "len": "1"
  }
}

**output**: {
  "pram1": {
    "vname": "dctresult",
    "typ": "signed short",
    "len": "8x8"
  }
}
### Performance analysis parameters

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{receive}$</td>
<td>Writing data to HW ACC directly or via DMA</td>
</tr>
<tr>
<td>$t_{transmit}$</td>
<td>Reading data from HW ACC directly or via DMA</td>
</tr>
<tr>
<td>$t_{decode}$</td>
<td>Decoded data in ACC</td>
</tr>
<tr>
<td>$t_{encode}$</td>
<td>Encode data in ACC</td>
</tr>
<tr>
<td>$t_{computation}$</td>
<td>HW ACC computation time</td>
</tr>
<tr>
<td>$t_{com-overhead}$</td>
<td>Communication overhead</td>
</tr>
<tr>
<td>$t_{acc_total}$</td>
<td>Total HW ACC execution time</td>
</tr>
<tr>
<td>$t_{sw}$</td>
<td>Execution time in of selected software section</td>
</tr>
<tr>
<td>$t_{app}$</td>
<td>Absolute application execution time</td>
</tr>
</tbody>
</table>

$$t_{acc_total} = t_{receive} + t_{decode} + t_{computation} + t_{encode} + t_{transmit}$$
### Performance analysis parameters

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_{\text{leakage}}$</td>
<td>Static power consumption</td>
</tr>
<tr>
<td>$P_{\text{internal}}$</td>
<td>Internal part of dynamic power consumption</td>
</tr>
<tr>
<td>$P_{\text{switching}}$</td>
<td>Switching part of dynamic power consumption</td>
</tr>
<tr>
<td>$P_{\text{total}}$</td>
<td>Total power consumption</td>
</tr>
<tr>
<td>$e_{\text{total}}$</td>
<td>Total energy consumption</td>
</tr>
</tbody>
</table>

\[
P_{\text{total}} = P_{\text{leakage}} + P_{\text{internal}} + P_{\text{switching}}
\]

\[
e_{\text{total}} = P_{\text{total}} \times t_{\text{app}}
\]
<table>
<thead>
<tr>
<th><strong>Test Applications</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Discrete Cosine Transformation (DCT)</strong></td>
</tr>
<tr>
<td><strong>Image Filtering (IF)</strong></td>
</tr>
<tr>
<td><strong>Image Integration (II)</strong></td>
</tr>
</tbody>
</table>
Combined accelerator delays

The overhead introduced by decoding, encoding and transmission can account for up to 70% of the overall accelerator delay (image integration).
Absolut application speed up

Highest speedup is achieved for computational complex image filtering (up to 6.5x)
Communication overhead

\[ C_{ot} = \frac{t_{receive} + t_{decode} + t_{encode} + t_{transmit}}{t_{acc\_total}} \]

- DCT_ACC
- DCT_ACC_DMA
- II_ACC
- II_ACC_DMA
- IF_ACC
- IF_ACC_DMA

Communication Overhead(%)
Power & Energy consumption relative to SW

- As the TL model of the ISS does not incorporate idle power states, the average power consumption is increased by the accelerator.
- Despite this limitation, the overall energy consumption is still significantly improved (between 18% and 62%).
Summary

• Automatic and fast framework for complex hardware/software systems
• Real world image processing demonstrator
• Improved engineering support for difficult DSE
• TL accelerator generation based on minimal user input
• Realistic performance, power and communication overhead analysis is performed using a state of the art virtual platform
  – Analyzing communication is of high importance (up to 70% of the computational delay)
  – Early analysis of energy consumption is critical for battery dependent systems
SoCRocket is available online: https://socrocket.github.io/

For more information do not hesitate to contact us!
Questions ?