SystemC Refinement of Abstract Adaptive Processes for Implementation into Dynamically Reconfigurable Hardware

F. Herrera
E. Villar
P.A. Hartmann
Outline

• Motivation
• Contribution
• Results
• Conclusions
Motivation (Requirements for Specification and Implementation)

- **Challenges**
  - Complexity, Changing Environments, Reusability, Performance

- **Modelling**
  - Abstraction (High-Level Modelling)
  - **Adaptivity:** A main aspect

- **Design**
  - ESL, Co-Design (HW/SW solutions)
  - **DRHW:** Hw Performance for Mutually Exclusive Algorithms fulfilling a same functionality
Motivation (SoA)

• **Adaptivity in SW Programming**
  – CLOS, Dylan, …

• **SystemC High-Level Modelling (ESL)**
  – SystemC-H, SysteMoC, HetSC
  – DRHW

• **ANDRES**
  – Adaptive HetSC (A-HetSC)
  – OSSS+R
Motivation (Previous Work: OSSS+R)

- **Modelling**
  - OO abstraction
  - functionality accessed from a common Interface (Adaptivity)

- **Link to Implementation**
  - Fossy
  - DRHW (Mutually Exclusive Semantics)
    - Clocked Model
    - Abstraction and Simulation Speed Improvable!
Motivation (Previous Work: A-HetSC)

- Untimed Specification of Adaptivity (HetSC Adaptive Processes, HAPs)
  - Fast Modelling & Simulation
- Formal Basis (ForSyDe)
  - High-Level Well Defined Semantics

- Link to HW implementation

1. Read Adaptation Input
2. Update Adaptation Context
3. Read Regular Inputs
4. Compute Adapted Functionality
5. Write Outputs
Motivation (From A-HetSC to OSSS+R)

A-HetSC

DSE

OSSS+R

fossy

SWGen

Embedded SW

Synthesizable HDL for DRHW
Contribution: Refinement Flow from A-HetSC to OSSS+R

A-HetSC

Proposed Refinement

(1) Adaptation Type

(2) Time Domain

(3) Reconf. Object Wrapping

(4) Fitting to Synthesis Subset

OSSS+R
Starting Model: Function Adaptation
Starting Model: Function Adaptation

```cpp
while (true) {
    a_in.read(*itf_p);
    // input data loop
    d_out.write(f[i]);
    // compute
    itf_p(f,f);
    // output data loop
    d_out.write(f[i]);
}
```
1st Ref.: Adaptation Type
while(true) {
  a_in.read(mode);
  // compute itf[mode](F,f);
  d_out.write(f[i]);
  // output data loop
  d_out.write(f[i]);
}

...
2nd Ref.: Time Domain
2nd Ref.: Time Domain

while (true) {
  a_in.read(mode);
  wait();...
}

// input data loop
... d_out.write(f[i]);

// compute
itf[mode](F,f);
wait();...

// output data loop
... d_out.write(f[i]);
  wait();...
3rd Ref.: Reconf. Object Wrapping
while(true) {
    ...
    switch(mode) {
    case IDCT_MODE;
        ro = IDCT(); break;
    case IHAAR_MODE;
        ro = IHAAR(); break;
    }
    ...
    // compute
    ro->do(F,f);
    wait();...
    ...
    // output data loop
    ... d_out.write(f[i]);
    wait();...
    }
4th Refinement: Fitting to Synthesis Subset

- OSSS+R specific
  - Pass by reference (instead by pointer)
  - Chain of “=“ splitted into single “=“ statements
  - Single compilation unit

- SystemC Synthesys Subset
  - SC_CTHREAD
  - Bounded loops

- Code transformations for efficiency
Use Case: fuIT Refinement in AVD

- Functional Validation

  - Dump to a raw video file (.yuv)
  - Displaying it
  - Check RT constraint

  30 fps, 200 frames 288x240

  - DCT and HAAR coding switching (T=10 frames, 0.33s)
  - 10MB/s, 8 bits (flash T=100ns)
Results: fuIT Refinement in AVD

AVD (RVC) Untimed

fuIT function

frame_decoder

fuMBR fuDCR fuIS
Results: fuIT Refinement in AVD

- Evaluation of functionality
- Blurring eliminated (adaptivity)
- fuIT (~ 50 code lines)
- 11s simulation time, close real time (6.67s)
- No violation of RT constraint
Results: Computational load

- Refinement of the fuIT worthy
Use Case: R1 Refinement

- Dump to a raw video file (.yuv)
- Displaying it
- Check RT constraint

30 fps, 200 frames 288x240

- DCT and HAAR coding switching (T=10 frames, 0.33s)
- 10MB/s, 8 bits (flash T=100ns)
Use Case: R1 Refinement

- Dump to a raw video file (.yuv)
- Displaying it
- Check RT constraint

✓ Simulation time not affected

✓ Functionally the same (bit accurate)

No violation of RT constraint
Use Case: R2 Refinement

- Use of HetSC BCs

- Dump to a raw video file (.yuv)
- Displaying it
- Check RT constraint

30 fps, 200 frames 288x240

- DCT and HAAR coding switching (T=10 frames, 0.33s)

- 10MB/s, 8 bits (flash T=100ns)
Use Case: impact of Time Domain Refinement (R2)

- Study of minimum frequency for fuIT computational load vs frequency
  - Assuming a Real Time execution

<table>
<thead>
<tr>
<th>F(MHz)</th>
<th>100</th>
<th>10</th>
<th>7.5</th>
<th>6.67</th>
<th>6.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>fuIT t(s)</td>
<td>0.44</td>
<td>1.99</td>
<td>5.78</td>
<td>6.65</td>
<td>6.69</td>
</tr>
<tr>
<td>fuIT load (%) Vs RT exec.</td>
<td>6.6</td>
<td>29.8</td>
<td>86.6</td>
<td>97.4</td>
<td>&gt;100</td>
</tr>
</tbody>
</table>

- Slow down of simulation speed
  - 81s (almost 8 times!)
  - with wave tracing, 160s (more than 14 times!)
Results: Reconfiguration times after 3rd Refinement

- Reconfiguration times considered in simulation
- In this case, +3.5ms (not noticeable in time performance)
- Simulation performance similar
Results: 4th Ref. Step

- Around 1500 VHDL lines of synthesizable code
- Post-synthesis validation
Conclusions

- Systematic Refinement of A-HetSC HAPs into OSSS+R constructs
- Linking abstract adaptive model with efficient implementation as DRHW
- Refinement in the SystemC domain
  - Smooth approach
  - High-Level analysis

- Future: automation