Automated code generation for Early AURIX™ VP

Pratheek Mahesh and Dineshkumar Selvaraj
Infineon Technologies
Introduction

• Virtual Prototype (VP) - Shift left platform expected to be available quite early in SoC development cycle.

• With the new architectural enhancements, protection mechanism is an essential feature of the next generation AURIX™ safety and security architecture.

• Enables application software to restrict read and write accesses to each slave function to a desired set of masters.

• In addition it supports CPU hypervisor functions with access protection control based on Virtual Machine ID.
Advantages

- Support multiple PROT/APU mechanism.
- Scalable to all the derivative devices.
- Flexible to extend/override in special cases. E.g. Dynamic configuration of resource allocation.
- Minimum time and effort.
- Easy integration into current VP development flow.
Implementation Details

Centralized implementation of UBS provides protection mechanism API’s.

Easy integration to the existing development flow.

Generated protection check files

Centralized implementation of PROT/APU
Implementation Details

Automatic generation of protection check functions at each resource level (Registers, memory).

Generated protection check functions, uses the API’s provided by the Centralized implementation.

Flexible as the API’s can be used to extend the implementation. E.g. Dynamic configuration of resource allocation.

Functions Implementing Protection Mechanism

```c
bool checkAccessProtocol_RegX(UbsKernel) {
    UbsKernel->isValidMasterSatisfied(ID_X)
    UbsKernel->isPROTSatisfied(REG_X)
    UbsKernel->checkAccess();
}

bool checkAccessProtocol_RegX1(UbsKernel) ...
bool checkAccessProtocol_RegX2(UbsKernel)
```

API’s Provided by UBS Kernel

```c
bool isValidMasterSatisfied();
bool isPROTSatisfied(); ...
bool checkAccess();
```
## Results Table

<table>
<thead>
<tr>
<th>Activity</th>
<th>Sub activity</th>
<th>No of days</th>
<th>Total days</th>
<th>Final effort (days)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementation</td>
<td>Spec study Development Reviews</td>
<td>4</td>
<td>4 * 41 = 164</td>
<td>164</td>
</tr>
<tr>
<td>Testing</td>
<td>Spec study Test case development Reviews</td>
<td>4</td>
<td>4 * 41 = 164</td>
<td>164</td>
</tr>
<tr>
<td>XML Refresh</td>
<td>XML Migration</td>
<td>10%</td>
<td>131(4 * [10 % of 328])</td>
<td>131</td>
</tr>
<tr>
<td>Bugs Found in the concept</td>
<td></td>
<td>10 bugs</td>
<td>10 days (Feed back to concept)</td>
<td>10</td>
</tr>
<tr>
<td>Deployment</td>
<td>Initial issues</td>
<td>1</td>
<td>41</td>
<td>-41</td>
</tr>
<tr>
<td>Code generator development</td>
<td>Spec study Scripting Verification</td>
<td>60</td>
<td>60</td>
<td>-60</td>
</tr>
</tbody>
</table>

**Total Effort saved**: 369
Conclusion

Automation resulted in significant development cost saving:

- Significant development cost (approximately 369 man days) was saved because of the automation process.
- In addition it will also help in reducing the time and effort for development of other derivative VP of the same family.

Early feedback to the concept on the new protection mechanisms and also on the correctness of single source XML
Questions