Technology and Design Tools for Portable Software Development for Embedded Systems

Boris Sedov, Alexey Syschikov, Vera Ivanova
{boris.sedov, alexey.syschikov, vera.ivanova}@guap.ru
Why do we need such technology?

1. “Two-in-one” developer is required:
   - skilled domain experts + skilled programmer

2. Contradictive requirements to hardware platforms

3. The development of an algorithm and program should be started before the selection of a specific platform configuration

4. Hardware platforms become out-of-date rapidly, but computation tasks that are executed on them changes rarely
The life cycle of programs for embedded systems

- Development
- Simulation
- Code generator

Platform 1 → Platform 2 → Platform 3 → Simulation → Development
Visual Development Environment (VIPE)
Cognitive advantages:

• clear view of development process
• traceability of the dependency graph
• calculations management structure
• natural parallelism
• potential pipelining
Asynchronous Growing Processes (AGP) formal computational model

AGP defines:
- language syntax
- semantic of language objects
- control units

AGP provides:
- formal verification
- debugging
- portability

Program scheme — oriented graph

$V_p$ – operator vertex,
$V_d$ – data-object vertex,
$W/R/RE$ – arcs (links) marking
Visual Programming Language (VPL)

- Separation of design and programming
- Easy program development
- Easy changes in program structure
- No direct programmer influence on a program scheme
- Local appearance of code errors
- Possibility of auto-documenting of the program scheme
- Effective program maintenance during the whole lifecycle
- Decreasing of errors possibility without sacrificing program obviousness
- Flexibility and ease-of-change on any design stage
Visual Programming Language (VPL)
Domain specific programming

- DSL libraries provide convenience of design within an application domain
- Easy re-use of development results
Interactive tools

Development process support tools:
- scheme validation
- verification
- interactive debugging etc.
Early estimation and evaluation

[Diagram showing the process of early estimation and evaluation, including steps such as Visual development environment, Validation, Verification, Debugging (interactive), XIR Executable specification, Platform description, Programmer, Library functions, Virtual simulator, Platform simulator, Optimizer, Code generators, Tables, Graphics, Statistics, Characteristics, Graphics, Statistics, Sequential C/C++ Assembler, Parallel OpenMP Threads.]
Virtual simulator

Allows estimate:
- maximal possible parallelism
- computation space (amount of computations, used memory etc.)
- computation time
- amount and intensity of data exchange
Platform simulator

Allows estimate:

- requirements to embedded system cores performance
- requirements to embedded system cores memory
- computation cores occupation for the different allocation variants and occupation balance
- amount and intensity of data exchange
- effectiveness of hardware loading
- bottlenecks of hardware platform, program and tasks allocation
Deployment to target platforms

- ANSI C
- C++
- Parallel OpenMP
- Parallel threads
- MPI
- Assembler MIPS, DSP
- Inhomogeneous systems
Conclusions

- Supporting of a whole embedded software lifecycle
- The formal model based development ensures the equal execution of a debugged program in any runtime environment
- An algorithm is created and debugged once Parallel threads
- Scheme optimization for platform granularity
- Supporting of heterogeneous embedded systems (hardware blocks, accelerators, DSP)
Thank you!