An Overseer Control Methodology for Data Adaptable Embedded Systems

Sean Whitsitt, Jonathan Sprinkle, Roman Lysecky

October 1st 2012

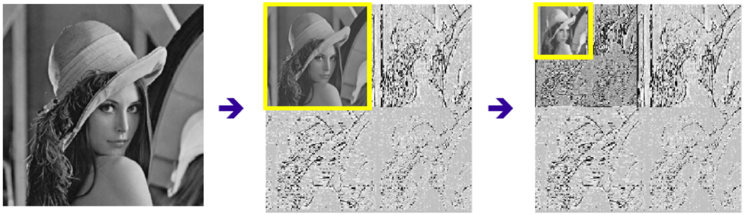
THE UNIVERSITY OF ARIZONA®

ARIZONA®

Outline

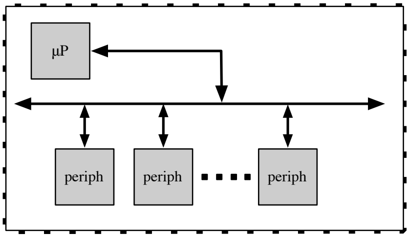
Problems!

- How do we speed up complex algorithms?
- How do we shrink the size of hardware necessary for these algorithms?
- Example
  - The speedup on small images may be minimal
  - Consider larger images (medical, satellite)
  - Each image may take hours to process



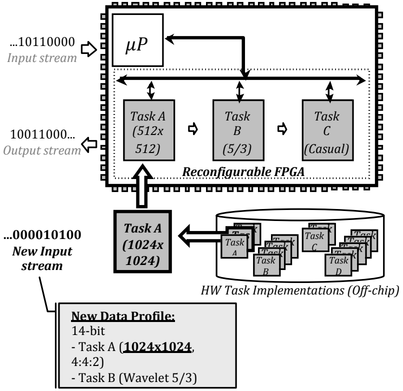
FPGAs

- Field Programmable Gate Arrays
- Can perform specific tasks in hardware
- Power consumption is decreasing
- Processing speed is increasing
- Available area is increasing
- Reconfigurable



Data Adaptable Reconfigurable Embedded Systems

- DARES for short.
- Address real-estate limitations.



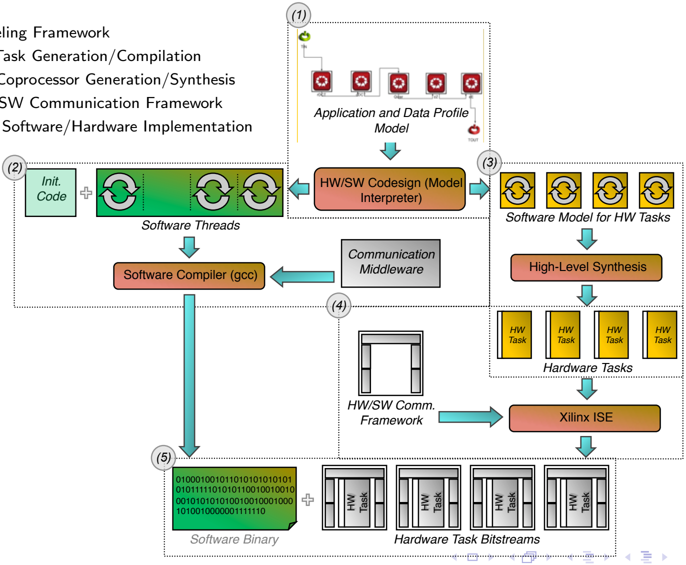
New Data Profile:
14-bit
- Task A (1024x1024, 4:4:2)
- Task B (Wavelet 5/3)

[1]

[1] S. Mahadevan. et al. Hardware/software communication middleware for data adaptable embedded systems.

Tools

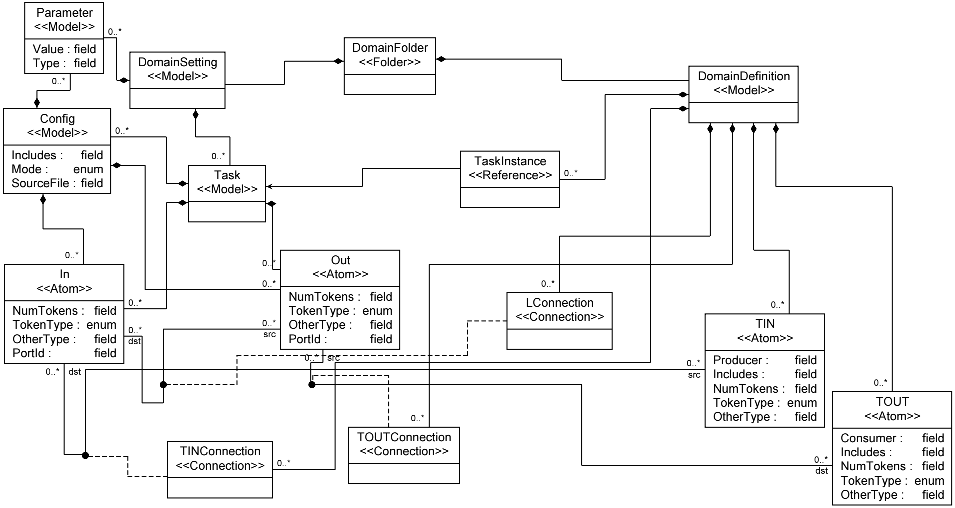
1. Modeling Framework
2. SW Task Generation/Compilation
3. HW Coprocessor Generation/Synthesis
4. HW/SW Communication Framework
5. Final Software/Hardware Implementation



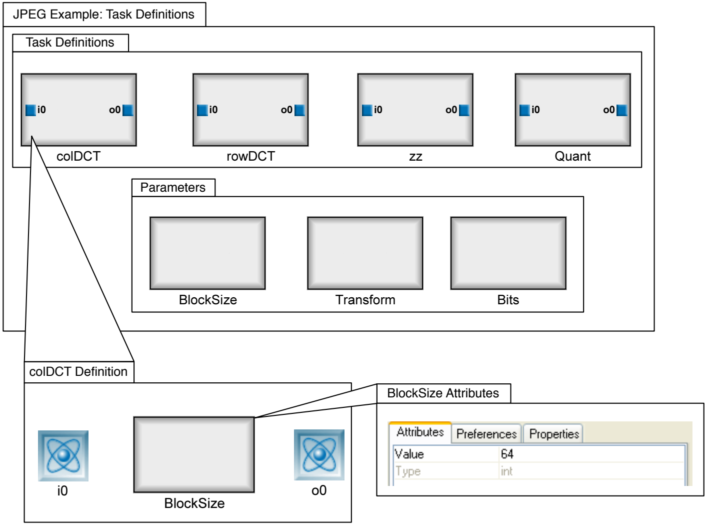
What is new?

- The setup
  - FIFOs (connections) between tasks
  - Configurations of the tasks
  - **Parameters that define those configurations**
- Runtime
  - **Switching between configurations**
  - **Routing data between hardware and software**
  - **“Optimal” configurations**

Metamodel

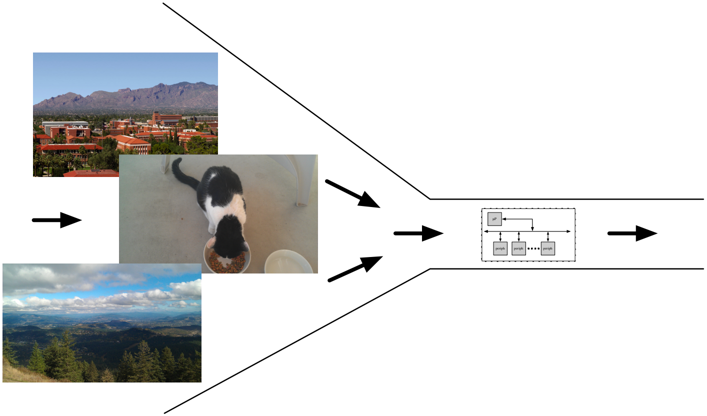

Parameter <<Model>>
Value : field
Type : field
DomainSetting <<Model>>
DomainFolder <<Folder>>
DomainDefinition <<Model>>
Config <<Model>>
Includes : field
Mode : enum
SourceFile : field
Task <<Model>>
TaskInstance <<Reference>>
In <<Atom>>
NumTokens : field
TokenType : enum
OtherType : field
PortId : field
Out <<Atom>>
NumTokens : field
TokenType : enum
OtherType : field
PortId : field
LConnection <<Connection>>
TIN <<Atom>>
Producer : field
Includes : field
NumTokens : field
TokenType : enum
OtherType : field
TOUT <<Atom>>
Consumer : field
Includes : field
NumTokens : field
TokenType : enum
OtherType : field
TINConnection <<Connection>>
TOUTConnection <<Connection>>
0..*
src
dst

Example

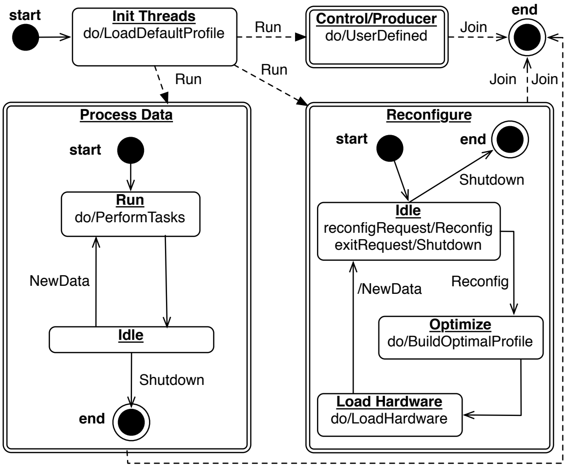


Value	64
Type	int

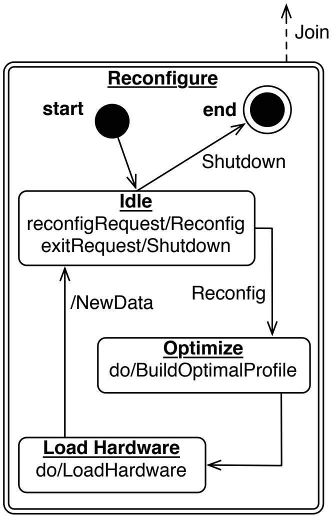
Runtime Control



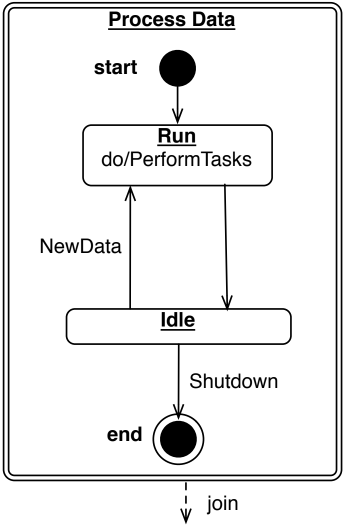
Statechart


start
Init Threads
do/LoadDefaultProfile
Run
Control/Producer
do/UserDefined
Join
end
Run
Run
Join
Join
Process Data
start
Run
do/PerformTasks
NewData
Idle
Shutdown
end
Reconfigure
start
end
Shutdown
Idle
reconfigRequest/Reconfig
exitRequest/Shutdown
/NewData
Reconfig
Optimize
do/BuildOptimalProfile
Load Hardware
do/LoadHardware

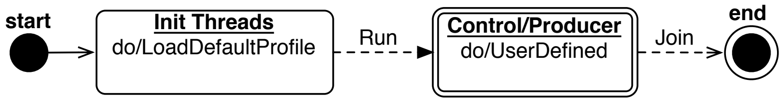
Reconfiguration



Data


Process Data
start
Run
do/PerformTasks
NewData
Idle
Shutdown
end
join

Overseer


start
Init Threads
do/LoadDefaultProfile
Run
Control/Producer
do/UserDefined
Join
end

Users

- Unknowns
  - How should the parameters be set?
  - When should reconfiguration occur?
  - What is “new” data?
- We can’t make assumptions about data organization.
- We can’t make assumptions about task functionality.
- Solution: let the user decide!

User Support

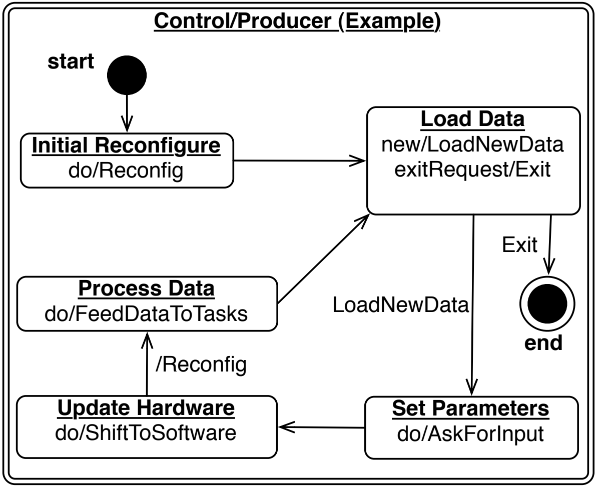
- DaresML provides the user with support through pragmas
  - IFDEFs
  - reconfigure system
  - set parameter
  - reset parameters
- Similar to the C/C++ preprocessor

Code!

```
#pragma DARES_PARAMETERS
#pragma DARES_INCLUDES
#pragma DARES_INPUT_STREAMS_DEFINITIONS
#pragma DARES_OUTPUT_STREAMS_DEFINITIONS

void #pragma DARES_HARDWARE_FUNCTION_NAME() {
    int var1, var2;
    #pragma DARES_READ_SINGLE_FIFO(i0, var1)
    #pragma IFDEF Hardware
        // this is hardware code
    #pragma ELSEIFDEF PARAM_WIDTH
        // this has a width
    #pragma ELSEIFDEF Software
        // this is software code
    #pragma ELSEDEF
        // this is the else portion
    #pragma ENDDEF
        // Begin Computation Logic
        // End Computation Logic
    #pragma DARES_WRITE_SINGLE_FIFO(o0, var1)
}
```

User Designed Producer Example



Future Work

- Middleware
  - Better reconfiguration support
  - Lower overhead when using software
- DaresML
  - integrating more extraneous setup
  - Optimal instead of “Optimal”
  - JPEG2000

Conclusion

- Parameterized task switching
- User guided task switching
- Overseer methodology
  - Control the minutia of task switching
  - Several threads working together
  - “Optimal” algorithm support

Thank you! Are there any questions?