

Automatic Code Generation For a Production Radar Signal Processor Using Gedae



Richard Hunt AMS

Automatic Code Generation Session

ESS 2004

Contents

- Project / Gedae Background
- Pilot Study
- Post Pilot Work
- Conclusions

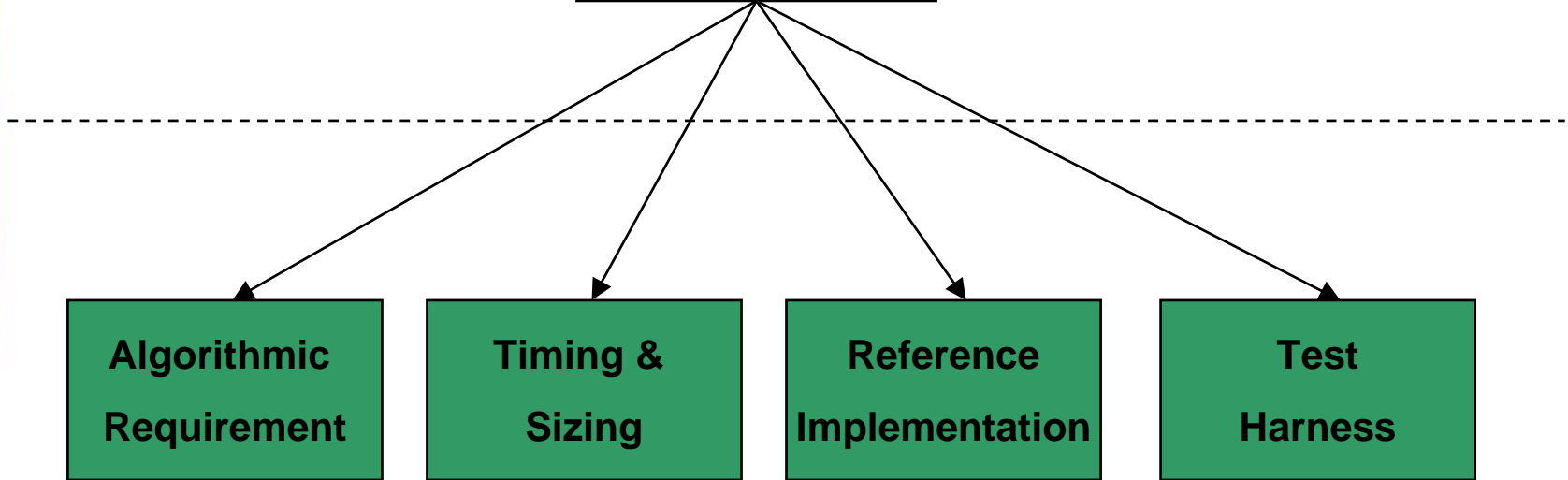
Project Background

- **Naval radar signal processor unit**
 - Existing system uses 1980's technology
 - Maintain existing functionality
 - Technology refresh & some enhancements
- **Replace with Quad G4 Processor Card**
 - Complete software re-write to re-target application
 - VxWorks with VSIPRO maths library
 - In-House inter-processor communications library
 - Solaris host

Existing Gedae Use

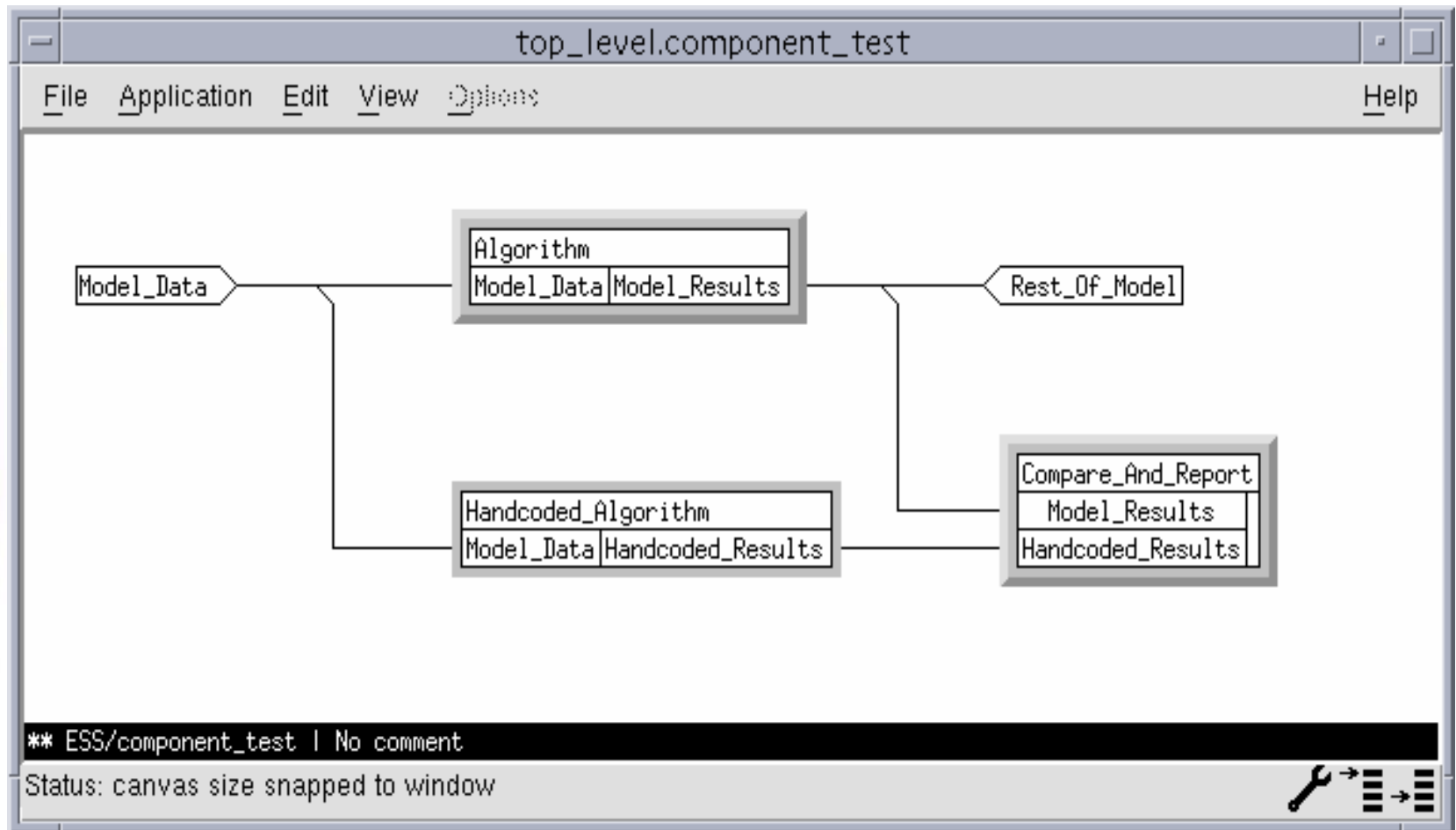
Systems

SPU Algorithmic
Model
in Gedae



Software

Component Test



Gedae Background

- **Why did we start exploring automatic code generation ?**
 - Monthly 'Signal Processor Opportunities and Threats' (SPOT) review
 - Project resource / schedule constraints
 - Questions asked at SPU Preliminary Design Review
 - PV evaluations providing confidence in capability of Gedae toolset

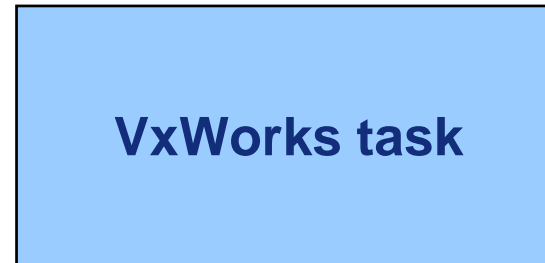
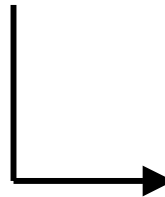
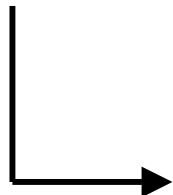
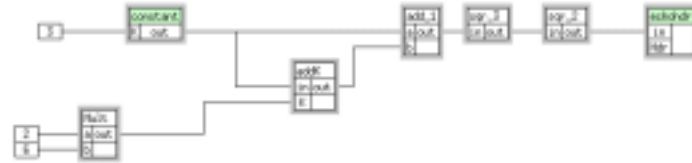
Pilot Study

- Pilot for automatic code generation of sub-task in deliverable application
 - Begun 8 months ago
 - Investigate embedding Gedae code into hand coded software
 - One / Multiple executables
 - Focussed on 3 Algorithm units
 - Technical (can it be done?)
 - Performance (latency)

What did we do?

- Produce a trivial component to ensure we are able to create a Gedae executable running as a sub-task
- Run the executable as a VxWorks task
- Repeat the process, this time using an actual Algorithm (Algorithm A)
- Implement a method of feeding data in and out of the Gedae executable from an external source
- Verify that the results from the self-contained Gedae executable are correct
- Produce a set of timings from the self-contained Gedae executable

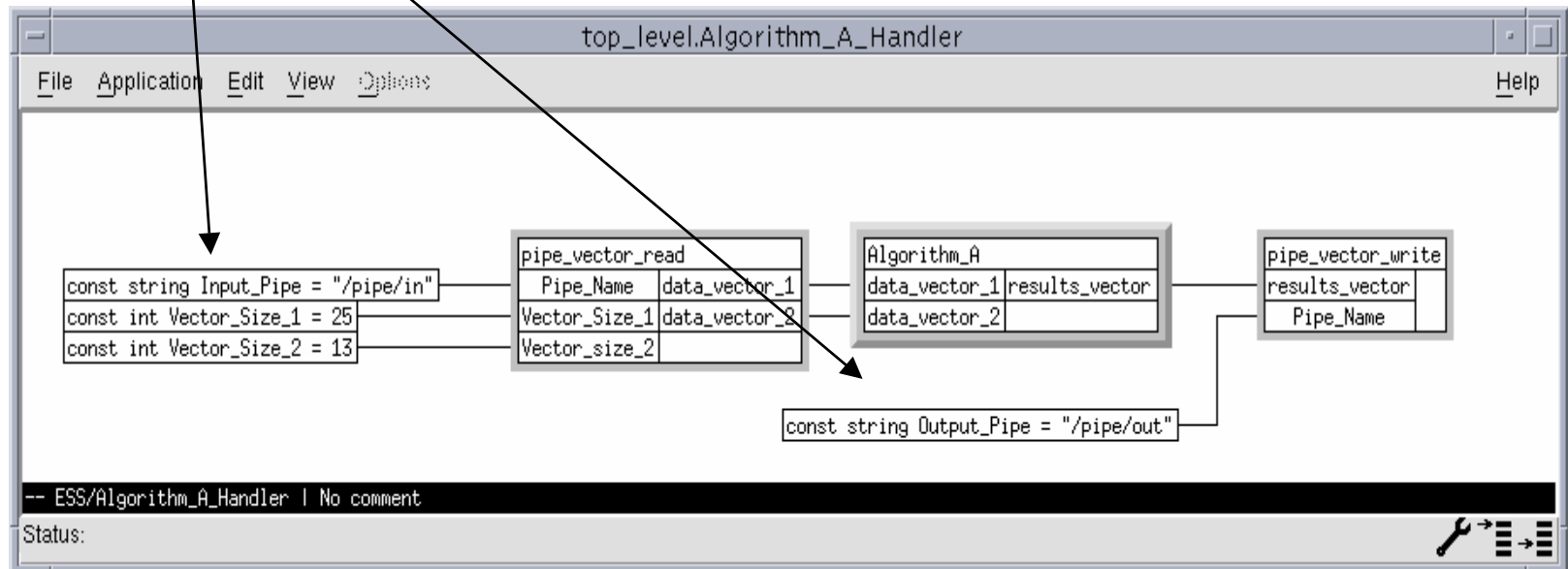
Steps



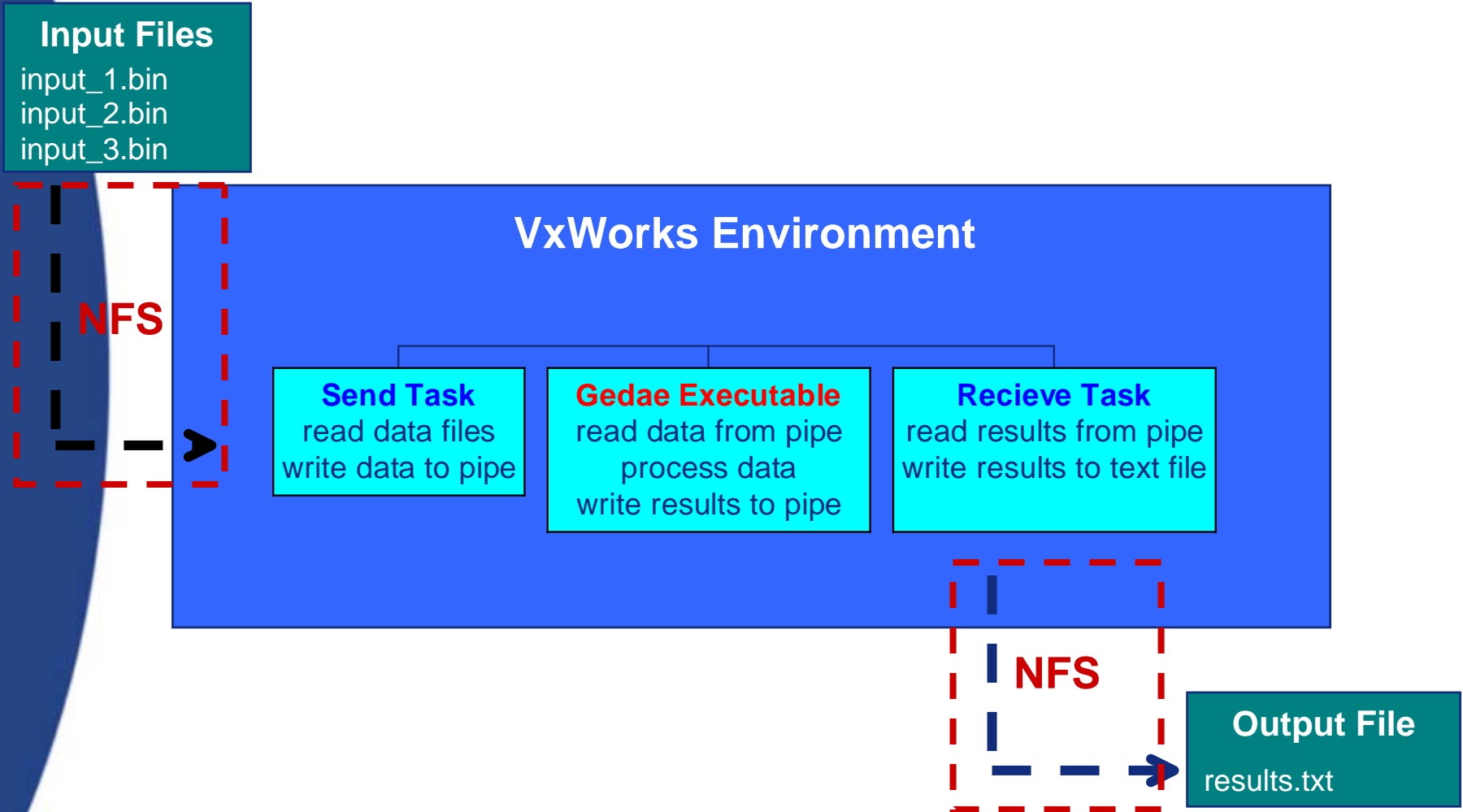
Using our Algorithm A

- Once we had established a method for creating an executable and running it as a task within VxWorks we moved on to Algorithm A
- Algorithm A is more complex
- Initially generated dummy input data by using Gedae source boxes
- Then fed with data files captured from model

Feeding data in and out of Gedae from an external source



Running it all from VxWorks



Verifying the Results

- Compare output of host Gedae model with embedded code
- A few differences due to number representation / truncation of floating point numbers

Initial Timings

Times in μsec per batch

Mode	1	2	3
Min	118	154	230
Median	120	180	258
Mean	122	431	542
Max	137	29842	61771

Timing Breakdown

Median times in μsec for mode 3

• TX /RX	40
• Control Processing	80
• Algorithm	140
• Total	260

Allocated Timing Budget 193
(Algorithm Only)

Optimisation

Timing points within Gedae task

Times in μsec per batch (Mode 3)

Min	148
Median	154
Mean	159
Max	382

Max time is still much larger than median

Optimisation

Increased Gedae task's priority

Times in μsec per batch (Mode 3)

Min	148
Median	155
Mean	155
Max	175

The maximum occurs when switching modes

Further Algorithms

- Candidates identified through knowledge of achievable Gedae generated code efficiency
- Algorithm B
 - Complex Filters
 - Little vector processing
 - Asynchronous data sampling
 - Possible candidate for automatic code generation
- Algorithm C
 - Processing large vectors of data
 - Gedae not expected to be efficient as no AltiVec optimised maths library
 - 'Control' algorithm

Algorithm B Result

(Initial unoptimized Gedae implementation)

Times in μsec per batch

Min	3055
Median	3093
Mean	3199
Max	4938

Allocated Timing Budget 500

Algorithms B Analysis

- Gedae model structured to show requirement
- Inefficient (very) Gedae implementation of complex filters
- Gedae application model restructured to improve execution efficiency
- Re-runs give times well within budget (80 μ sec)

Algorithm C Results

Times in μsec per batch

Mode	1	2
Min	57047	55738
Median	57227	57187
Mean	57342	57266
Max	58727	60319

Actual coded time 1000 μsec

- Results as expected for AltiVec maths library versus raw 'C'

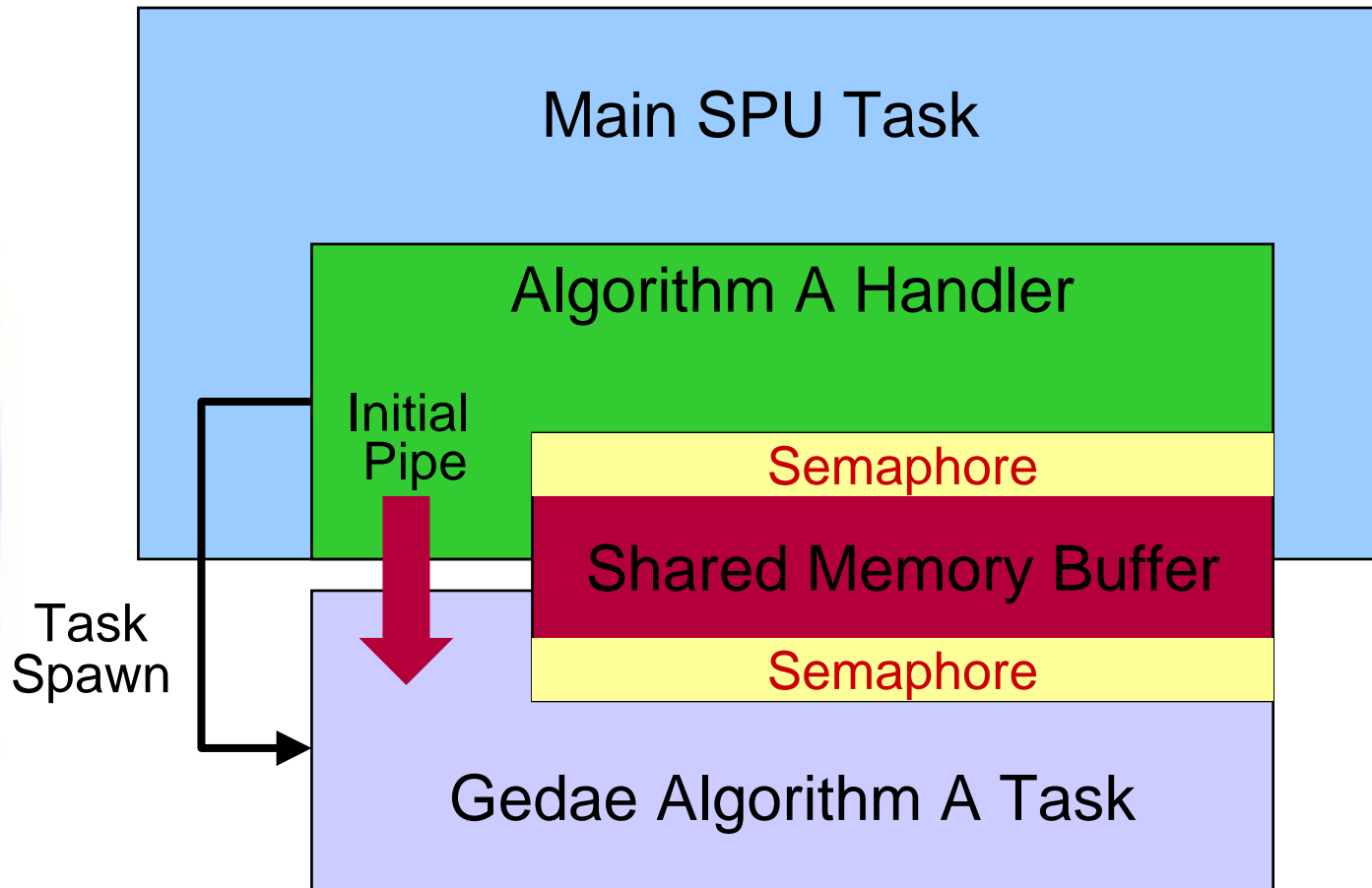
Pilot Outcome

- Algorithm 'A' selected as a sub-task of the deliverable application to be automatically coded using Gedae
 - Why?
 - It is self contained
 - It has limited requirements on math libraries
 - Large development effort to hand code
 - Gedae latency within budget

Post-Pilot Work

- **Explored debug capabilities**
 - Needed to pinpoint source of errors
 - Gedae Inc have added post-mortem routines
 - Can still use normal VxWorks tools
- **Improved interfacing**

Improved Interfacing



Post-Pilot Work

- **Talked with BAE Systems Edinburgh**
 - Advanced Gedae user
 - Developing Eurofighter Captor radar in Gedae
 - Much more demanding application
 - Gave extra confidence & lots of ideas
- **Software review of Gedae workproduct**
 - Walkthrough of flowgraphs
 - Workproduct review of 'C' code primitives
 - Re-organisation for maintainability
 - HTML tool & line 'counter' produced

Current Work

- **Developing hand coded Gedae interface**
- **Further model development**
 - Gedae side of interface
 - Changes for maintainability
- **Further automatic code generation**
 - Considering Algorithm B
 - Can wait for lessons learned from Algorithm A development

Conclusions

- **Automatic coding of algorithms viable**
 - Even using non-optimised library
- **Possibility of significant savings**
 - Hand coding of Algorithm A was budgeted at sixteen man months
 - Will not realise all the potential savings
 - Six man months of effort still required for interface & maintainability work
- **Less opportunity for defects to be introduced**
- **Easier implementation of requirement changes**
- **Provides basis for automatic code generation on future projects**



Eastwood House, Glebe Road
Chelmsford, Essex CM1 1QW
England, United Kingdom
T+44 (0) 1245 702702
F+44 (0) 1245 702700

Via Tiburtina Km 12,400
00 131 Roma, Italia
T+39 06 41501
F+39 06 4131133

www.amsjv.com

The copyright in this document is owned by
the Subsidiaries of AMS NV and may not
be reproduced without written consent.

© AMS Limited 2004