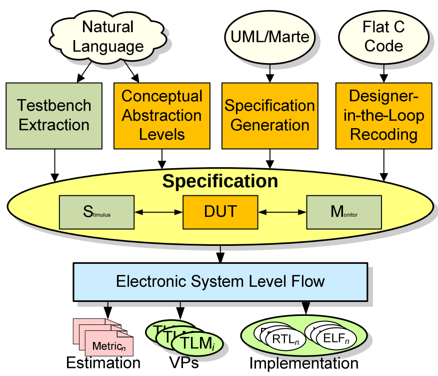
Designer-in-the-Loop Recoding to Create Safe Parallel ESL Models

Tutorial SD1: High-Level Specifications to Cope with Design Complexity



Rainer Dömer
CECS
University of California, Irvine
USA

Outline

- Embedded System Design Challenge
  - Productivity Gap
  - System Level Modeling Concepts
- Computer-Aided Recoding
  - Introduction and Motivation
  - Recoding Transformations
  - Recoding Analysis
- Prototype Implementations
  - Interactive Source Recoder
  - Eclipse-based Recoding Platform
- Experiments and Results
  - Classroom Case Studies
- Conclusions

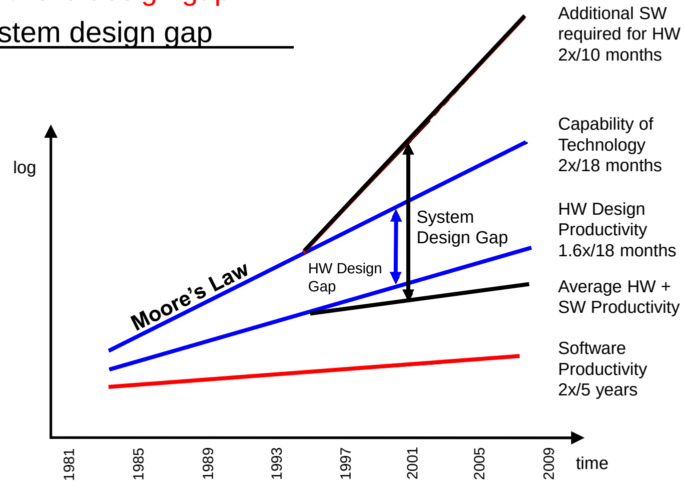
Embedded System Design

- Productivity Gap

  Hardware design gap

  \+ Software design gap

  = System design gap



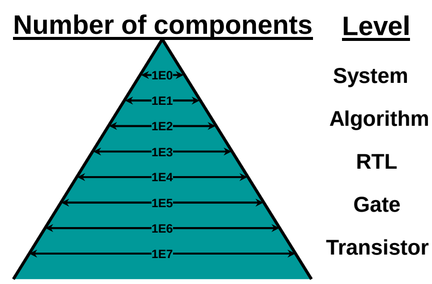
(source: *"Hardware-dependent Software"*, Ecker et al., 2009)

Embedded System Design

- How can we overcome the productivity gap?

  International Technology Roadmap for Semiconductors (ITRS) 2004: *higher-level abstraction and specification* is the first promising solution

- System Level Design
  - Unified HW and SW design
  - Higher level of abstraction
    - Fewer, more complex components
    - Maintain system overview
      - Without overwhelming details
    - Compose a system of algorithms
  - System Level Design Languages
    - SpecC [Gajski et. al, 2000]
    - SystemC [Groetker et. al, 2002]

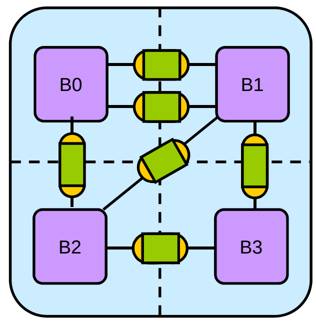


Source: "System Design: A Practical Guide with SpecC", 2001

Embedded System Design

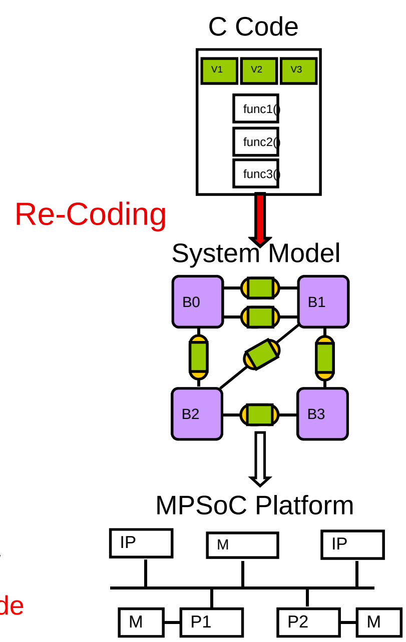
- System Level Modeling
  - Abstract description of a complete system
  - Hardware + Software
- Key Concepts in System Modeling
  - Explicit Structure
    - Block diagram structure
    - Connectivity through ports
  - Explicit Hierarchy
    - System composed of components
  - Explicit Concurrency
    - Potential for parallel execution
    - Potential for pipelined execution
  - Explicit Communication and Computation
    - Channels and Interfaces
    - Behaviors / Modules

System Model



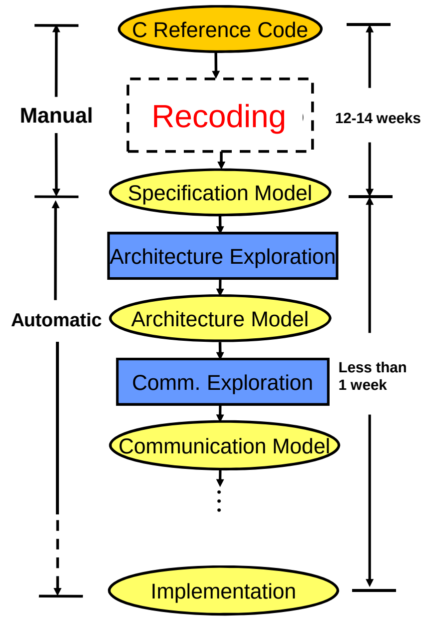
Computer-Aided Recoding

- Embedded System Design Flow
  - Input: System model
  - Output: MPSoC platform
- Actual Starting Point
  - C reference code
  - Flat, unstructured, sequential
  - Insufficient for system exploration
- Need: System model
  - System-Level Description Language (SLDL)
  - Well-structured
    - Explicit computation, explicit communication
    - Potential parallelism explicitly exposed
  - Analyzable, synthesizable, verifiable
- Research: Automatic *Re-Coding*
  - How to get from flat and sequential C code to a flexible and parallel system model?



Motivation

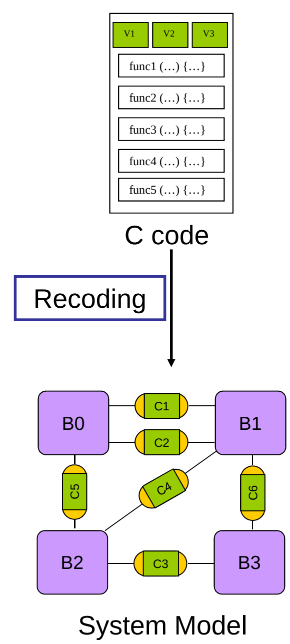
- Extend of Automation
  - Refinement-based design flow
  - Automatic
    - Specification model down to implementation
    - Example: SCE (mostly automatic)
    - MP3 decoder: less than 1 week
  - Manual
    - Source code transformations
    - C reference code to SpecC specification model
    - MP3 decoder: 12-14 weeks!
- Automation Gap
  - 90% of overall design time is spent on re-coding!
- Proposal: Automatic Recoding



Source: *System Design: A Practical Guide with SpecC*

Problem Definition

- How to get from flat, sequential C code to a flexible, parallel system model?
- Recoding
  - Create structural hierarchy
  - Partition code and data
    - Expose concurrency (parallelize/pipeline)
  - Expose communication
  - Eliminate pointers
  - Make the code compliant to the design tools, …
- Our approach
  - Computer-Aided Recoding
    - Interactive source code transformations



Computer-Aided Recoding

- Complete Automation is Infeasible!
  - Today's parallelizing compilers are largely ineffective
    - Heterogeneous architectures
    - Complexity of embedded applications
    - Hard problems (eliminating pointers, exposing parallelism, etc.)
  - Modeling requires understanding of the application
  - Recoding is not a monolithic transformation
    - Multiple transformations in application-specific order

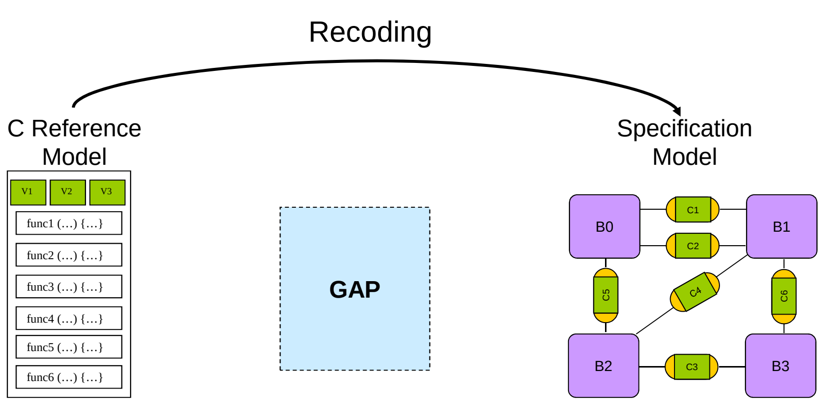
➢Interactive Approach
  - "Designer-in-the-loop"
  - Designer can utilize application knowledge

- *Designer-controlled* Transformations
  - Designer makes decisions
  - Tool automatically transforms the source code

Tutorial SD1, ASPDAC '14, Singapore January 20, 2014 9

Overcoming the Specification Gap

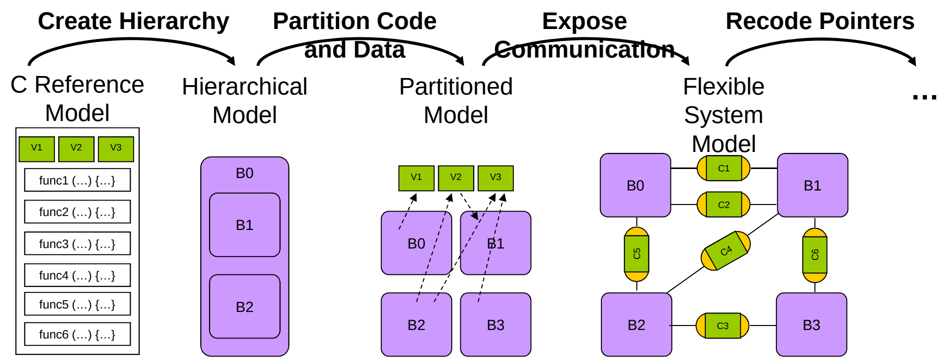
- Recoding Transformations



Tutorial SD1, ASPDAC '14, Singapore January 20, 2014 10

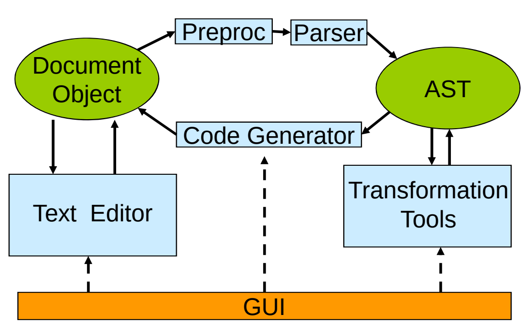
Overcoming the Specification Gap

- Recoding Transformations
  - Creating structural hierarchy [ASPDAC'08]
  - Code and data partitioning [DAC'07]
  - Creating explicit communication [ASPDAC'07]
  - Recode pointers [ISSS/CODES'07]



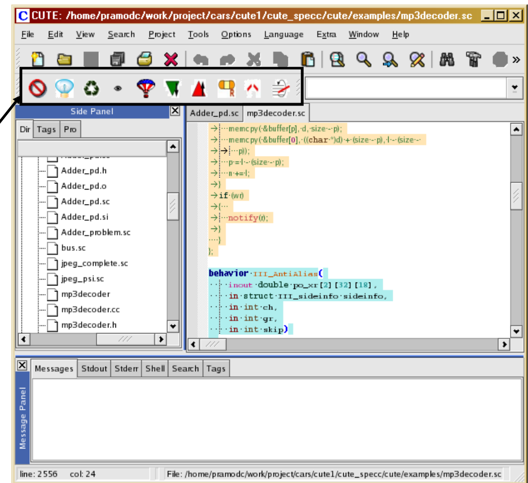
Prototype 1: Interactive Source Recoder

- Prototype Implementation (by P. Chandraiah)
  - Integrated Development Environment (IDE)
- *Cute* tool is a union of
  - Text editor
  - Abstract Syntax Tree (AST)
  - Parser
  - Transformations
  - Code generator



Prototype 1: Interactive Source Recoder

- Interactive Environment
  - Scintilla + QT + AST + Transformations
- Basic editing
  - Syntax highlighting
  - Auto-completion
  - …
- Recoding Transformations
  - Dependency analysis
  - Code and data splitting
  - Variable re-scoping
  - Port insertion
  - …



Tutorial SD1, ASPDAC '14, Singapore January 20, 2014 **13**

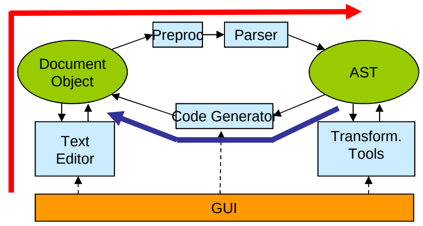
Prototype 1: Experiments and Results

- We have conducted various sets of experiments
- Goals
  - Responsiveness of the "compiler in the editor"
  - Estimated Productivity Gains
    - Extrapolation based on the number of lines of code changed
  - Measured Productivity Gains
    - Class of graduate students
- Design examples
  - GSM Vocoder (voice codec in mobile phones)
  - MP3 Decoder (audio decoder, e.g. iPod)
    - Fixed-point version
    - Floating-point version
  - JPEG Encoder (image encoder, e.g. digital camera)
  - …

Tutorial SD1, ASPDAC '14, Singapore January 20, 2014 **14**

Prototype 1: Responsiveness

- Why measure Responsiveness?
  - To check feasibility
- Responsiveness
  - Response to designer actions
  - Time to synch AST
    - On editing
  - Time to synch Editor
    - On transformation
  - Depends on the size of the AST
- Design examples
  - JPEG, MP3, GSM
  - ➢ << 1 sec (on a 3 GHz Linux PC)
  - File I/O overhead (20%)



Operation	Simple	JPEG	MP3	GSM
Lines of code	174	1642	7086	7492
Objects in AST	1073	5338	31763	26009
Synch AST	**0.15 secs**	**0.19 secs**	**0.68 secs**	**0.55 secs**
Synch Editor	**0.16 secs**	**0.20 secs**	**0.73 secs**	**0.59 secs**

Prototype 1: Experimental Results

- Productivity Gain
  - Creating structural hierarchy
    - Manually
      - estimation
    - Automatically
      - measured
- Results
  - Manual time
    - ➢ weeks
  - Recoding time
    - ➢ minutes

Properties	JPEG	Float-MP3	Fix-MP3	GSM
Lines of C code	1K	3K	10K	10K
C Functions	32	30	67	163
Lines of SpecC code	1.6K	7K	13K	7K
Behaviors created	28	43	54	70
Re-Coding time	≈ 30 mins	≈ 35 mins	≈ 40 mins	≈ 50 mins
Manual time	1.5 weeks	3 weeks	2 weeks	4 weeks
Productivity gain	120	205	120	192

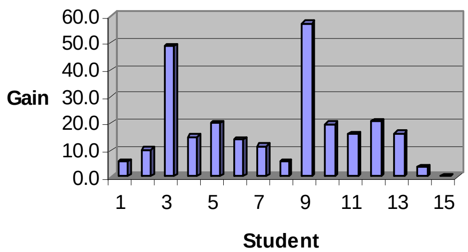
[ASPDAC'08]

➢ Significant productivity gains!

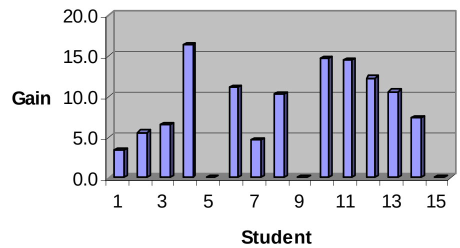
Prototype 1: Productivity Gains

- Measured Productivity Gains
  - Class of 15 graduate students
  - Recode an MP3 design example
    - Manually (given detailed instructions)
    - Automatically (using the Source Recoder)
- Results

Creating Structural Hierachy



Pointer Recoding

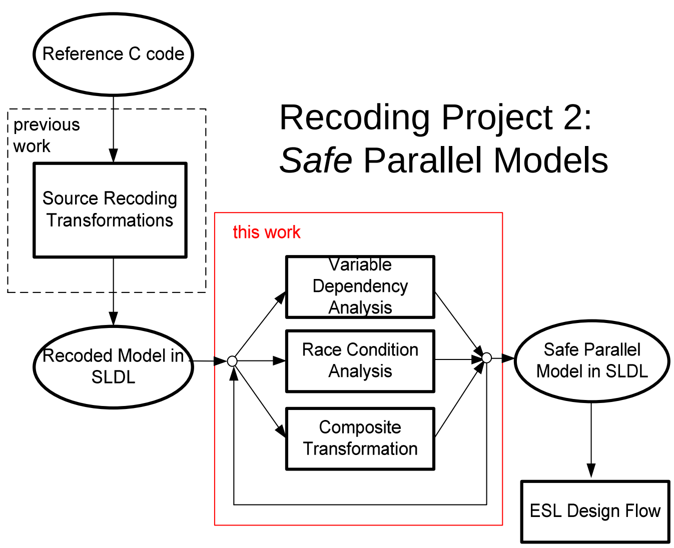


  - Productivity factors vary, but show significant gains!

Recoding for Safe Parallel ESL Models

Recoding Project 1: Creation of Parallel Models

Recoding Project 2:
Safe Parallel Models



Recoding for Safe Parallel ESL Models

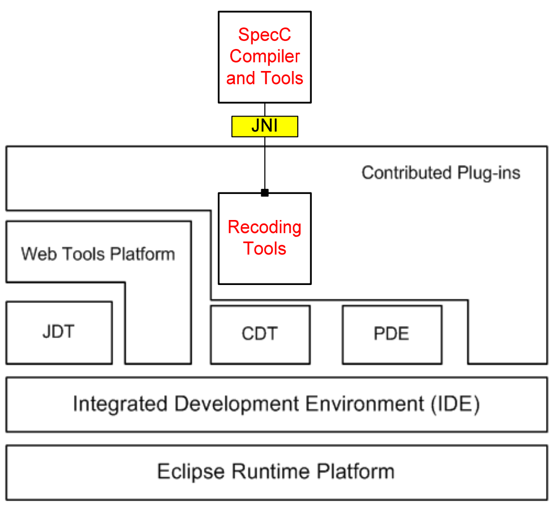
- Recoding Project 1: Creation of Parallel Models
  - Prototype 1: Interactive Source Recoder
    - by Pramod Chandraiah
  - Focus on designer-controlled source code transformations

➢ Recoding Project 2: Recoding for Safe Parallelism
  - Prototype 2: Eclipse-based Recoding Platform
    - by Xu Han, Weiwei Chen
  - Focus on Advanced Model Analysis
    - ➢ Case Study on a Canny Edge Decoder
    - Variable Dependency Analysis
    - Static Parallel Access Conflict Analysis
    - Race Condition Analysis

Tutorial SD1, ASPDAC '14, Singapore January 20, 2014 19

Eclipse-Based Recoding Platform

- Eclipse Framework
  is an extensible platform to build IDEs

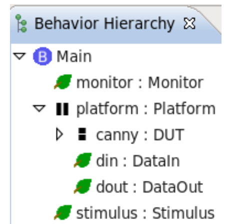


Eclipse SDK: source: Carlson, Eclipse Distilled, 2005

Tutorial SD1, ASPDAC '14, Singapore January 20, 2014 20

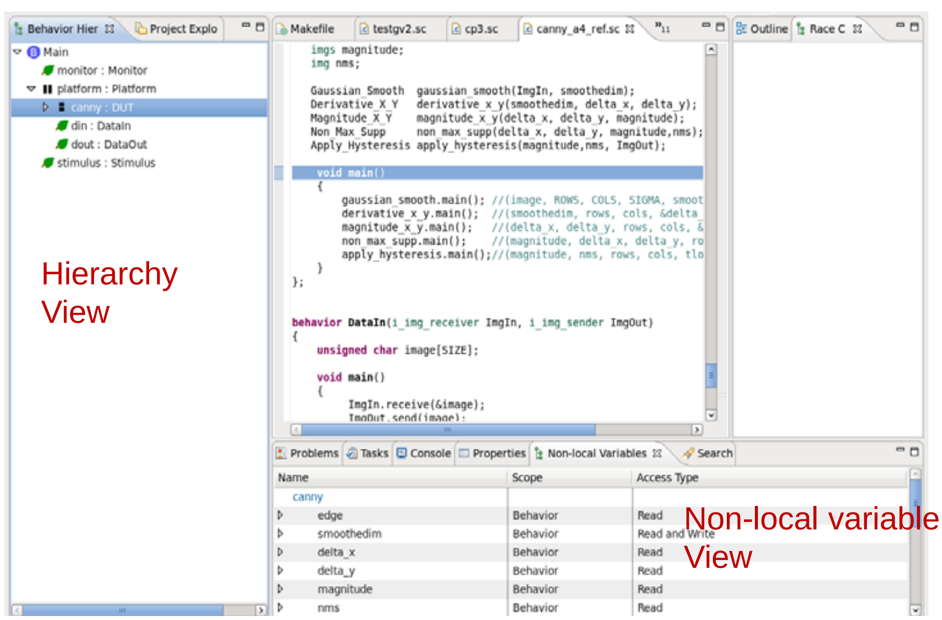
Eclipse-Based Recoding Platform

- Eclipse Framework:
  Integrated Recoding Operations
  - Automatic Compiling
    - Compilation in the background
    - Static design analysis
    - Variable dependency analysis
  - Hierarchy View
    - Behavior hierarchy display
    - Behavior hierarchy navigator
    - Context menu for advanced analysis
  - Non-local Variable View
    - Dependent variables display
    - Conflicting variable access display



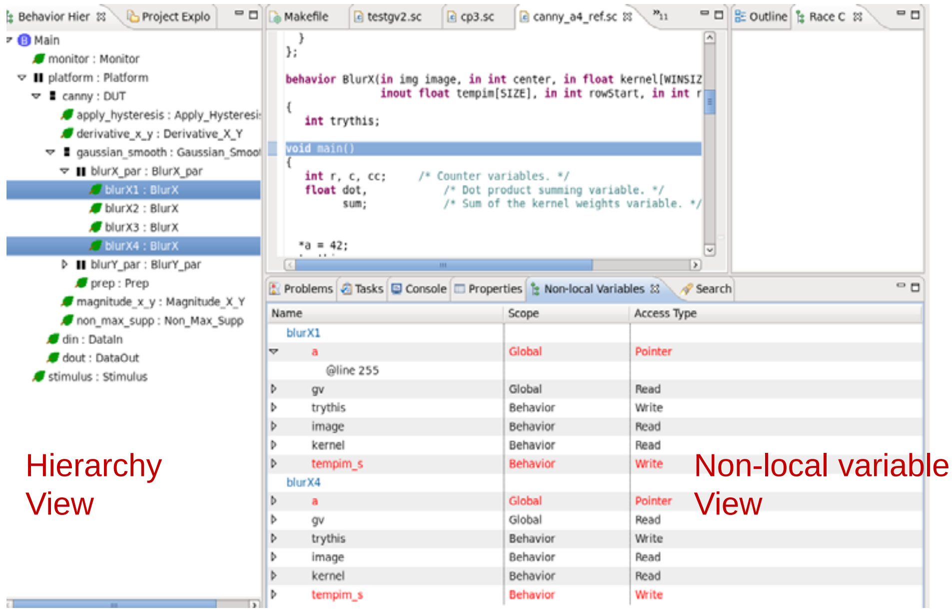
Tutorial SD1, ASPDAC '14, Singapore January 20, 2014 21

Eclipse-Based Recoding Platform

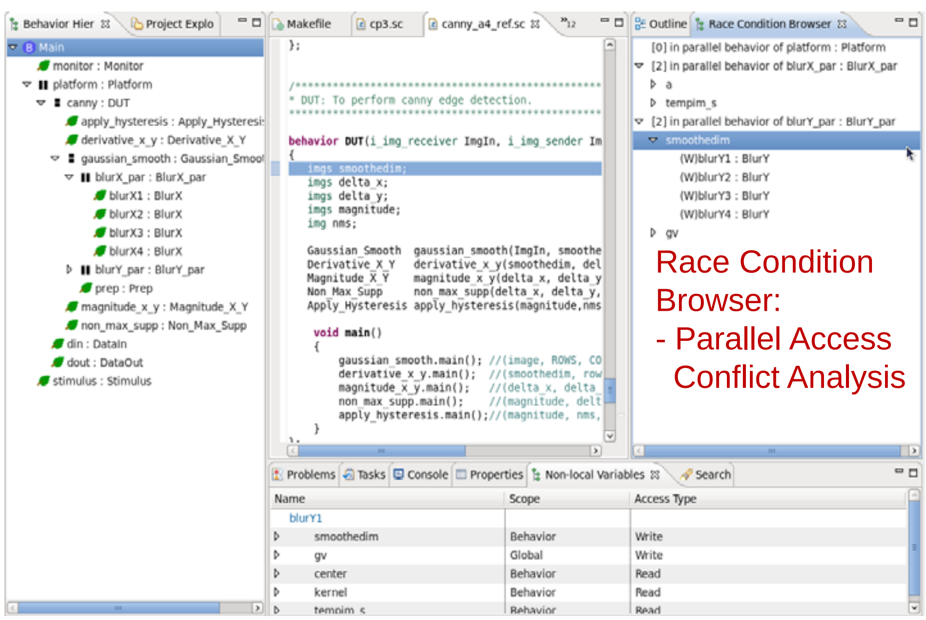


Tutorial SD1, ASPDAC '14, Singapore January 20, 2014 22

Eclipse-Based Recoding Platform



Eclipse-Based Recoding Platform



Classroom Case Study

- Case Study Experimental Setup
  - A class of 68 graduate students
  - Individually assigned recoding task
  - SpecC-extended Eclipse offered as an *optional* tool
- Assigned task:
  - Recode Canny edge detector from C reference to SpecC SLDL

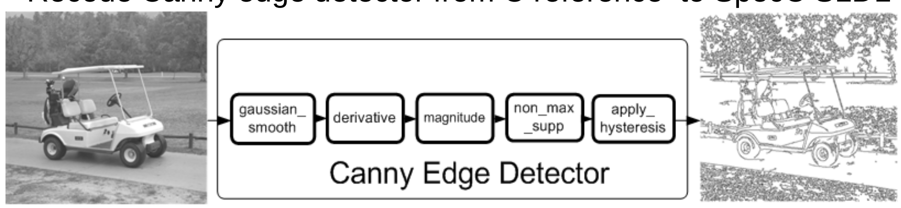


Image Input Image Output

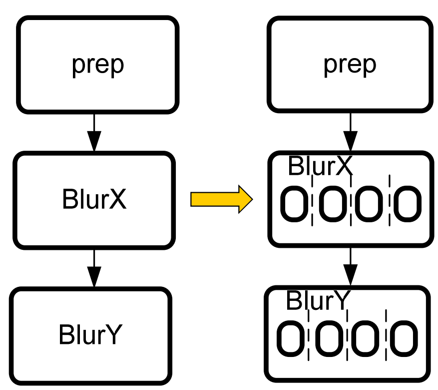
1. Analysis: *gaussian_smooth* contains 50% of the computation
2. Decision for parallelization: parallelize *gaussian_smooth*
3. Structure and variable recoding

Classroom Case Study

- Recode the *gaussian_smooth* function

```
gaussian_smooth (unsigned char *imgin, int rows,
int cols, float sigma, short int *smoothedimg)
 { int r, c, rr, cc, windowsize, center;
   float tempim[SIZE], kernel[WINSIZE], dot, sum;
   /* Create a 1-dimensional gaussian smoothing kernel */
   make_gaussian_kernel(sigma, kernel, &windowsize);
   center = windowsize / 2;
   /* Blur in the x - direction */
   for(r=0;r<rows;r++){
     for(c=0;c<cols;c++){
       dot = 0.0;
       sum = 0.0;
       for(cc=(-center);cc<=center;cc++){
         if(((c+cc) >= 0) && ((c+cc) < cols)){
           dot += (float)imgin[r*cols+(c+cc)] * kernel[center+cc];
           sum += kernel[center+cc];}}
       tempim[r*cols+c] = dot/sum;
     }}
   /* Blur in the y - direction */
   for(c=0;c<cols;c++){
     for(r=0;r<rows;r++){
       sum = 0.0;
       dot = 0.0;
       for(rr=(-center);rr<=center;rr++){
         if(((r+rr) >= 0) && ((r+rr) < rows)){
           dot += tempim[(r+rr)*cols+c] * kernel[center+rr];
           sum += kernel[center+rr]; }}
       smoothedimg[r*cols+c] =
                   (short int)(dot*FACTOR/sum + 0.5); }}
 }
```

➢Structure Recoding

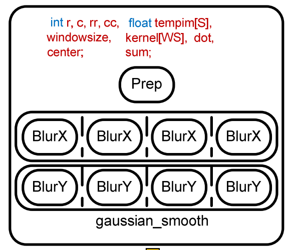


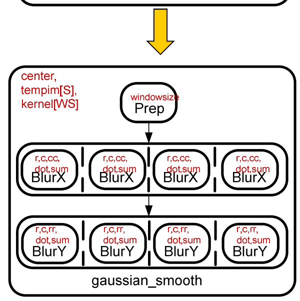
Classroom Case Study

- Recode the *gaussian_smooth* function

```
gaussian_smooth (unsigned char *imgin, int rows,
int cols, float sigma, short int *smoothedimg)
{  int r, c, rr, cc, windowsize, center;
   float tempim[SIZE], kernel[WINSIZE], dot, sum;
   /* Create a 1-dimensional gaussian smoothing kernel */
   make_gaussian_kernel(sigma, kernel, &windowsize);
   center = windowsize / 2;
   /* Blur in the x - direction */
   for(r=0;r<rows;r++){
     for(c=0;c<cols;c++){
       dot = 0.0;
       sum = 0.0;
       for(cc=(-center);cc<=center;cc++){
         if(((c+cc) >= 0) && ((c+cc) < cols)){
           dot += (float)imgin[r*cols+(c+cc)] * kernel[center+cc];
           sum += kernel[center+cc];}}
       tempim[r*cols+c] = dot/sum;
     }}
   /* Blur in the y - direction */
   for(c=0;c<cols;c++){
     for(r=0;r<rows;r++){
       sum = 0.0;
       dot = 0.0;
       for(rr=(-center);rr<=center;rr++){
         if(((r+rr) >= 0) && ((r+rr) < rows)){
           dot += tempim[(r+rr)*cols+c] * kernel[center+rr];
           sum += kernel[center+rr]; }}
       smoothedimg[r*cols+c] =
                  (short int)(dot*FACTOR/sum + 0.5); }}
}
```

➢Variable Recoding





Classroom Case Study

- Recode the *gaussian_smooth* function

```
gaussian_smooth (unsigned char *imgin, int rows,
int cols, float sigma, short int *smoothedimg)
{  int r, c, rr, cc, windowsize, center;
   float tempim[SIZE], kernel[WINSIZE]
   /* Create a 1-dimensional gaussian s
   make_gaussian_kernel(sigma, kerne
   center = windowsize / 2;
   /* Blur in the x - direction */
   for(r=0;r<rows;r++){
     for(c=0;c<cols;c++){
       dot = 0.0;
       sum = 0.0;
       for(cc=(-center);cc<=center;cc++
         if(((c+cc) >= 0) && ((c+cc) < co
           dot += (float)imgin[r*cols+(c+
           sum += kernel[center+cc];}}
       tempim[r*cols+c] = dot/sum;
     }}
   /* Blur in the y - direction */
   for(c=0;c<cols;c++){
     for(r=0;r<rows;r++){
       sum = 0.0;
       dot = 0.0;
       for(rr=(-center);rr<=center;rr++){
         if(((r+rr) >= 0) && ((r+rr) < rows
           dot += tempim[(r+rr)*cols+c] '
           sum += kernel[center+rr]; }}
       smoothedimg[r*cols+c] =
                  (short int)(dot*FACTC
}
```

behavior Prep
Reads: *sigma, imgin, windowsize*
Writes: *center*
RW: *kernel*

behavior BlurX
Reads: *cols, rows, kernel, image, center*
Writes: *tempim*
RW: *r, c ,cc, dot, sum*

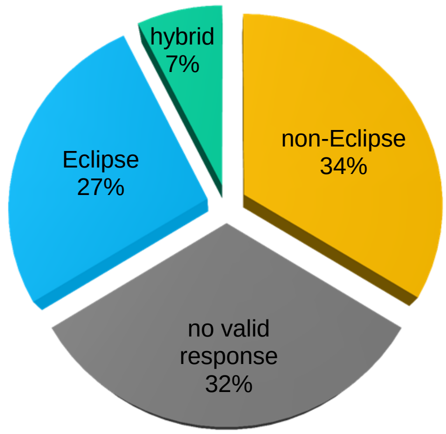
behavior BlurY
Reads: *cols, rows, kernel, image, center, tempim*
Writes: *smoothedim*
RW: *r, c ,rr, dot, sum*

➢Variable Dependency Analysis
- Designer-controlled options
  - re-locate
  - localize
  - duplicate
  - channels/ports

Classroom Case Study

- Case Study Results

Student Distribution

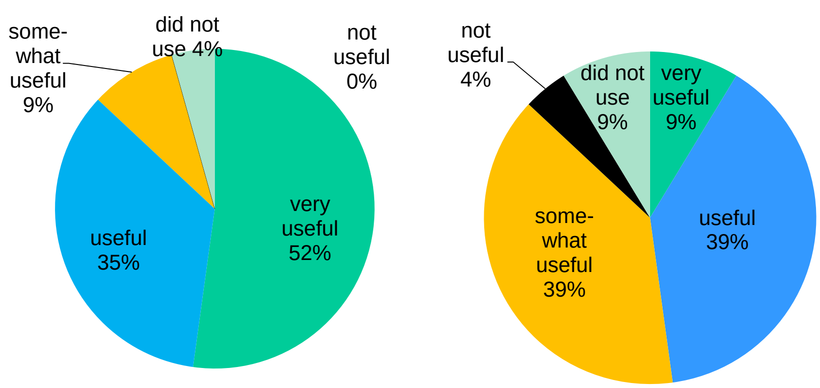


Tutorial SD1, ASPDAC '14, Singapore January 20, 2014 29

Classroom Case Study

- Case Study Results

Student Ratings

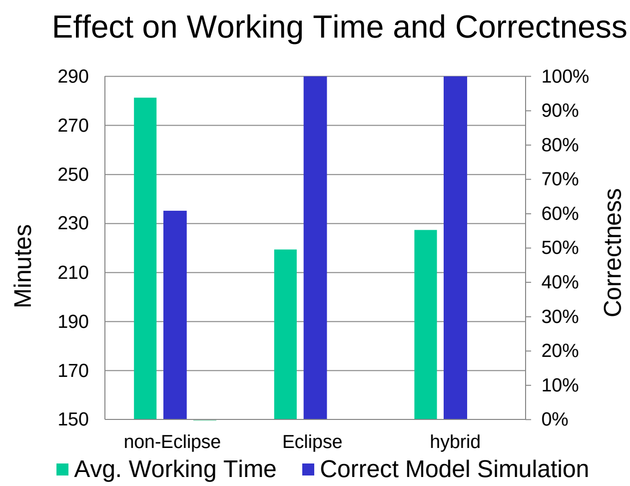


Hierarchy View Non-local Variable View

Tutorial SD1, ASPDAC '14, Singapore January 20, 2014 30

Classroom Case Study

- Case Study Results



➢Eclipse users: needed less time, yet made less mistakes!

Conclusions

- Embedded System Design
  - Start from higher level of abstraction
  - Need flexible system models in SLDL
- Motivation
  - Automation gap between C reference and SLDL system models
  - 90% of the overall design time spent on “coding” and “re-coding”
  - Need for design automation
- Problem
  - Complete automation is difficult
- Approach
  - *Computer-Aided Recoding* using Source Recoder
  - Designer-in-the-loop
- Results
  - Significant gains in productivity
  - Significant improvements in correctness
- Future work
  - SystemC!

References

- [ASPDAC'07] P. Chandraiah, J. Peng, R. Dömer, *"Creating Explicit Communication in SoC Models Using Interactive Re-Coding"*, Proceedings of the Asia and South Pacific Design Automation Conference 2007, Yokohama, Japan, January 2007.
- [IESS'07] P. Chandraiah, R. Dömer, *"An Interactive Model Re-Coder for Efficient SoC Specification"*, Proceedings of the International Embedded Systems Symposium, "Embedded System Design: Topics, Techniques and Trends" (ed. A. Rettberg, M. Zanella, R. Dömer, A. Gerstlauer, F. Rammig), Springer, Irvine, California, May 2007.
- [DAC'07] P. Chandraiah, R. Dömer, *"Designer-Controlled Generation of Parallel and Flexible Heterogeneous MPSoC Specification"*, Proceedings of the Design Automation Conference 2007, San Diego, California, June 2007.
- [ISSS+CODES'07] P. Chandraiah, R. Dömer, *"Pointer Re-coding for Creating Definitive MPSoC Models"*, Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis, Salzburg, Austria, September 2007.
- [ASPDAC'08] P. Chandraiah, R. Dömer, *"Automatic Re-coding of Reference Code into Structured and Analyzable SoC Models"*, Proceedings of the Asia and South Pacific Design Automation Conference 2008, Seoul, Korea, January 2008.
- [TCAD'08] P. Chandraiah, R. Dömer, *"Code and Data Structure Partitioning for Parallel and Flexible MPSoC Specification Using Designer-Controlled Re-Coding"*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems vol. 27, no. 6, pp. 1078-1090, June 2008.
- [DATE'09] R. Leupers, A. Vajda, M. Bekooij, S. Ha, R. Dömer, A. Nohl, *"Programming MPSoC Platforms: Road Works Ahead!"*, Proceedings of Design Automation and Test in Europe, Nice, France, April 2009.
- [ACM TECS'12] P. Chandraiah, R. Dömer, *"Computer-Aided Recoding to Create Structured and Analyzable System Models"*, ACM Transactions on Embedded Computer Systems, vol. 11S, no. 1, article 23, 27 pages, June 2012.
- [HLDVT'12] W. Chen, C. Chang, X. Han, R. Dömer, *"Eliminating Race Conditions in System-Level Models by using Parallel Simulation Infrastructure"*, Proceedings of the International High Level Design Validation and Test Workshop 2012, Huntington Beach, California, November 2012.
- [HLDVT'13] X. Han, W. Chen, R. Dömer, *"Designer-in-the-Loop Recoding of ESL Models using Static Parallel Access Conflict Analysis"*, Proceedings of the 16th International Workshop on Software and Compilers for Embedded Systems, St. Goar, Germany, June 2013.