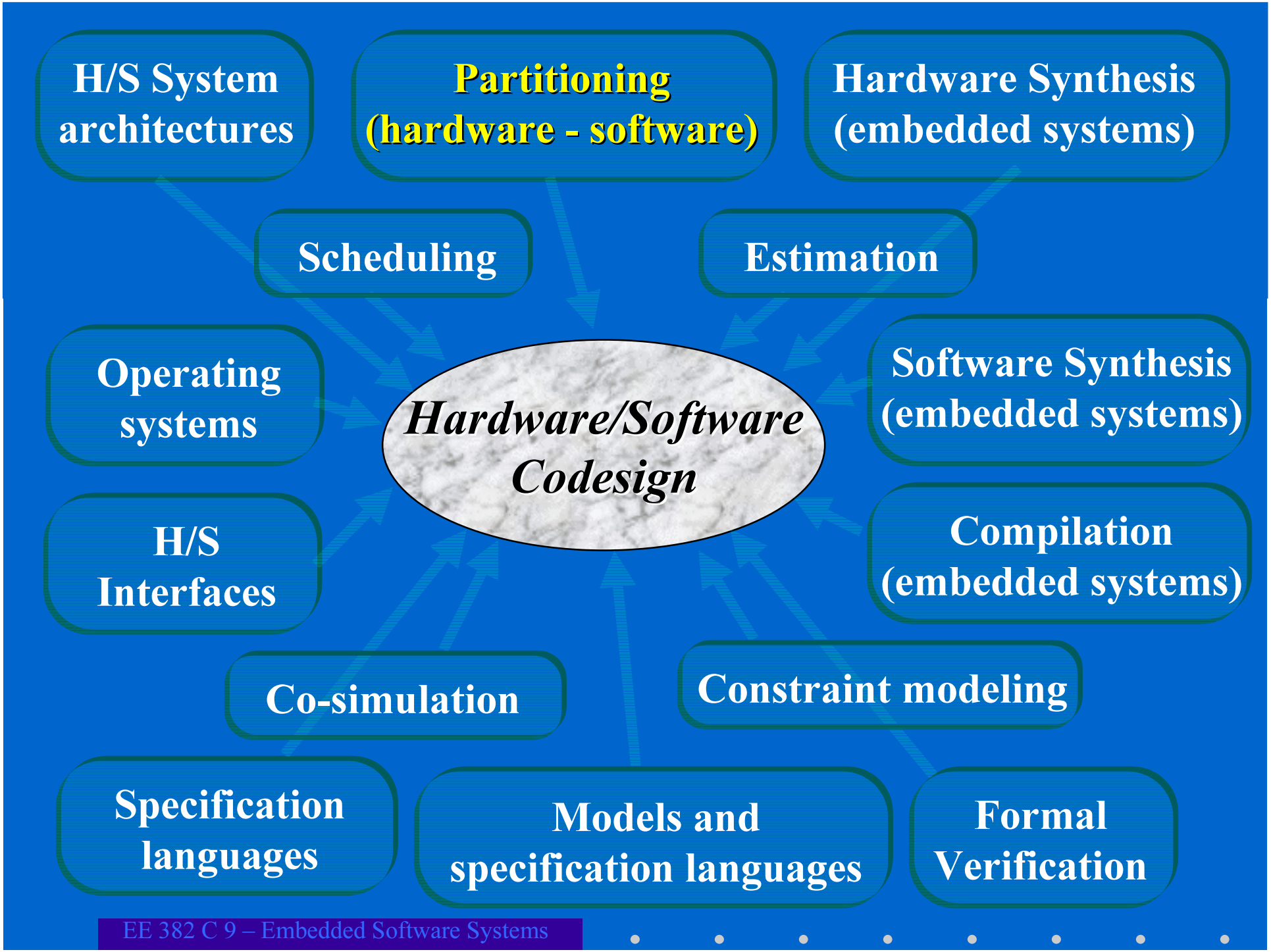
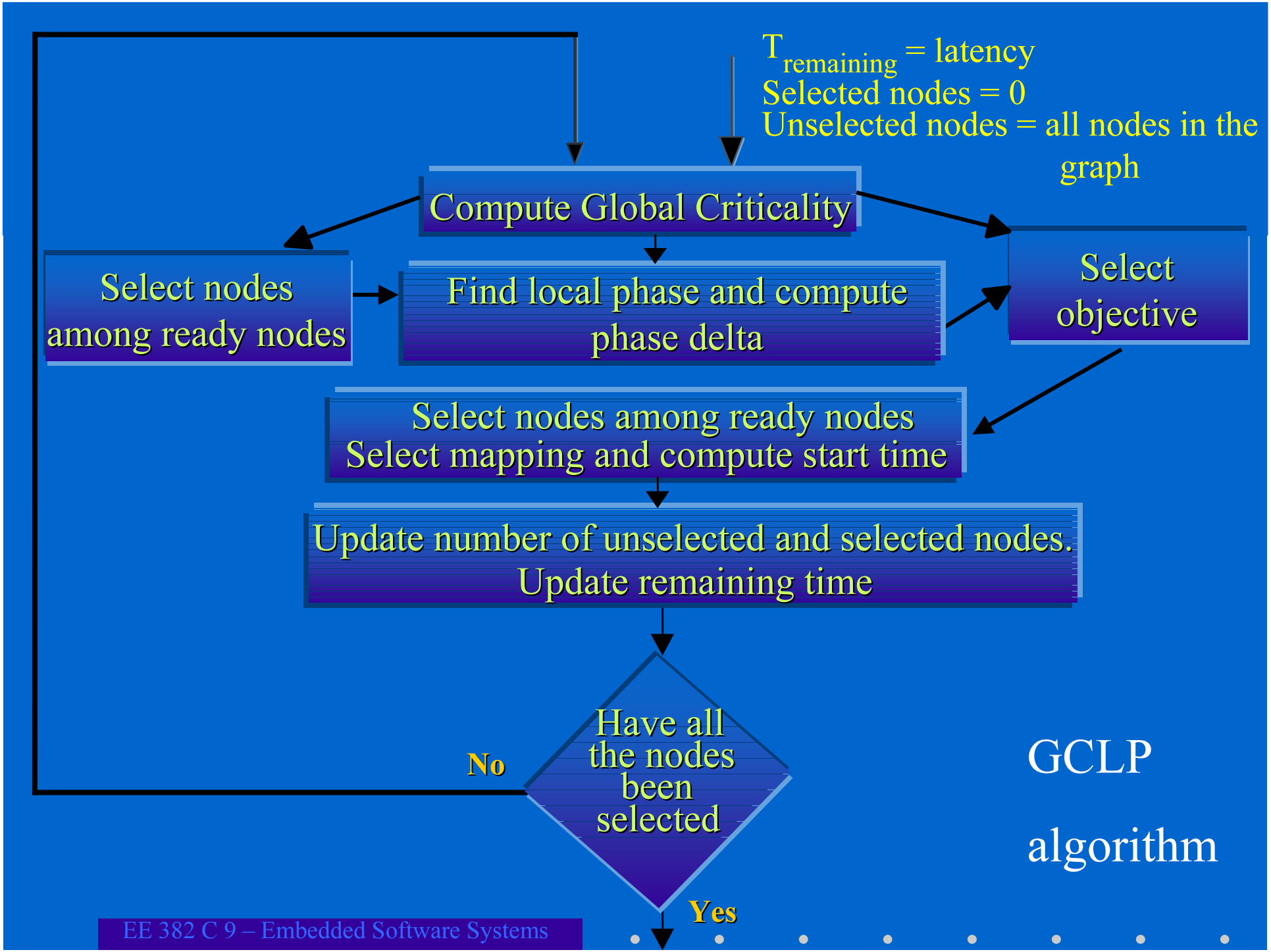
Hardware/Software Partitioning of Synchronous Dataflow Graphs in Code Generation domain of Ptolemy

Heather Hanson
Gayathri Manikutty


H/S System architectures
Partitioning (hardware - software)
Hardware Synthesis (embedded systems)
Scheduling
Estimation
Operating systems
Hardware/Software Codesign
Software Synthesis (embedded systems)
H/S Interfaces
Compilation (embedded systems)
Co-simulation
Constraint modeling
Specification languages
Models and specification languages
Formal Verification



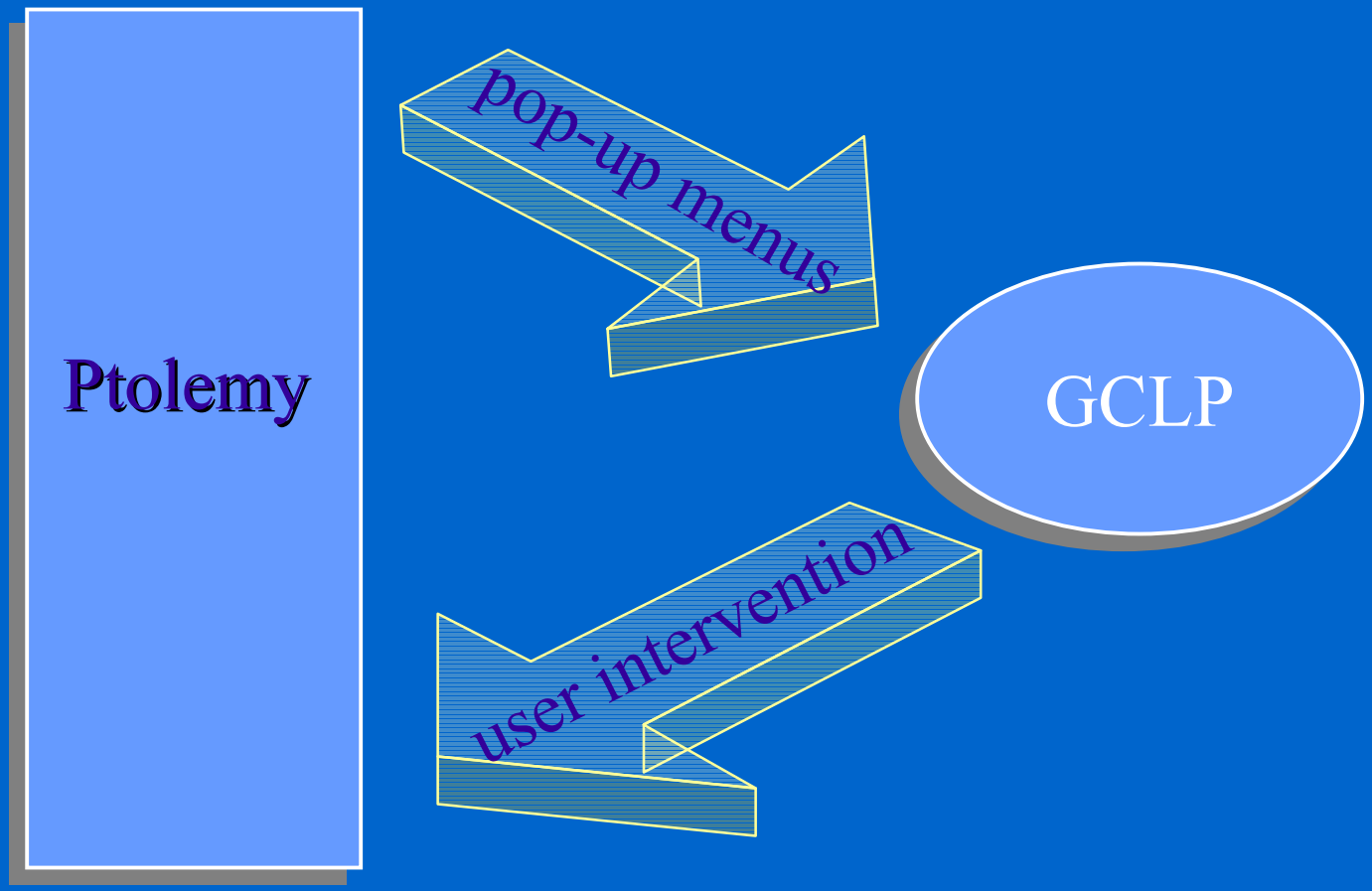
GCLP algorithm

Project Goals

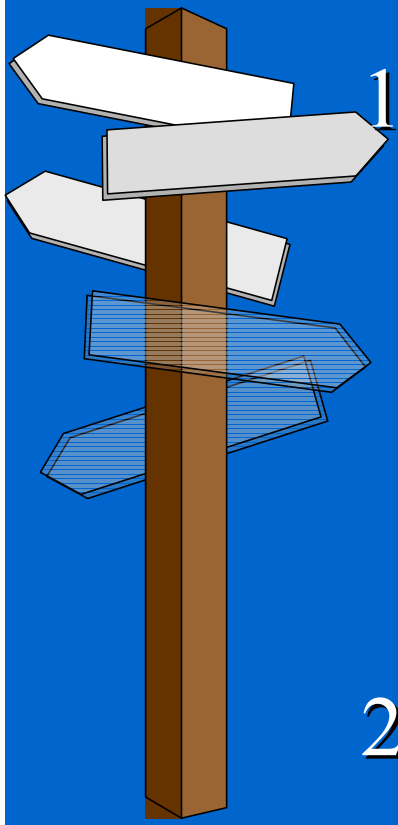
1. Integrate existing GCLP partitioning algorithm into Ptolemy's CGC domain.

2. Test partitioning with existing Ptolemy demos.

Integration



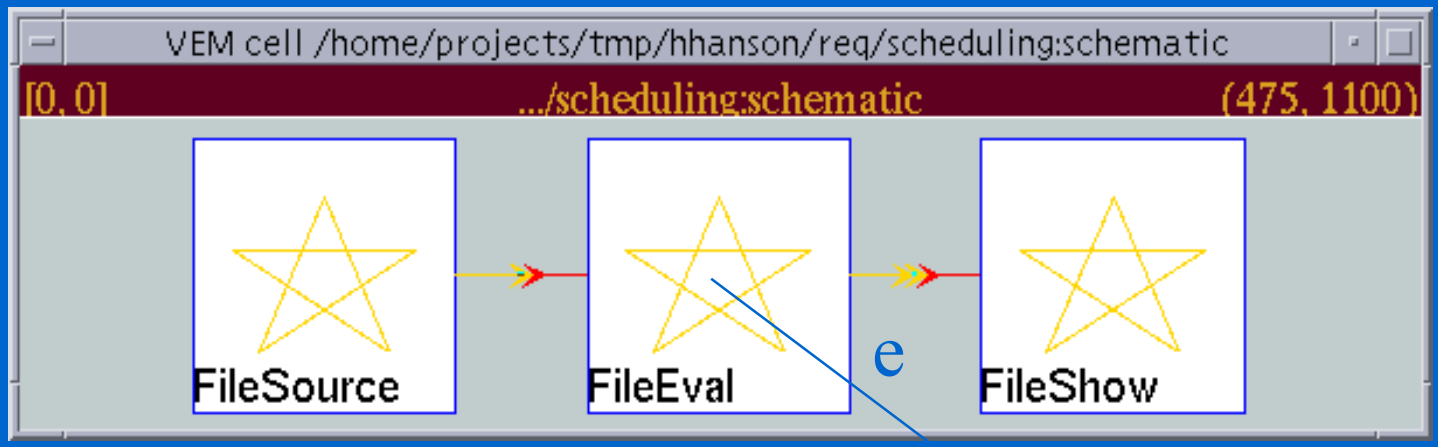
Project Plan



1. Create design flow universe to guide the user through partitioning tasks:
   - translate CGC universe into node list (for input to GCLP code)
   - partition design with GCLP algorithm.
2. Compare partitioning results with pure CGC, pure VHDL, and mixed-domain simulations.

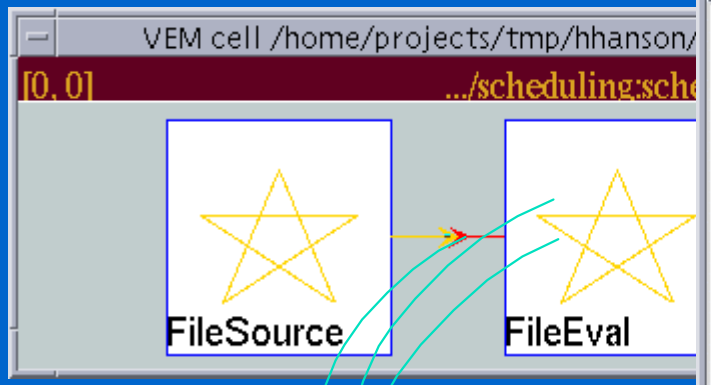
Design Methodology Universe

- Translate CGC universe into text-based ptcl format



specifies which design to use and where to put the ptcl file

Preparing input files for GCLP



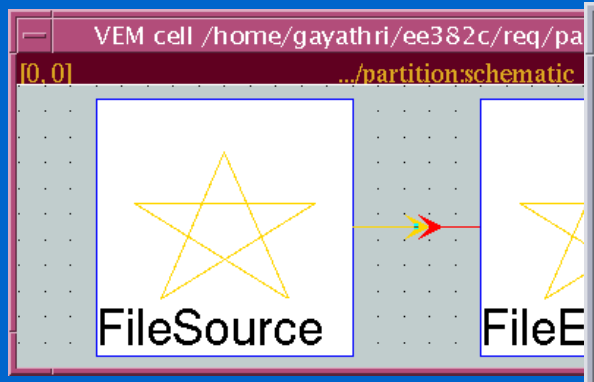
/home/gayathri/ee382c/req/temp.pt

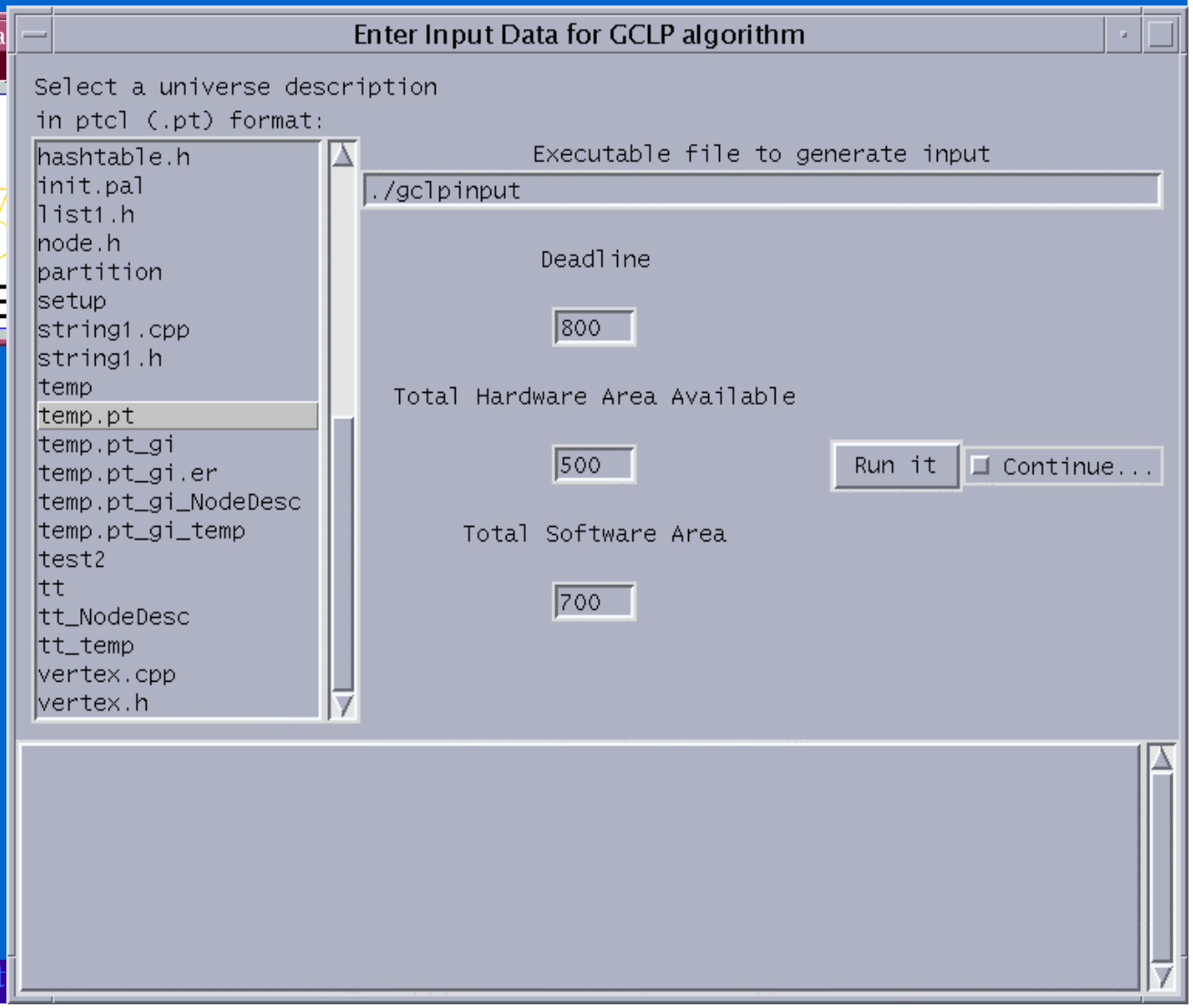
File Edit Window Format Special Help

```
GALAXY: butterfly
Descriptor: An interpreted galaxy
Contained blocks: Star Gain1
Star Gain2
Star Gain3
Star Add1
Star Mpy.input=21
Star Mpy.input=22
Star Mpy.input=23
Star Ramp1
Star PolarToRect1
Star Sin1
Star Cos1
Star Cos2
Star Exp1
Star XYgraph1
Star auto-fork-node1
Star auto-fork-node2
Star auto-fork-node3
--------------------------------------------------------------------------------

Star: butterfly.Gain1
Descriptor: Amplifier: output is input times "gain" (default 1.0).
Ports:
        Input input
        Connected to port: butterfly.auto-fork-node3.output#1(numTokens = 1)

        Output output
        Connected to port: butterfly.Cos1.input(initValues = )(numTokens = 1)

States:
        procId type: INT A_DYNAMIC
        gain type: FLOAT A_DYNAMIC
--------------------------------------------------------------------------------

Star: butterfly.Gain2
Descriptor: Amplifier: output is input times "gain" (default 1.0).
Ports:
        Input input
        Connected to port: butterfly.auto-fork-node3.output#2(numTokens = 1)
```

Close

Menus: preparing input files



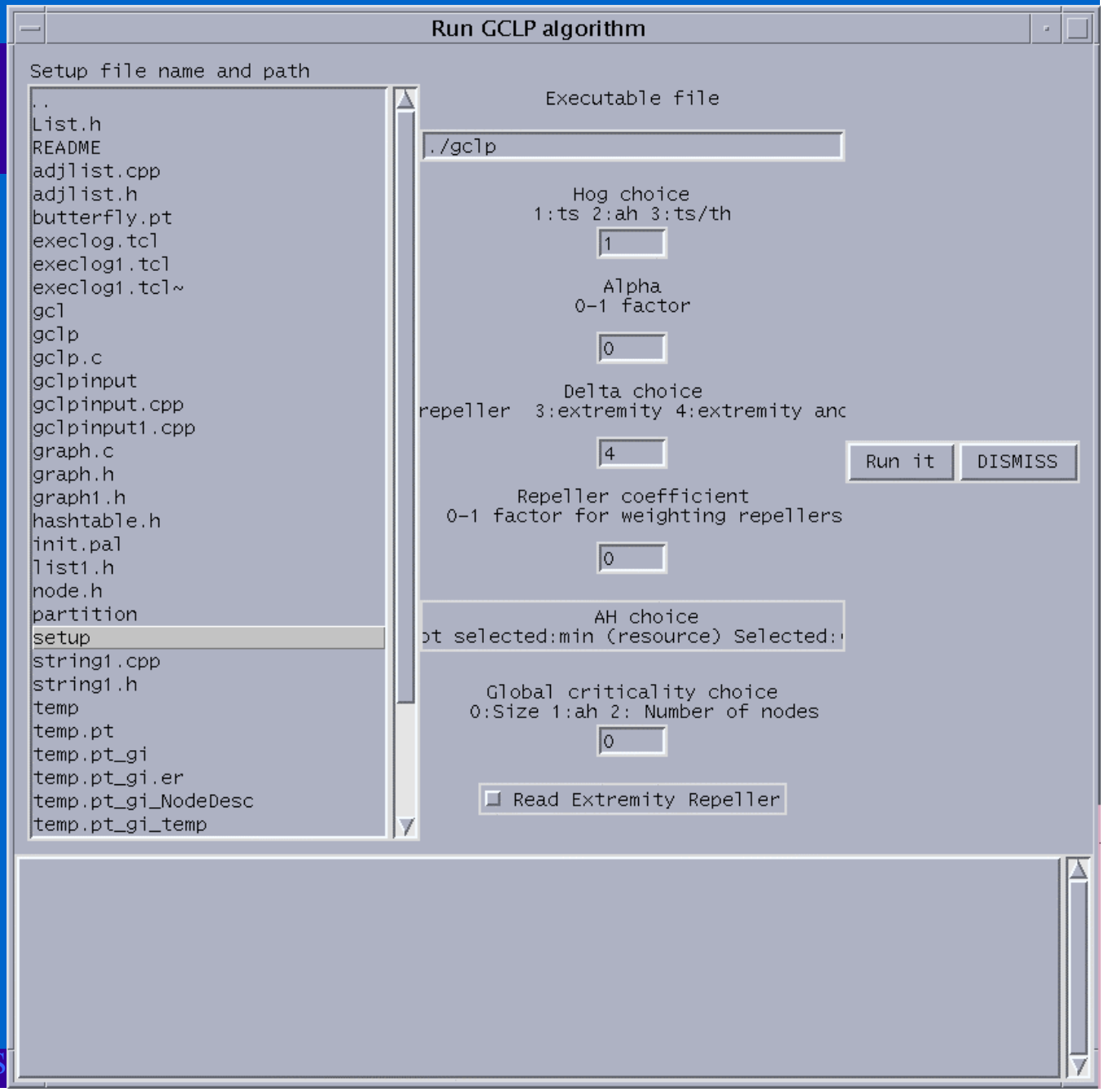


Add design constraints, size, and execution time estimates

Choose parameters for GCLP algorithm

then

partition the design.



Construct partitioned design

The designer uses GCLP's partitioning recommendation to construct a VHDL wormhole within the CGC universe.

This step could be automated, but currently is the responsibility of the designer.

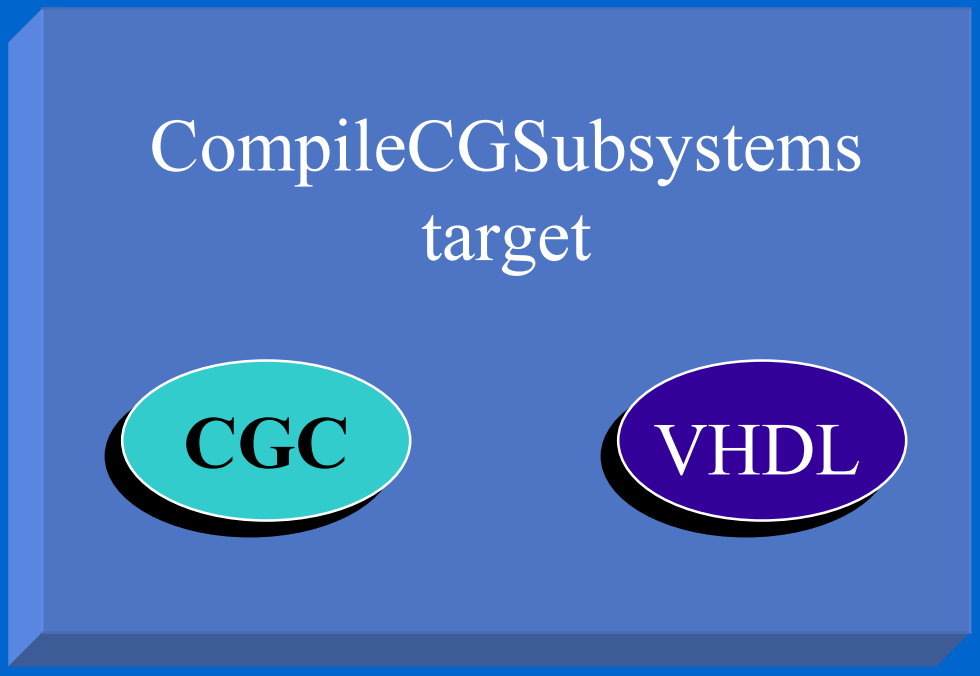
Compile and Simulate

Code generation:

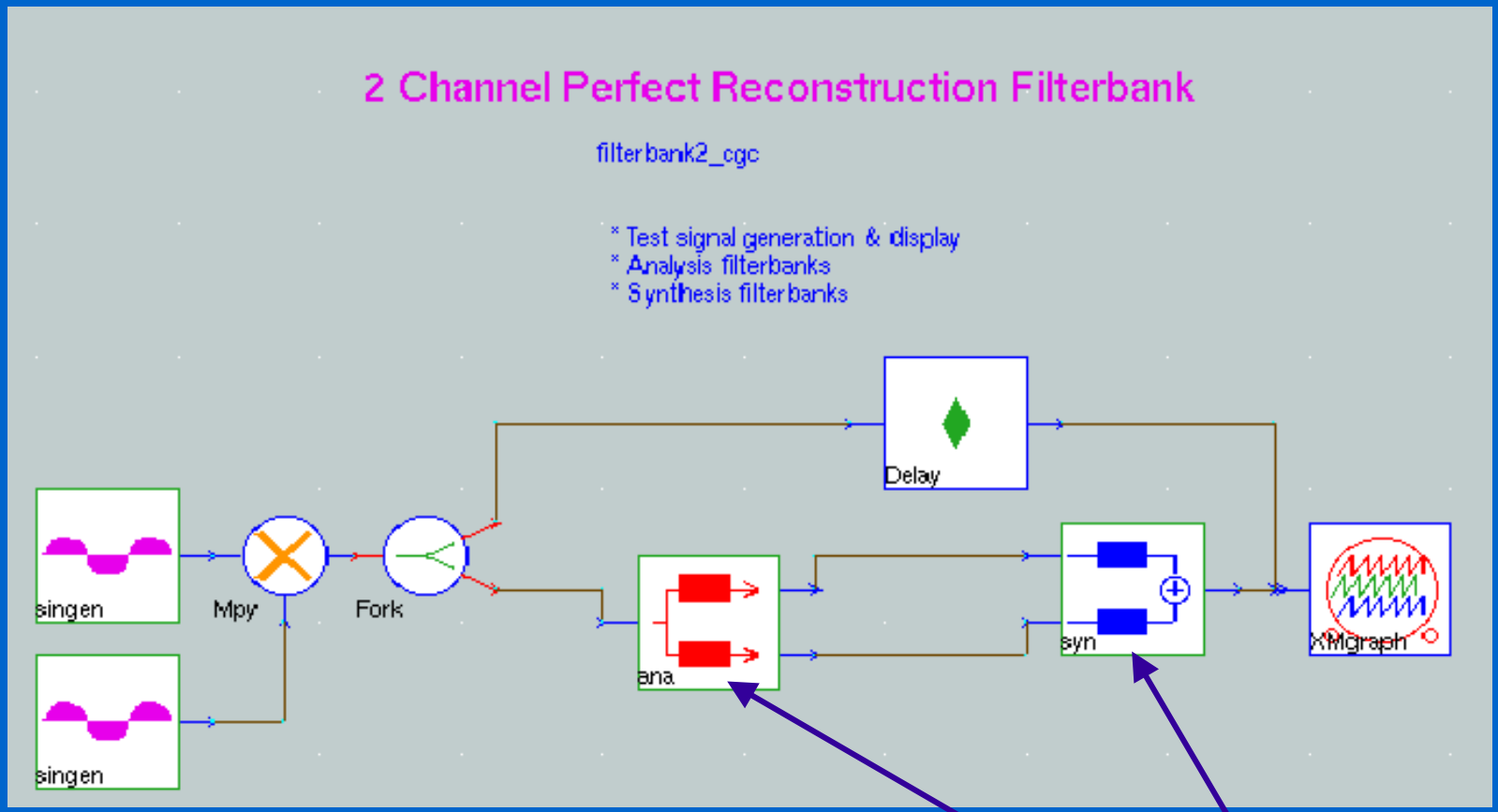
target specifies separate files for C, VHDL code.

Cosimulation:

C program calls VHDL simulator



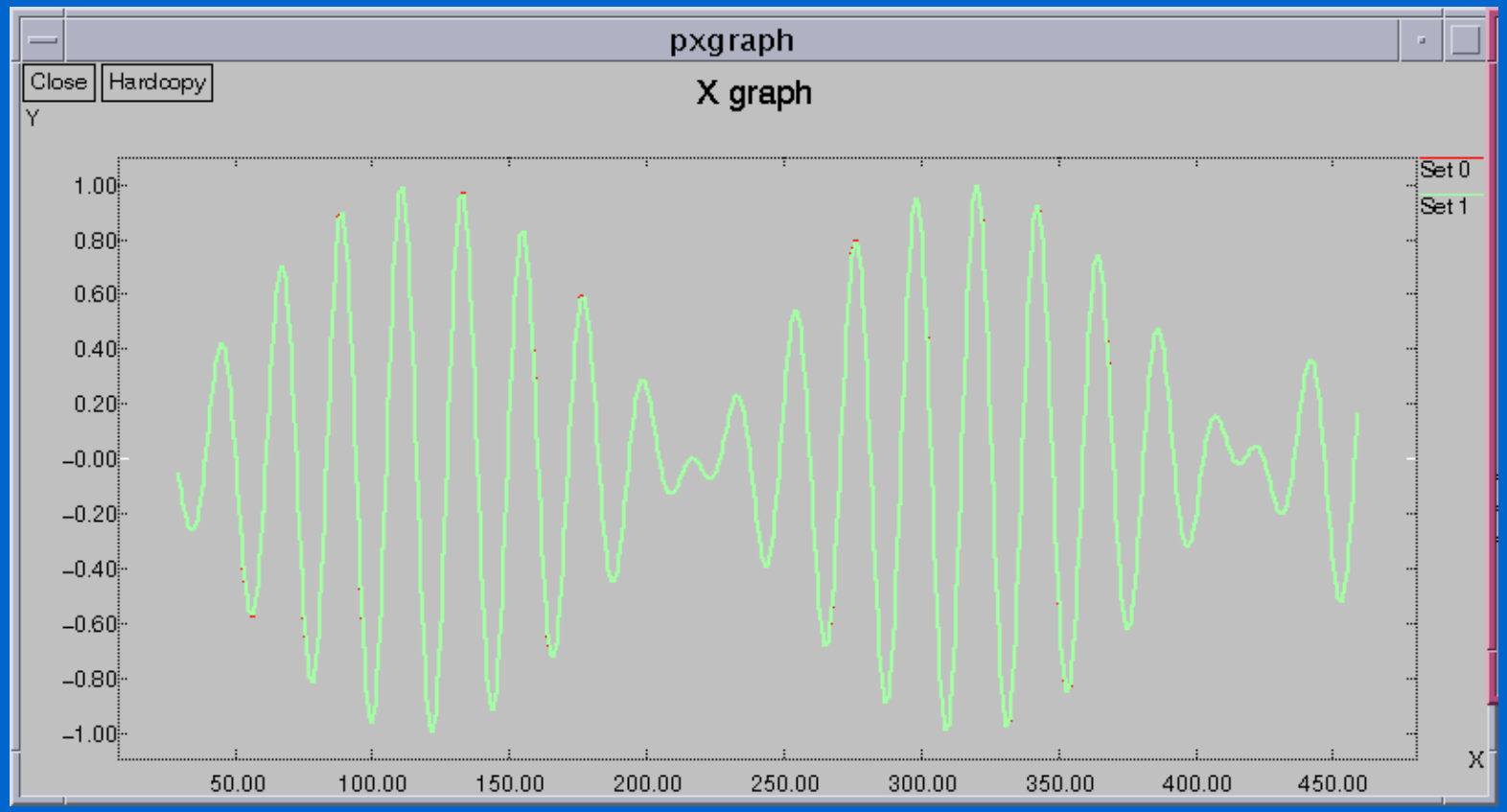
Test Case: filterbank



Tests replace analysis, synthesis blocks with VHDL components

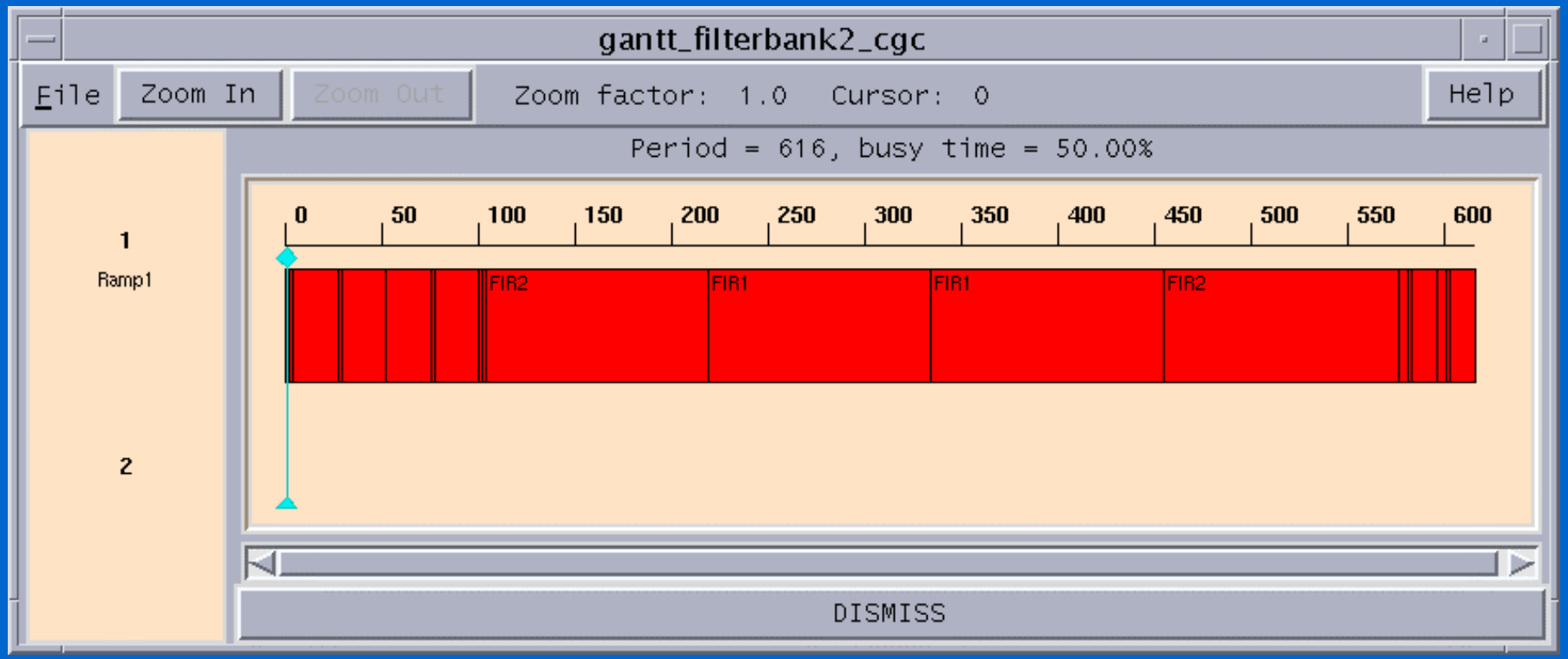
Filterbank Output

(all tests have same output)



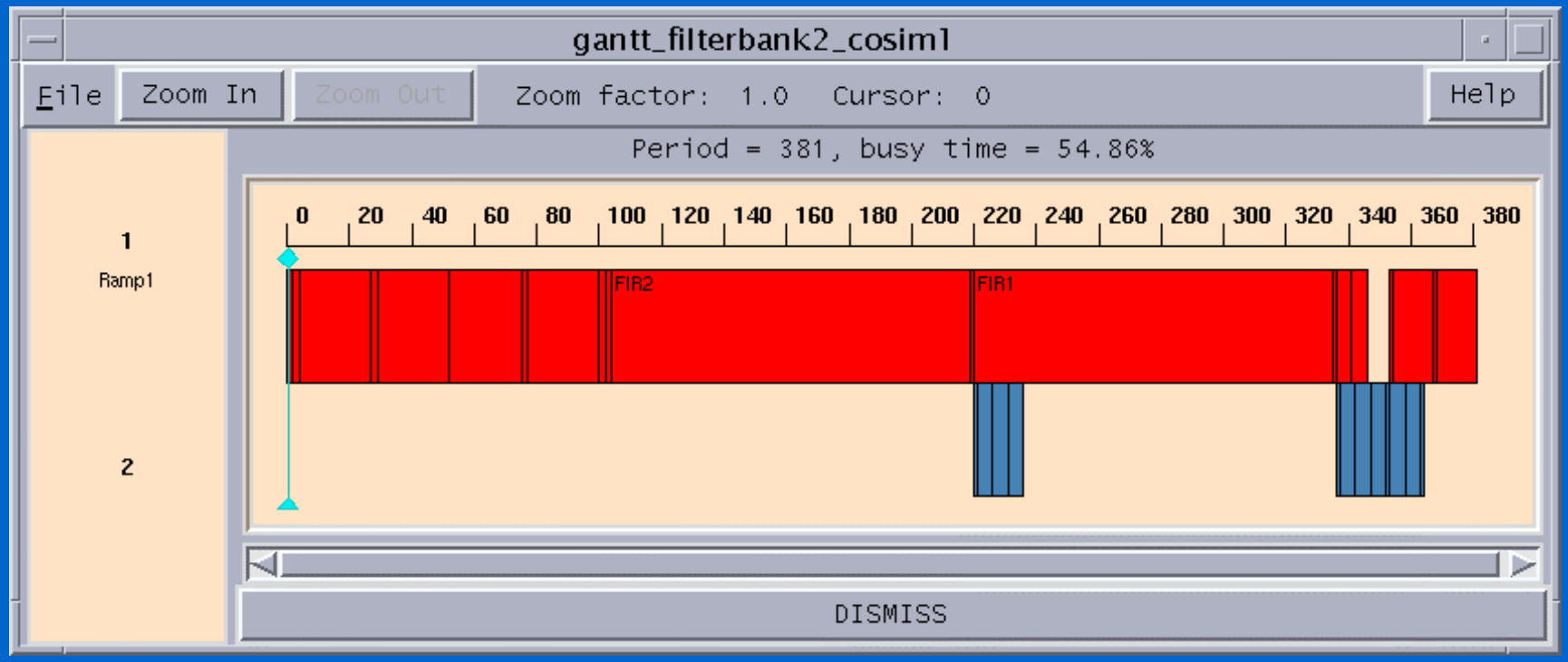
Gantt Charts: scheduling for Software and Hardware

Original universe in C (software) only



Gantt Charts: scheduling for Software and Hardware

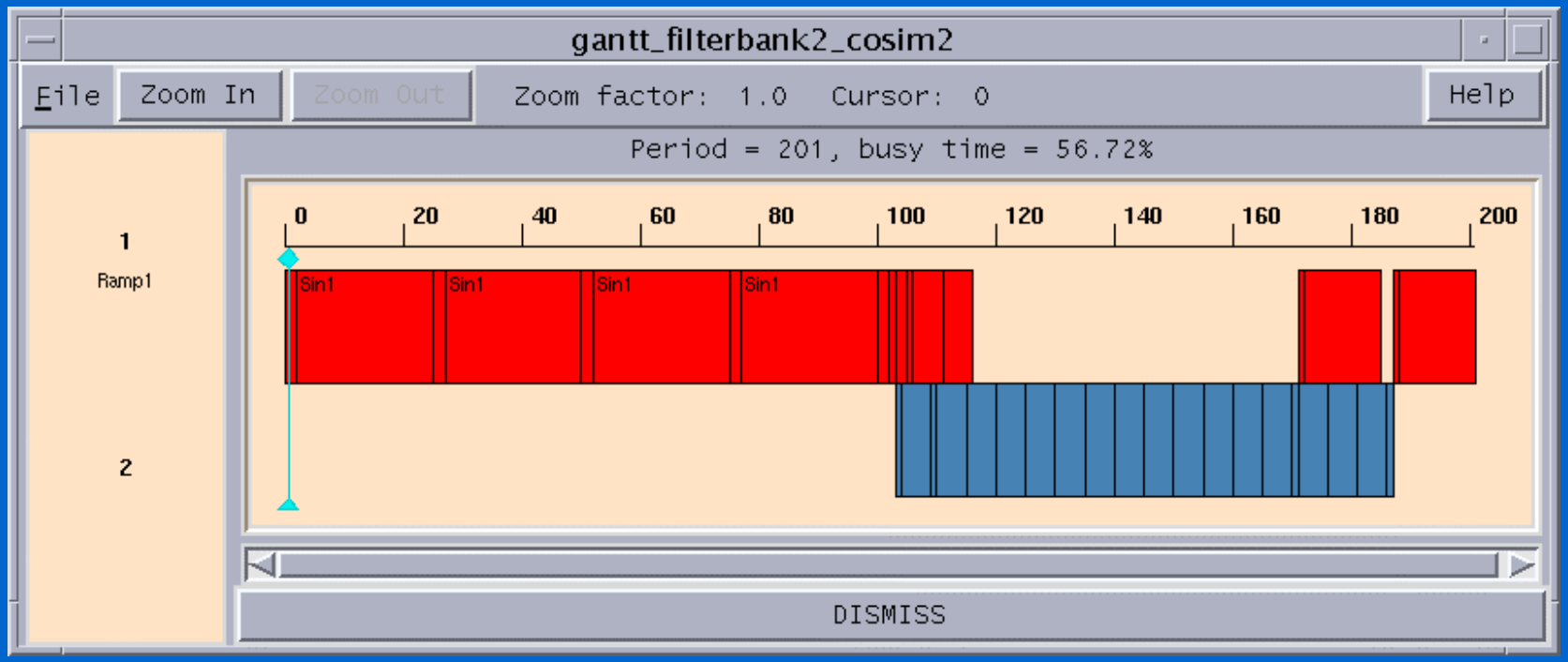
Test 1: synthesis block in VHDL



Gantt Charts: scheduling for Software and Hardware

Test 2: analysis and synthesis blocks in VHDL

(GCLP recommendation with hardware-biased inputs)



Conclusion

Integrating the GCLP code into Ptolemy helps a user partition a design into hardware and software:

- "front end" menus are automated
- "back end" is not--still need user intervention.

Future work could be continued in ACS domain.