Source level debugging Handel-C

Debug the FPGA, Herman Roebbers
07-Sep-2008
Introduction

- TASS and Handel-C history
- The problem
- Solution
- Approaches
- Screen shots
- Conclusions
- Future work
TASS and Handel-C, a history

- TASS is a software house in Eindhoven, NL
- Ex-Philips
- Embedded Software
- Approx. 200 people
- Using Handel-C for student projects since 2001
- Several Commercial projects with Handel-C
The problem

- Sometimes Handel-C simulation and reality do not agree
- Handel-C simulation of HW components can take ages
- If it doesn’t work, where/what is the problem
- A quick peek would make things clear very quickly
Solution

- We want a graphical Handel-C source level debugger
- We know there are restrictions, but we may be able to live with them.
- Let’s put some students to work!
Approaches

1) Insert code at the EDIF level
   – Difficult to know names of variables at EDIF level
   – Variables or names are optimized away
   + No changes to Handel-C source
   + Not so intrusive
Approaches

2) Insert code at the Handel-C level
   – Need a (primitive) Handel-C parser
   – No replicated par{ } support yet
   – Need to hide extra Handel-C code when debugging
   – Use more FPGA resource
   + Easier to do
   + Can relate to names/arrays
Requirements

- Source level debugger
- Set / remove breakpoints
- Detect which breakpoints are hit
- Inspect/change program variables
- Do single stepping
- Make debugger communication independent of communication mechanism (RS232/JTAG)
Results

Source level debugger based on approach 2
- Set / remove breakpoints
- Detect which breakpoints are hit
- Inspect/change program variables when program is in breakpoint
- Do single stepping by setting/removing bpts
- RS-232 communication with debugger
Results
Results
Results

- Inspect / change variables
- Variables presented in declared type (signed/unsigned, char)
- Only after associated breakpoint is hit
- Only change ticked variables
Limitations

- Need a separate program to
  - Show Handel-C source without added instrumentation
  - Indicate variables to be inspected
  - Set / remove / inspect breakpoints
- Only RS-232 comms with debugger implemented
- No support for
  - chan, ram, rom, signal, WOM, struct, mpram
Future work

- Use Handel-C parser instead of homebrew
- Integrate with Handel-C DK GUI
- Add JTAG /USB communication mechanism
- Add conditional / data breakpoints
- Add more complex triggering
- Support for replicated par’s