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

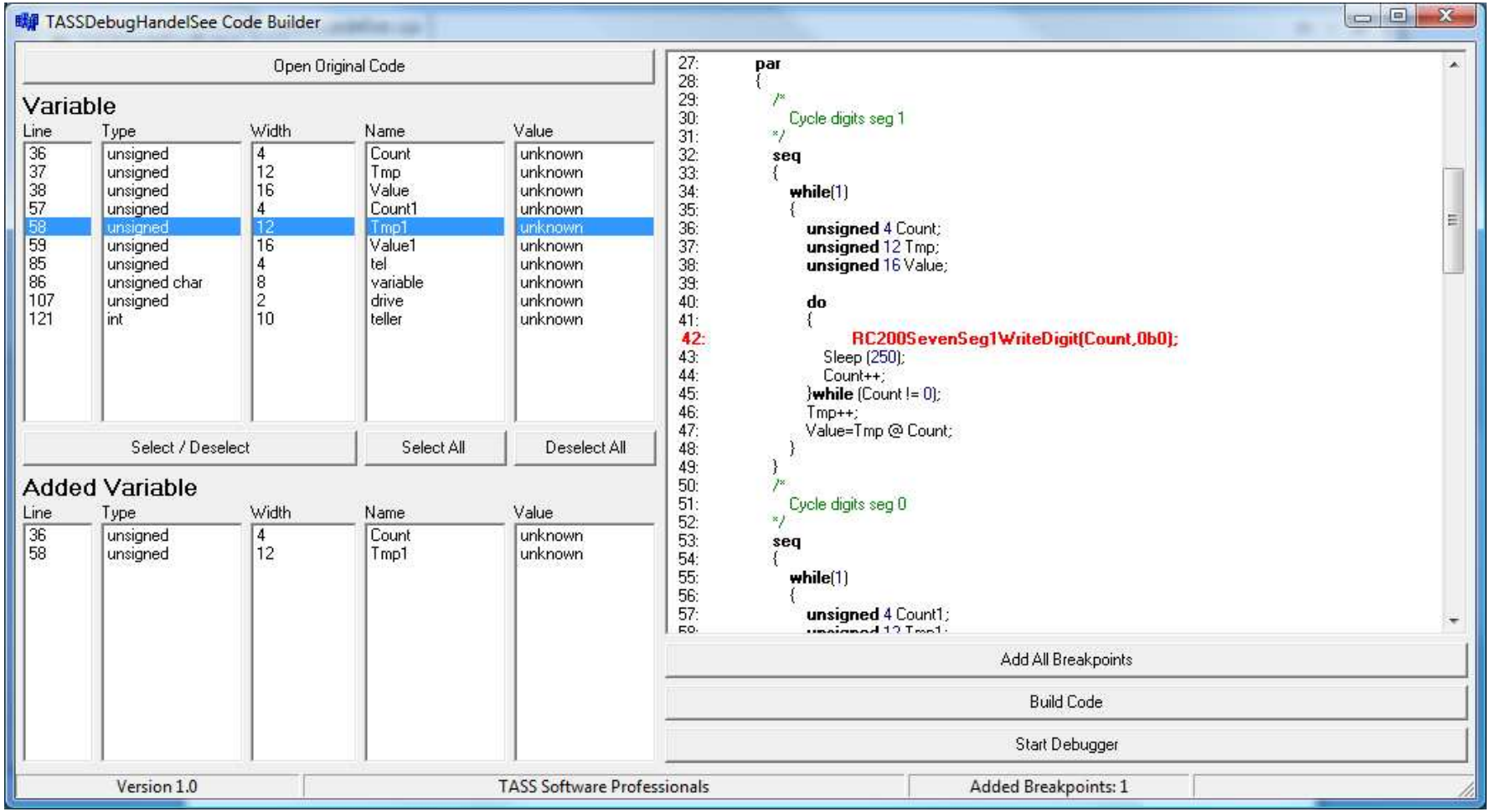
- Source level debugger
- Set / remove breakpoints
- Detect which breakpoints are hit
- Inspect/change program variables
- Do single stepping
- Make debugger communication independent of communication mechanism(RS232/JTAG)

Results

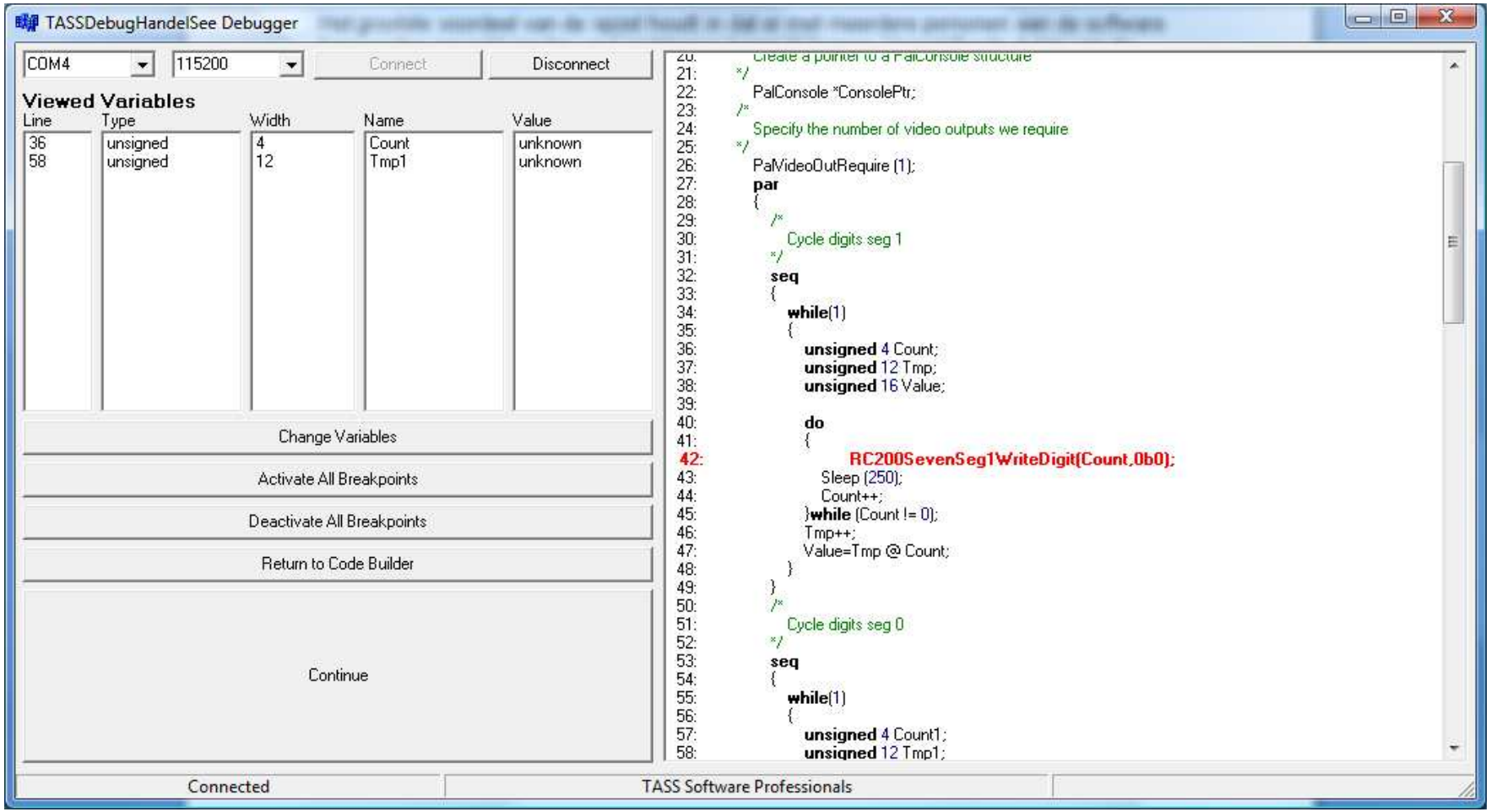
Source level debugger based on approach 2

- 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

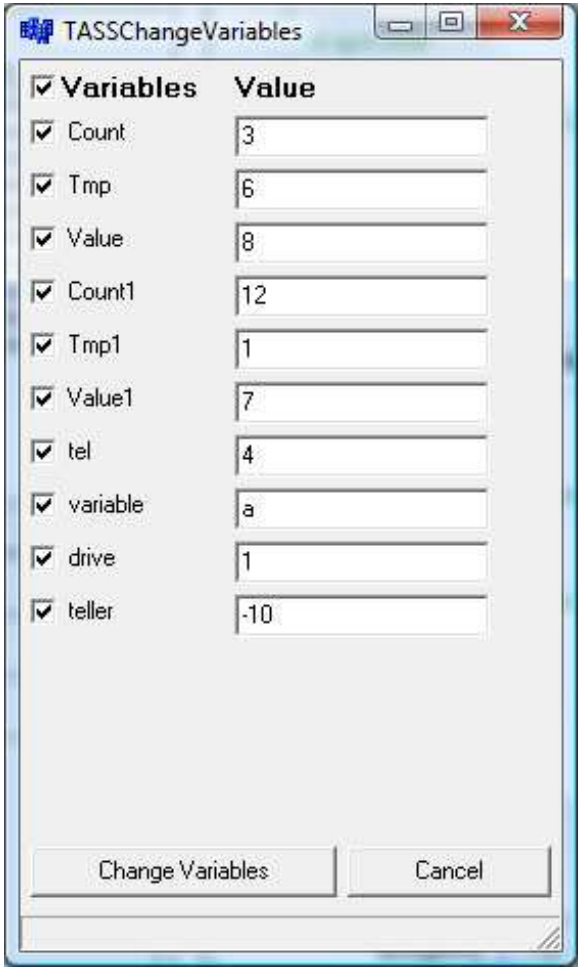

TASSDebugHandelSee Code Builder
Open Original Code
Variable
Line Type Width Name Value
36 unsigned 4 Count unknown
37 unsigned 12 Tmp unknown
38 unsigned 16 Value unknown
57 unsigned 4 Count1 unknown
58 unsigned 12 Tmp1 unknown
59 unsigned 16 Value1 unknown
85 unsigned 4 tel unknown
86 unsigned char 8 variable unknown
107 unsigned 2 drive unknown
121 int 10 teller unknown
Select / Deselect
Select All
Deselect All
Added Variable
Line Type Width Name Value
36 unsigned 4 Count unknown
58 unsigned 12 Tmp1 unknown
27: par
28: {
29: /*
30: Cycle digits seg 1
31: */
32: seq
33: {
34: while(1)
35: {
36: unsigned 4 Count;
37: unsigned 12 Tmp;
38: unsigned 16 Value;
39:
40: do
41: {
42: RC200SevenSeg1WriteDigit(Count,0b0);
43: Sleep (250);
44: Count++;
45: }while (Count != 0);
46: Tmp++;
47: Value=Tmp @ Count;
48: }
49: }
50: /*
51: Cycle digits seg 0
52: */
53: seq
54: {
55: while(1)
56: {
57: unsigned 4 Count1;
Add All Breakpoints
Build Code
Start Debugger
Version 1.0
TASS Software Professionals
Added Breakpoints: 1

Results


TASSDebugHandelSee Debugger
COM4
115200
Connect
Disconnect
Viewed Variables
Line
Type
Width
Name
Value
36
58
unsigned
unsigned
4
12
Count
Tmp1
unknown
unknown
Change Variables
Activate All Breakpoints
Deactivate All Breakpoints
Return to Code Builder
Continue
Create a pointer to a PalConsole structure
*/
PalConsole *ConsolePtr;
/*
Specify the number of video outputs we require
*/
PalVideoOutRequire (1);
par
{
/*
Cycle digits seg 1
*/
seq
{
while(1)
{
unsigned 4 Count;
unsigned 12 Tmp;
unsigned 16 Value;
do
{
RC200SevenSeg1WriteDigit(Count,0b0);
Sleep (250);
Count++;
}while (Count != 0);
Tmp++;
Value=Tmp @ Count;
}
}
/*
Cycle digits seg 0
*/
seq
{
while(1)
{
unsigned 4 Count1;
unsigned 12 Tmp1;
Connected
TASS Software Professionals

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


TASS
software professionals