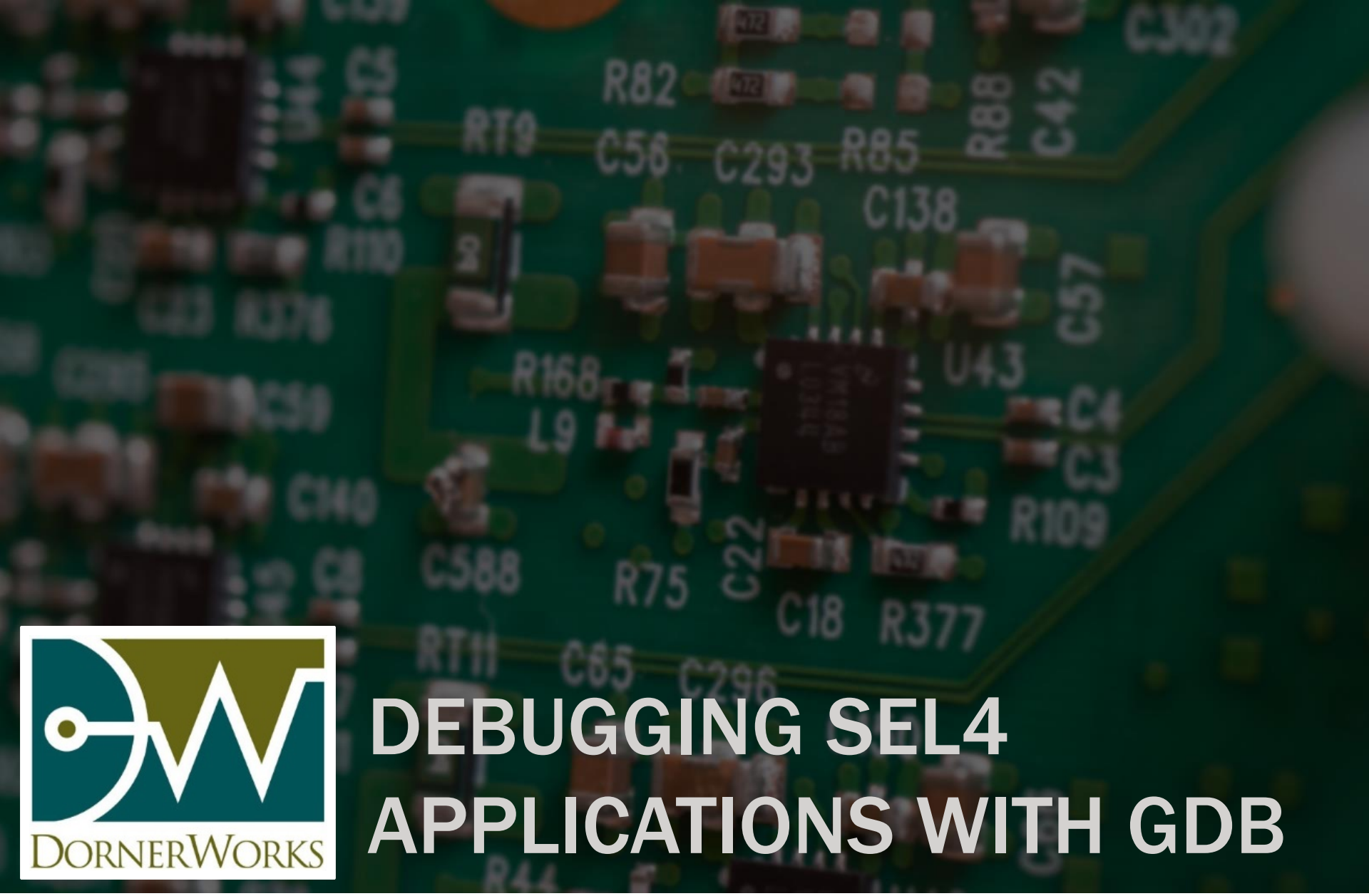


DEBUGGING SEL4 APPLICATIONS WITH GDB

This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions, and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. Approved for Public Release, Distribution Unlimited

Agenda

- Debugging Need for Embedded Systems
- GDB Overview
- 3 Implementations
  - GDB-stub ARM – DW
  - GDB-server x86 – Data61
  - GDB-server ARM – DW
- Areas for Improvement
- Questions

About DornerWorks

- Embedded Systems Engineering located in Grand Rapids, MI
- DornerWorks provides expert technology engineering to help product developers create standout products so that they are free to focus on what they do best.
- We specialize in platforms
  - Hardware development
  - Software and firmware designs
  - FPGA logic designs
- Experience with seL4
  - DARPA SBIR Project: “An seL4 – Enabled Safe & Secure Soldier Helmet Display”

Why Debuggers?

- No matter the length and readability of your code, there is **always** the possibility for bugs
- Print debugging is inflexible
  - UART access may be limited when porting to a new platform
  - Doesn't work well with timing-sensitive functions or interrupt handlers
- Debug functions allow for easier error identification

About GDB

- GNU Project Debugger, Developed in 1986 by Richard Stallman
- Most widely used embedded debugger
- Allows the user to see what happens ‘inside’ the program while it is executing
- Available Functionality:
  - Breakpoints
  - Step-Through
  - Variables
  - Stack
  - Many, Many others…
- How can this be implemented in seL4?

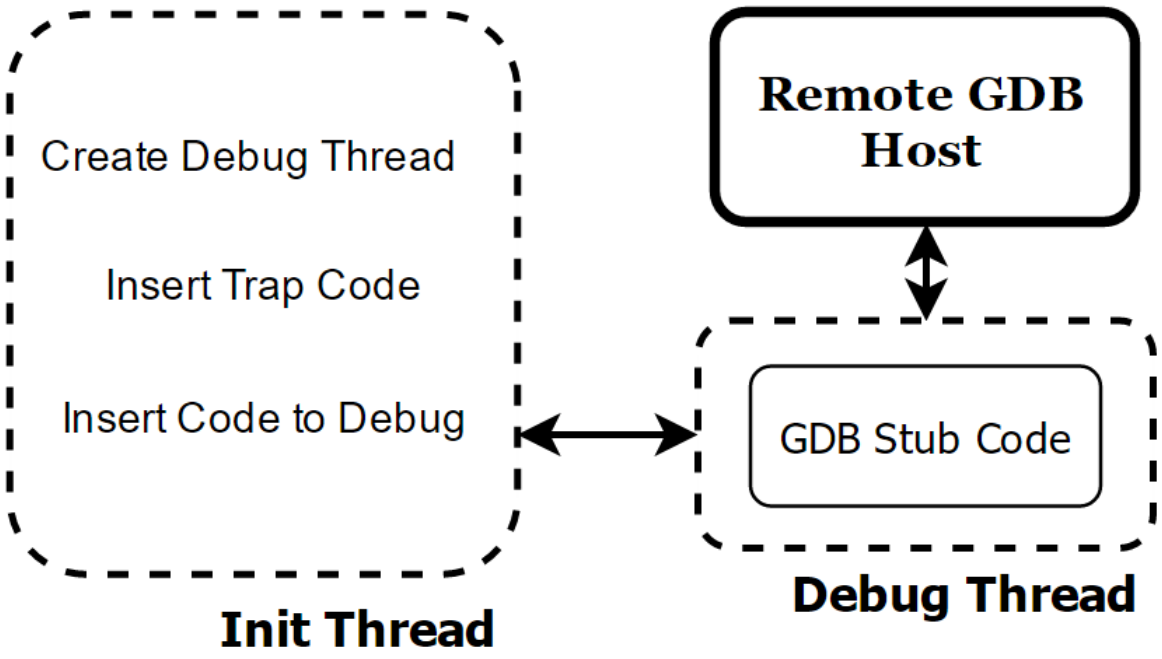
Overview of seL4 GDB Solutions

- GDB Stub Application
  - gdbstub: Code must be linked directly with the application to debug with GDB
  - For programs written with seL4 System Calls
    - Not for CAmkES applications
- GDB Server Applications
  - gdbserver: control program which allows you to connect your program with gdb without linking in the usual debugging stub.
  - CAmkES application for x86 (Data61) and ARM (DW)

GDB Stub

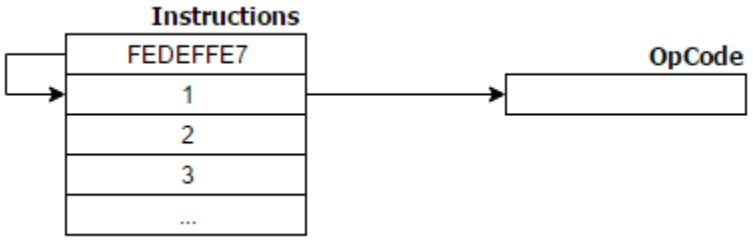
- Developed by DornerWorks
- Only tested on Zynq7000 Platform
  - Other ARM Platforms should work
- https://github.com/dornerworks/gdbstub-app
- https://github.com/dornerworks/gdbstub-app-manifest

GDB Stub


Create Debug Thread
Insert Trap Code
Insert Code to Debug
Init Thread
Remote GDB Host
GDB Stub Code
Debug Thread

GDB Stub

- GDB Stub communicates with the GDB application over UART0
  - Normal Serial Communication occurs over UART1
- Wait for user commands
  - Software Breakpoints
  - Read/Write bytes
  - Step
  - Continue



- Step/continue resumes init thread with updated PC

GDB Stub

- Strengths:
  - Can debug seL4 code!
- Weaknesses
  - Hard Coded Stub
  - No CAmkES support
  - Uses init thread to debug code

x86 GDB for CAmkES

- Developed by Data61
- https://github.com/smaccm/camkes_debug_manifest/blob/gtt_deliverable/simple.xml
- https://github.com/smaccm/camkes-tool/blob/gtt_deliverable_debug_no_rt/debug/README.md

x86 GDB for CAmkES

- Debug tool performs does the following:
  - Parses the top level assembly file & outputs new assembly file
  - Creates a delegate and a fault endpoint from template files
  - Outputs a .gdbinit

x86 GDB for CAmkES

- Available commands:
  - Read general registers
  - Memory read/write
  - Set SW breakpoint
  - Set HW breakpoint
  - Step

x86 GDB for CAmkES

- Strengths:
  - GDB Server
  - HW/SW Breakpoints
  - CAmkES compatible
- Weaknesses
  - No ARM support

ARM GDB for CAmkES

- Based off the Data61 GDB Server
- Developed by DornerWorks
- Modified in the following ways:
  - ARM Support
  - Additional GDB functionality

ARM GDB for CAmkES

- Debug tool slightly modified
  - Searches for the source file of the debug component, adds a `breakpoint()` function call
  - Creates a backup of the source code
- Creates a new target file (.camkes.dbg)
- Creates a .gdbinit file

ARM GDB for CAmkES

```
assembly {
  composition {
      component Sender1 sender1;
      component Receiver1 receiver1;
      connection seL4RPC conn1(from sender1.out1, to receiver1.in1);
  }
  configuration {
    receiver1.debug = "False";
    sender1.debug = "True";
  }
}
```

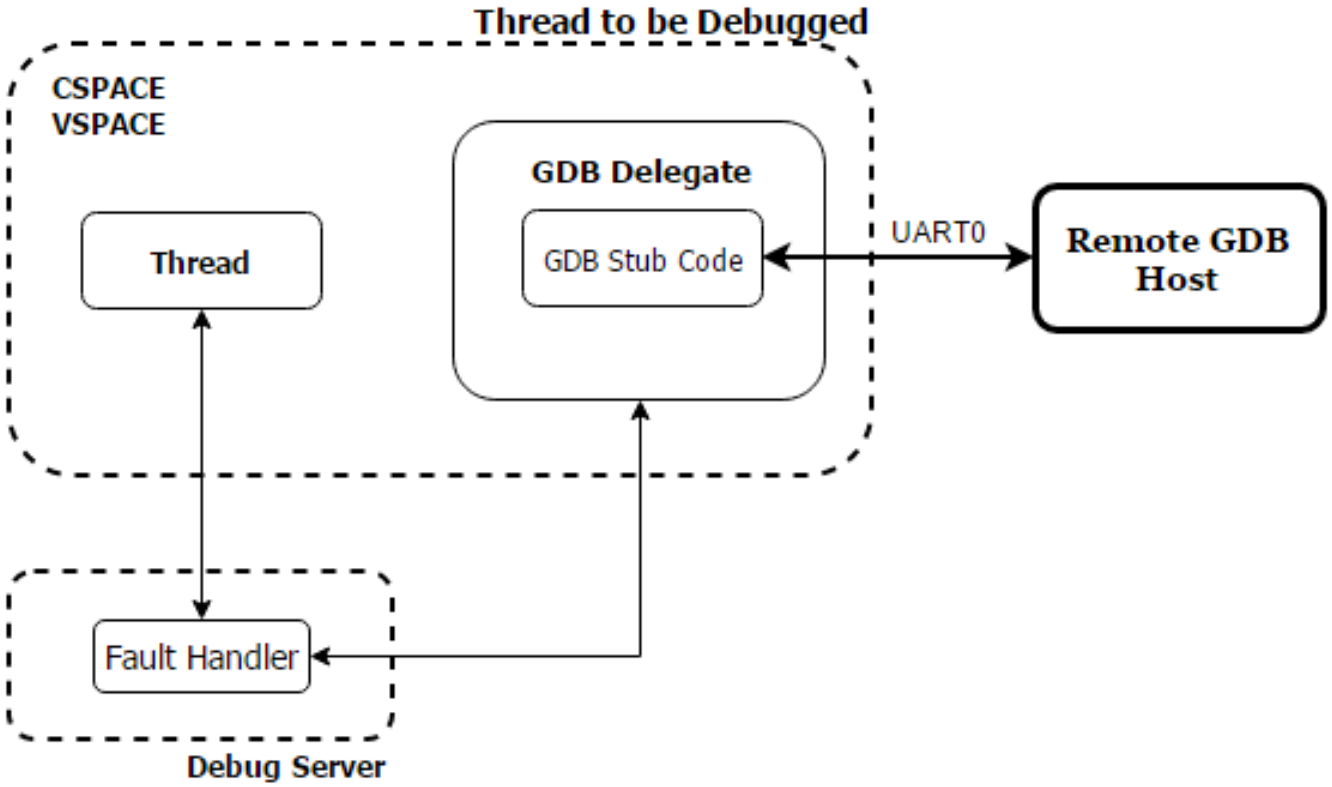
```
assembly {
  composition {
    component debug_server debug;
    component debug_Serial debug_hw_serial;
    component Sender1 sender1;
    component Receiver1 receiver1;
    connection seL4HardwareMMIO uart_mem (from debug.mem, to debug_hw_serial.mem);
    connection seL4Debug debug0_delegate (from debug.sender1_GDB_delegate, to sender1.GDB_delegate);
    connection seL4GDB debug0 (from sender1.fault, to debug.sender1_fault);
    connection seL4HardwareMMIO uart_mem0 (from sender1.mem, to debug_hw_serial.mem);
    connection seL4RPC conn1 (from sender1.out1, to receiver1.in1);
  }
  configuration {
    receiver1.debug = "False";
    sender1.debug = "True";
    debug_hw_serial.mem_attributes = "0xe0000000:0x1000";
  }
}
```

ARM GDB for CAmkES

```
component Sender1 {
  control;
  uses MyInterface out1;
}
```

```
component Sender1 {
  control;
  provides CAmkES_Debug GDB_delegate;
  uses MyInterface out1;
  uses CAmkES_Debug fault;
  dataport Buf mem;
}
```

ARM GDB for CAmkES



ARM GDB for CAmkES

- Fault Endpoint
  - Component generated by debug tool
    - Linked to template files
  - EP is generated by the python script & templates
    - Connected to the fault EP of the debugged thread
  - Fault handler waits for something to occur on the EP
    - Gets the TCB Capability so GDB can read registers, manipulate the instructions, etc…

ARM GDB for CAmkES

- Delegate:
  - Component generated by the debug tool
    - Linked to template files
  - “Provided” by the debugged component.
    - Runs in same vspace/cspace
  - Capability to vspace – allows for instruction modification
    - Required for SW Breakpoints
  - Activated when called from the Fault Endpoint
  - Has a slightly modified version of the stub application for GDB functionality

ARM GDB for CAmkES

- Available commands:
  - Read/Write general registers
  - Memory read/write
  - Set/Remove SW breakpoint
  - Step/Continue
  - Kill

ARM GDB for CAmkES

- Strengths:
  - GDB Server
  - Improved SW Breakpoints
  - ARM/CAmkES compatible
- Weaknesses
  - Cannot debug multiple threads at once

Where Do We Go From Here?

- Multi-Thread capable GDB Server
- HW breakpoints for ARM
- “Detach” operation
- Communicate through Ethernet
- Refactor ARM/x86 applications to be similar
  - Build one application based off the required platform
- Integrate GDB Server to a seL4 IDE
  - Once a seL4 IDE is developed!

Questions??