Ada on CHERI

Daniel King

April 2024

DSbD Partner

Content

- Edge Avionics project
- Porting Ada code to CHERI
- Language Limitations
- Ada vs CHERI bounds checking
- CHERI exception handling

Edge Avionics - Consortium

The RAF Rapid Capability Office (RCO) is funding a collaboration between GE Aerospace UK, its partners and Dstl to demonstrate Morello / CHERI in a defence environment

Edge Avionics - Project Aims

- Demonstrate Morello / CHERI in a defence environment
- Testing performance at scale
  - Dstl owned and modified air platform mission system will be used to check the impact of the new security controls
- Investigating legacy software rework overheads
- Evaluate resistance to common attack vectors

Edge Avionics - Phase II

- **Rehosted** Dstl proprietary avionics mission system (~1.5 million SLOC)
- **Compiled** with GNAT Pro for Morello (GCC and LLVM)
- **Running on** VxWorks for Morello
- **On top of** Wind River Helix Platform Hypervisor for Morello
- **On** an integrated Arm Morello development board connected to a GE Remote Interface Unit (RIU)

GNAT Pro Ada for Morello

- We have ported the GNAT Pro bare-metal toolchain to purecap Morello
- Both GCC and LLVM toolchains
- Three flavours of bare-metal runtime:

Runtime Profile	Deallocation?	Tasking?	Exception Propagation?
light	no	no	no
light-tasking	no	yes	no
embedded	yes (newlib)	yes	yes

- Next phase: port the toolchains to Wind River VxWorks for Morello

Capability Types in Ada

- `System.Address`
  - Represents a machine address capable of addressing individual storage elements
  - 128-bit capability type on Morello
  - An opaque type (definition is private), but has arithmetic operators for: +, -, mod
- `System.Storage_Elements.Integer_Address`
  - Integer representation of an address
  - Defined as a 64-bit unsigned integer on Morello
  - **not** a capability
- Functions exist to convert between `Address` and `Integer_Address`
- New package `Interfaces.CHERI` defined with operations to manipulate capabilities

Porting Ada Programs to CHERI

- Very low effort in our experience
- Majority of pointer/address arithmetic is hidden “behind the scenes” by the compiler
- Low-level code (e.g. memory allocators) may need minor changes
  - Adjusting address arithmetic to preserve capability provenance
  - Fixing assumptions that no longer hold on CHERI targets

Address-to-Integer conversions

- Unchecked_Conversion can't be used to convert between Address / Integer_Address
  - The source and target types are required to have the same size and alignment
  - This assumption breaks on Morello

```
--  Compile time error on Morello
function Convert is new Ada.Unchecked_Conversion
  (Source => System.Address,
   Target => System.Storage_Elements.Integer_Address);

Addr     : System.Address := Get_Addr;
Int_Addr : System.Storage_Elements.Integer_Address := Convert (Addr);
```

- Use `System.Storage_Elements.To_Integer` instead

```
Int_Addr := System.Storage_Elements.To_Integer (Addr);
```

Address Arithmetic

- Address arithmetic is sometimes needed, e.g. to align an address
- Capability provenance needs to be preserved during the calculation
- Integer_Address is not a capability

```
Int_Addr := To_Integer (Addr);
Int_Addr := Int_Addr - (Int_Addr  mod 16);
Addr     := To_Address (Int_Addr);
--  Addr is now an invalid capability
```

- Use the arithmetic operations in `System.Storage_Elements` to perform arithmetic directly on type `System.Address`:

```
with System.Storage_Elements;  use System.Storage_Elements;

Addr := Addr - (Addr  mod 16);
```

Language Limitations

- Reading `System.Address` from streams
  - Would allow creation of pointers to arbitrary addresses
  - Limited use cases for this (if any)
  - Workaround: stream an Integer_Address and manually mint an Address (capability)
- `Ada.Tags.Internal_Tag`
  - Returns the “tag” corresponding to a given external tag (a string)
  - GNAT implements external tags as a string representation of an address
  - Would allow creation of pointers to arbitrary addresses

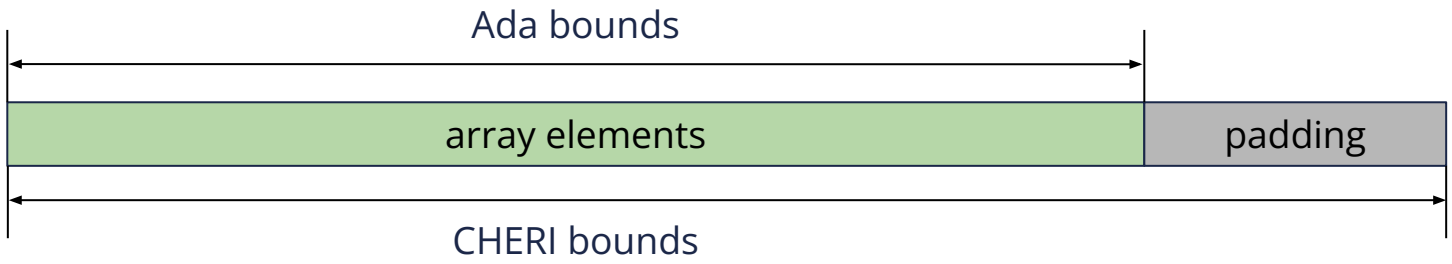
Covering the Gaps

- Some language constructs are not covered by language-defined run-time checks
- The term “unchecked” is used for most of these:
  - Unchecked_Conversion
  - Unchecked_Deallocation
- Memory Overlays are also unchecked
  - Programmer is responsible for ensuring correct size & alignment
  - CHERI catches misuse where the language doesn’t:

```
declare
   U8  : Unsigned_8;
   U32 : Unsigned_32 with Import, Address => U8'Address;
begin
   U32 := 0; --  CHERI bounds error
end;
```

Ada vs CHERI Bounds Checks

- Ada has language-defined run-time bounds checking on arrays
- Can we replace those software run-time checks with CHERI hardware checks?
- The Problem: Bounds compression on Morello
  - For large arrays, the CHERI bounds are imprecise (for bounds alignment)
  - Would allow accesses past the end of the array



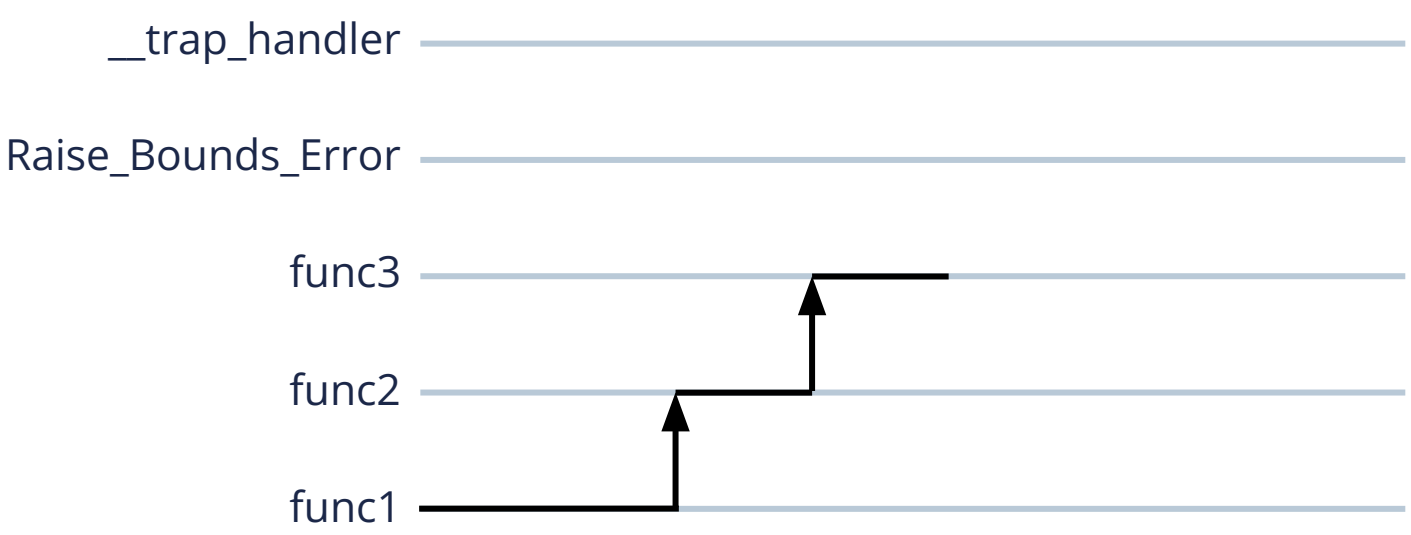
CHERI Exception Handling

- Goal: Allow CHERI exceptions to be caught and handled like regular Ada exceptions
- Four new exception types defined for bounds / permissions / tag / sealed CHERI errors

```
procedure Example is
   U8  : Unsigned_8;
   U32 : Unsigned_32 with Import, Address => U8'Address;
begin
   U32 := 0; --  Triggers a capability bound error

exception
   when Interfaces.CHERI.Capability_Bounds_Error =>
      --  Handle the error here
end Example;
```

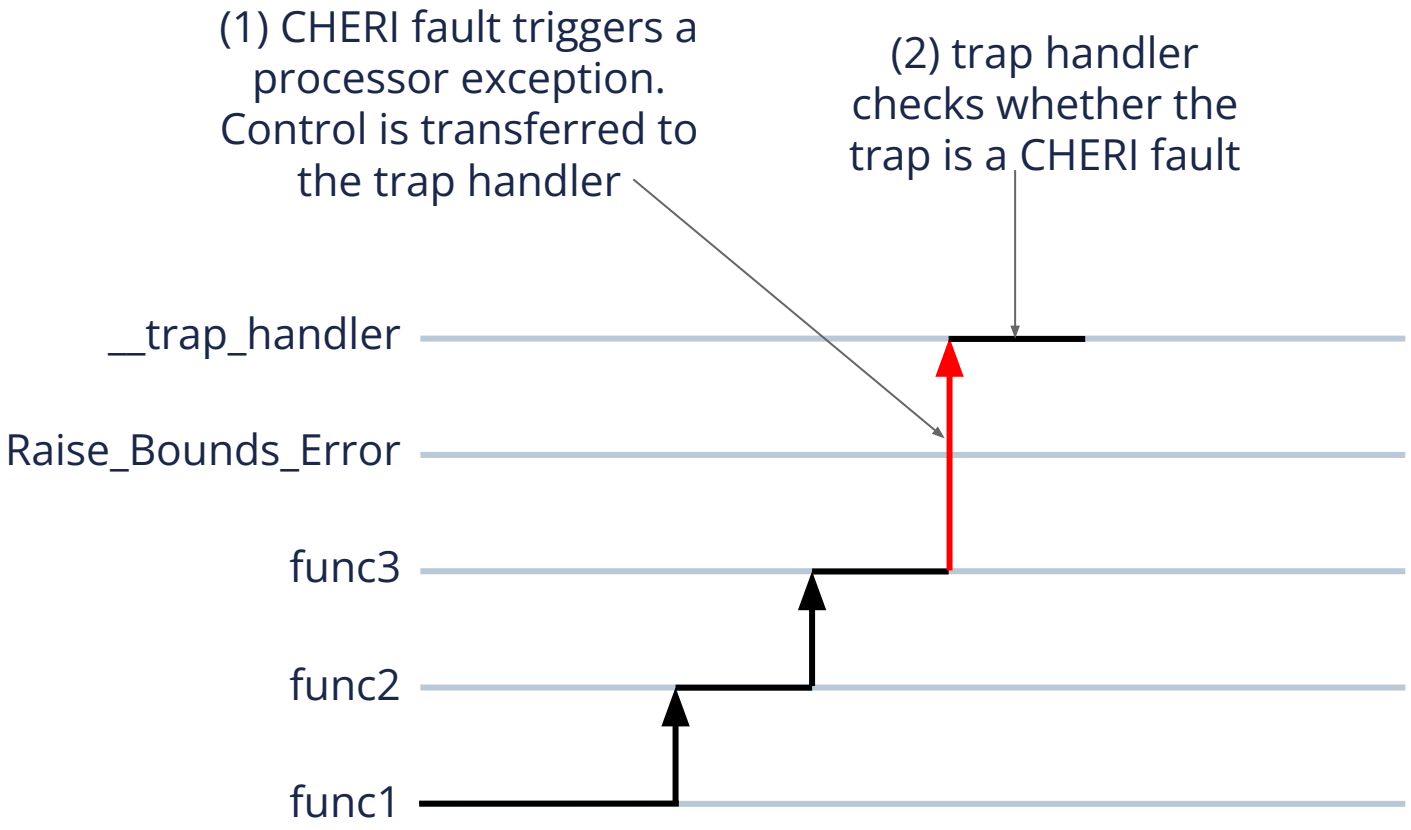
CHERI to Ada Exception Conversion



Call stack

func1 call frame
func2 call frame
func3 call frame

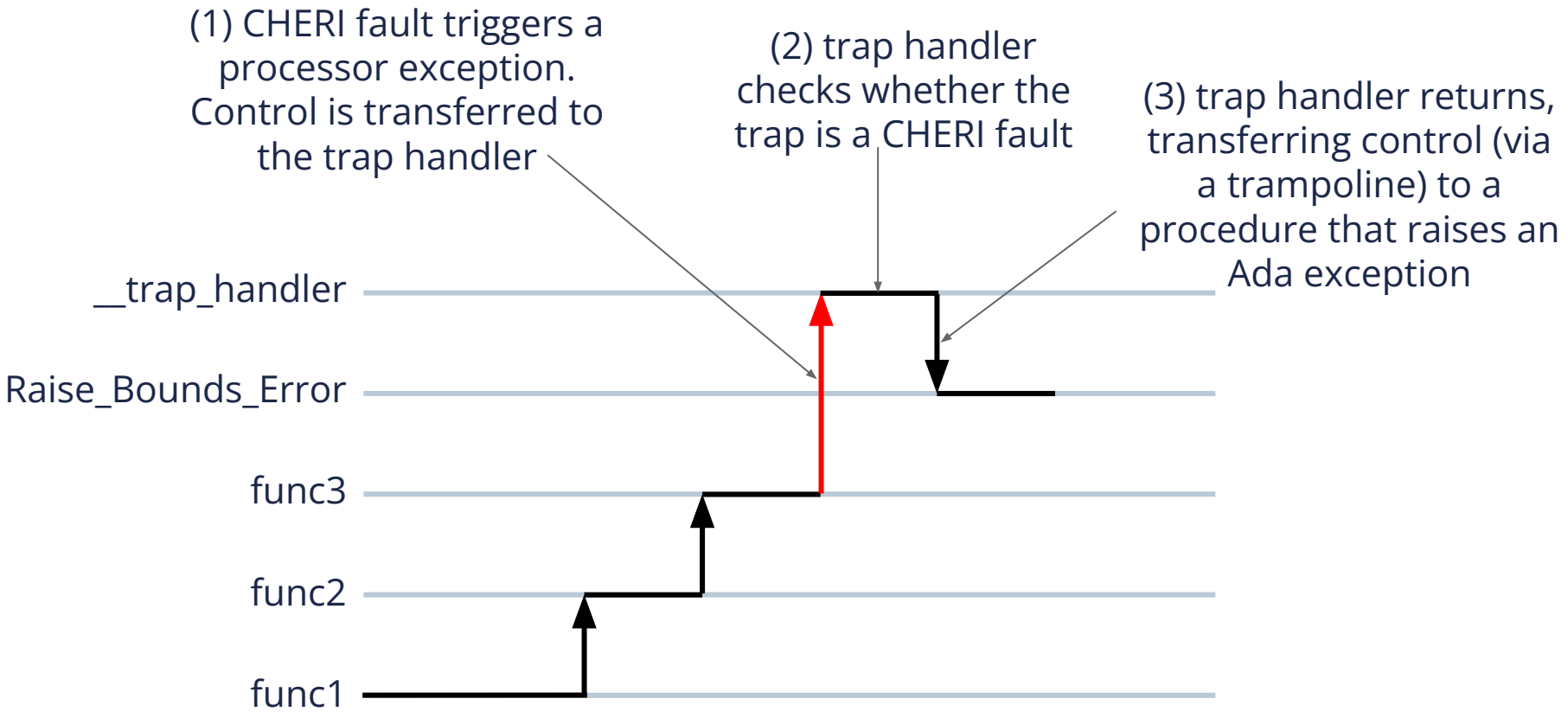
CHERI to Ada Exception Conversion



Call stack

func1 call frame
func2 call frame
func3 call frame
signal frame

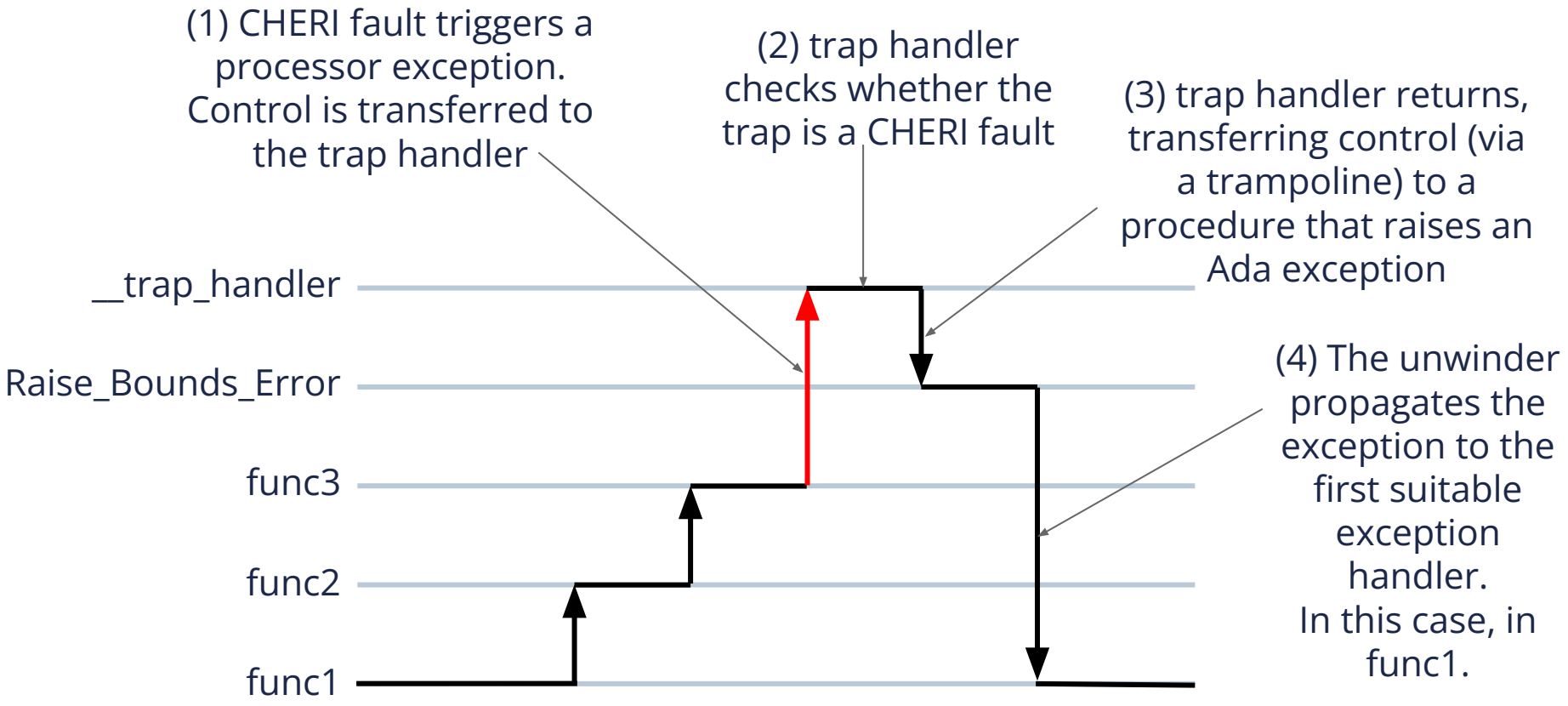
CHERI to Ada Exception Conversion



Call stack

func1 call frame
func2 call frame
func3 call frame
signal frame
Raise_Bounds_Error call frame

CHERI to Ada Exception Conversion



Call stack

func1 call frame

Exception Handling Benefits

- Reduce the effects of unexpected memory safety errors
  - No need to “stop the world” - failure limited to individual task
  - Unaffected tasks can continue execution
- Fail secure / degraded
  - Tasks can “fail secure”
  - Tasks dependent on the failed task could enter a “degraded” mode of operation
- Recovery from errors
  - Restart affected tasks
- Vulnerability logging
  - Log the system state and triggering scenario to aid in reproducing & fixing the bug

Testing on Morello

- CHERI can uncover bugs that go undetected on conventional targets
  - We found a memory safety compiler bug by running existing test suites on Morello
  - Both Valgrind and AddressSanitizer failed to detect the bug
  - Details presented in pre recorded presentation for Nov 2023 DSbD all hands event
- Recommendation: run unit tests / fuzzing on a CHERI target (hardware or emulator)

Conclusions

- Edge Avionics project is evaluating CHERI in a defence environment
- Ada code runs on purecap Morello with very little effort and few limitations
- CHERI provides safety for unchecked parts of the Ada language
- CHERI errors can be handled in Ada using existing Ada exception handling machinery
- In addition to being a good deployment target (due to hardware security measures), CHERI is capable verification tool and can find anomalies within software that other tools fail to detect.

Thank you

Daniel King

dmking@adacore.com