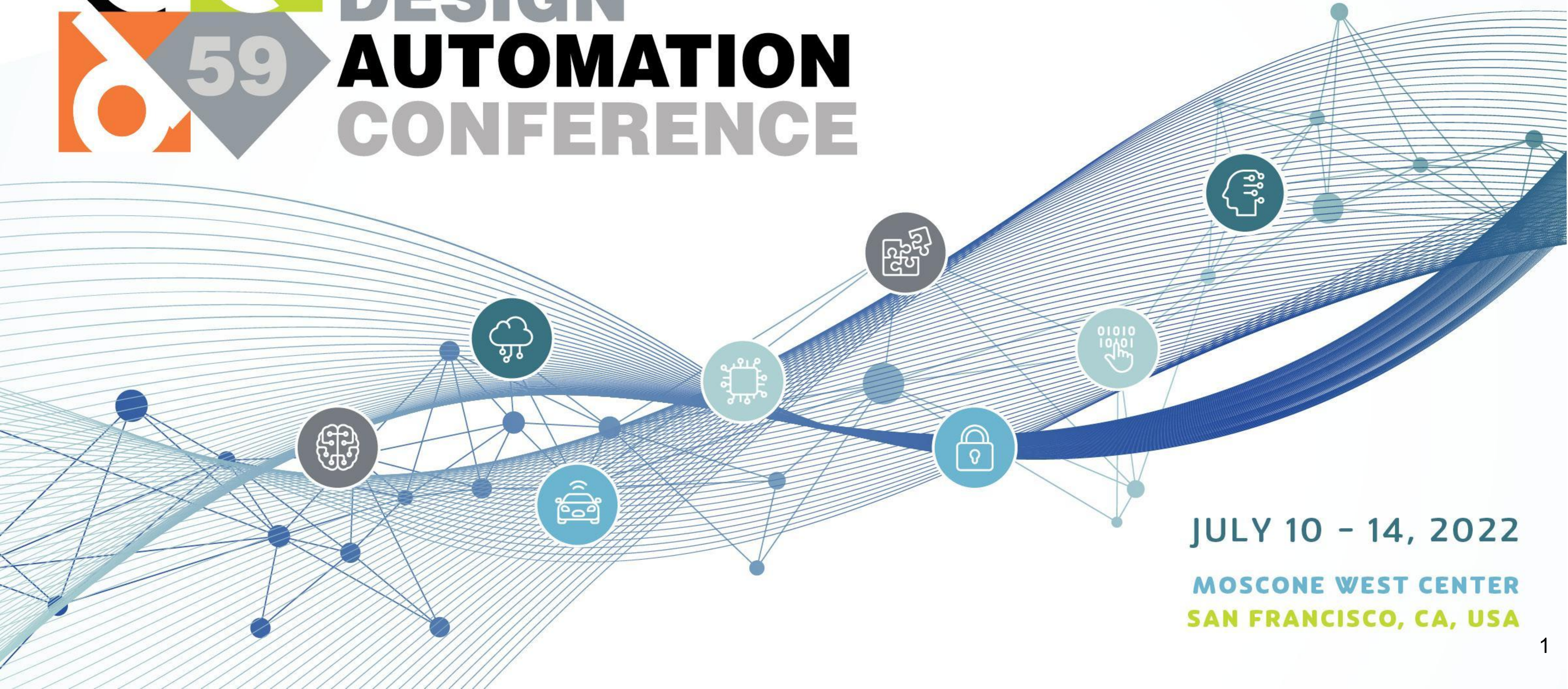
59

DESIGN AUTOMATION CONFERENCE



JULY 10 - 14, 2022

MOSCONE WEST CENTER

SAN FRANCISCO, CA, USA

ASAP: Reconciling Asynchronous Real-Time Operations and Proofs of Execution in Simple Embedded Systems

Adam Caulfield — Rochester Institute of Technology

Norrathep Rattanavipanon — Prince Songkla University Phuket

Ivan De Oliveira Nunes — Rochester Institute of Technology

Embedded (IOT) Devices

- Connect physical & digital worlds
- Resource constrained
- Targeted by exploits & attacks

Safety Critical Systems Rely on Embedded Devices

Embedded Devices used for safety critical settings

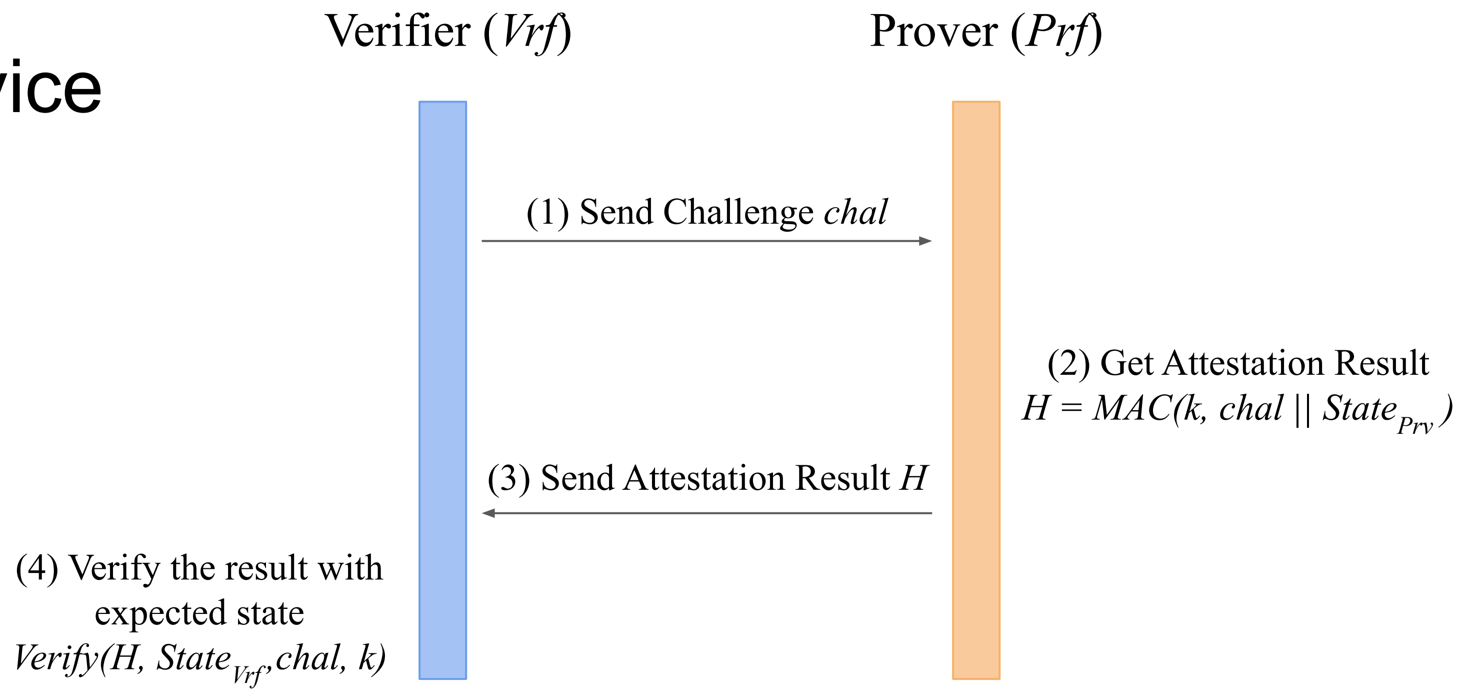
- Smoke detector in a household
- Remote controlled syringe pump for telemedicine

Device controllers rely on sensor values and signals from the remote device being correct and untampered

Remote Attestation (RA)

- Remotely verify the binary currently installed on the device

- Enforce protocol through hardware support


Verifier (*Vrf*)
Prover (*Prf*)
(1) Send Challenge *chal*
(2) Get Attestation Result
$H = MAC(k, chal \| State_{Prv})$
(3) Send Attestation Result *H*
(4) Verify the result with expected state
$Verify(H, State_{Vrf}, chal, k)$

Proof of Execution (*PoX*)

- Require hardware to monitor additional security properties to provide the verifier with a *PoX:*

  Existing architecture *APEX* (USENIX Security ‘20) provides:
  - Software Immutability
  - Memory Protection
  - Execution Atomicity
  - Response Protection

Proof of Execution (*PoX*)

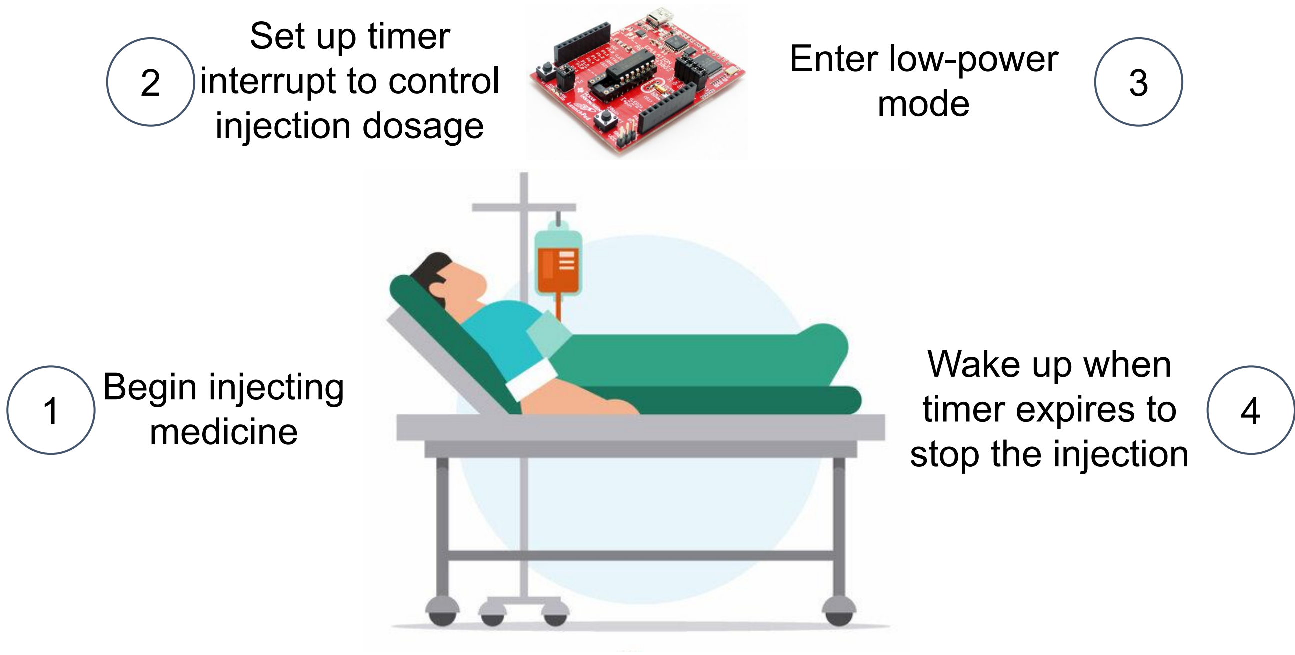
- Require hardware to monitor additional security properties to provide the verifier with a *PoX:*

  Existing architecture *APEX* (USENIX Security '20) provides:
  - Software Immutability
  - Memory Protection
  - **Execution Atomicity**
  - Response Protection

  Limitation: Asynchronous processing cannot benefit from *PoX*

Remote Syringe Pump for Telemedicine

- Use of timer-based interrupt allows for energy efficiency
- Cannot benefit from *PoX* since is asynchronous


1 Begin injecting medicine
2 Set up timer interrupt to control injection dosage
3 Enter low-power mode
4 Wake up when timer expires to stop the injection

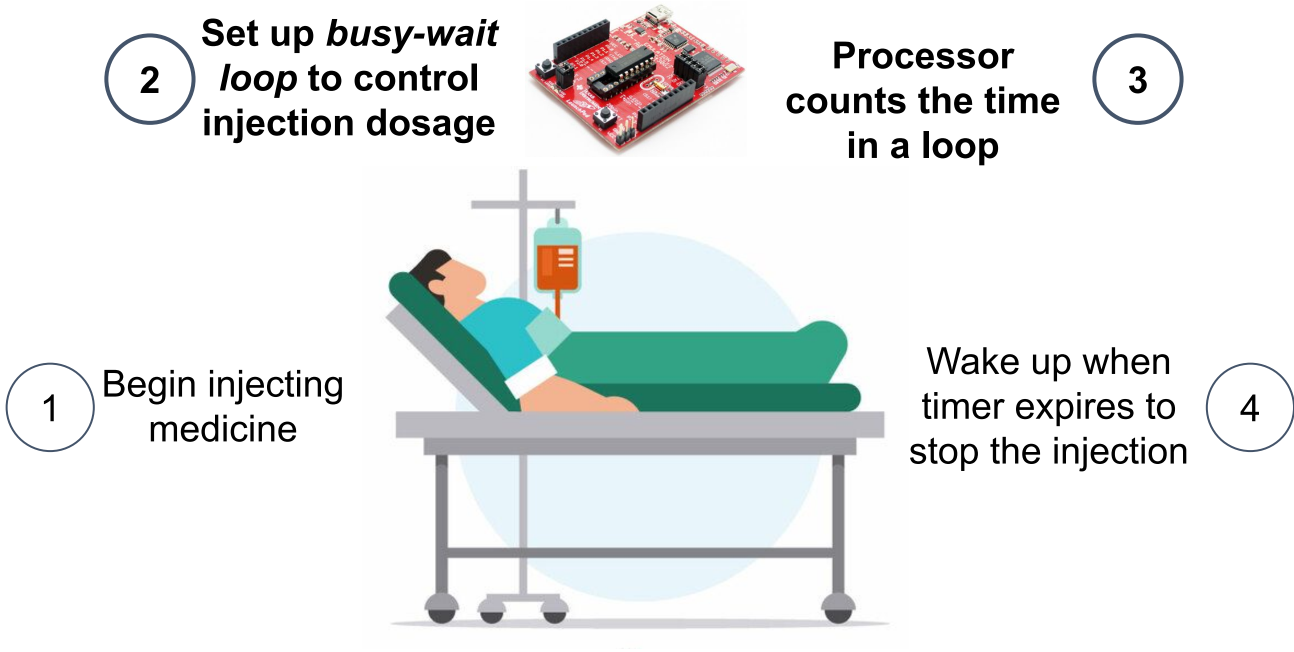
Remote Syringe Pump for Telemedicine

Use *busy-wait* approach

- Disable all interrupts
- Require the processor to wait using a for loop

Problems:

- Power consumption
- Uninterruptible in case of emergency



2 Set up *busy-wait loop* to control injection dosage
Processor counts the time in a loop 3
1 Begin injecting medicine
Wake up when timer expires to stop the injection 4

ASAP: Architecture for Secure Asynchronous Processing in *PoX*

- Builds upon existing *PoX* architecture to support Asynchronous Processing

- Achieved by ensuring the Ephemeral Immutability and Integrity of:

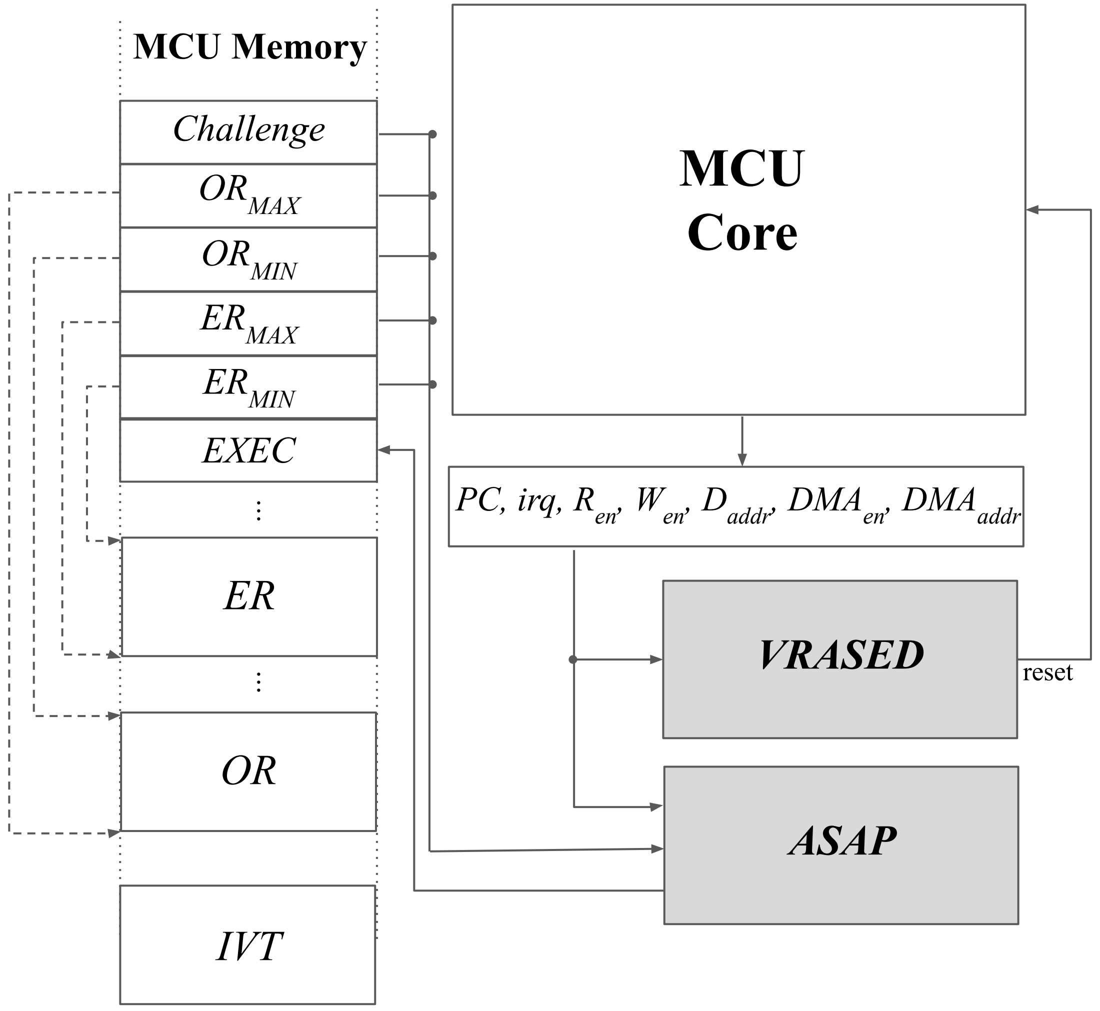
  1) The Interrupt Vector Table (IVT) within the MCU address space

  2) Any interrupt service routines (ISRs) that are known and expected prior to device deployment

System Overview

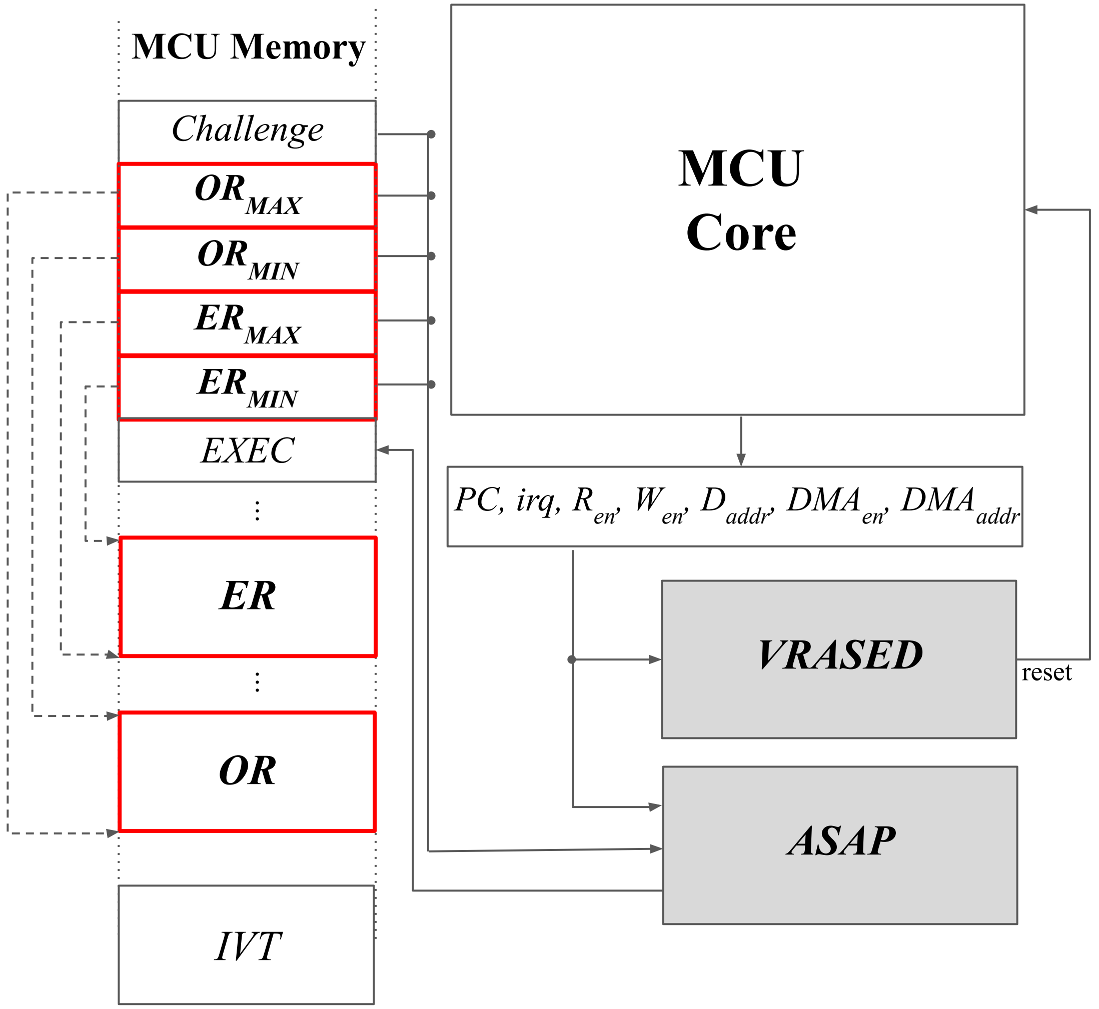
- *ASAP* monitors MCU signals to determine the state

- Alongside architecture *VRASED* which supports *RA*



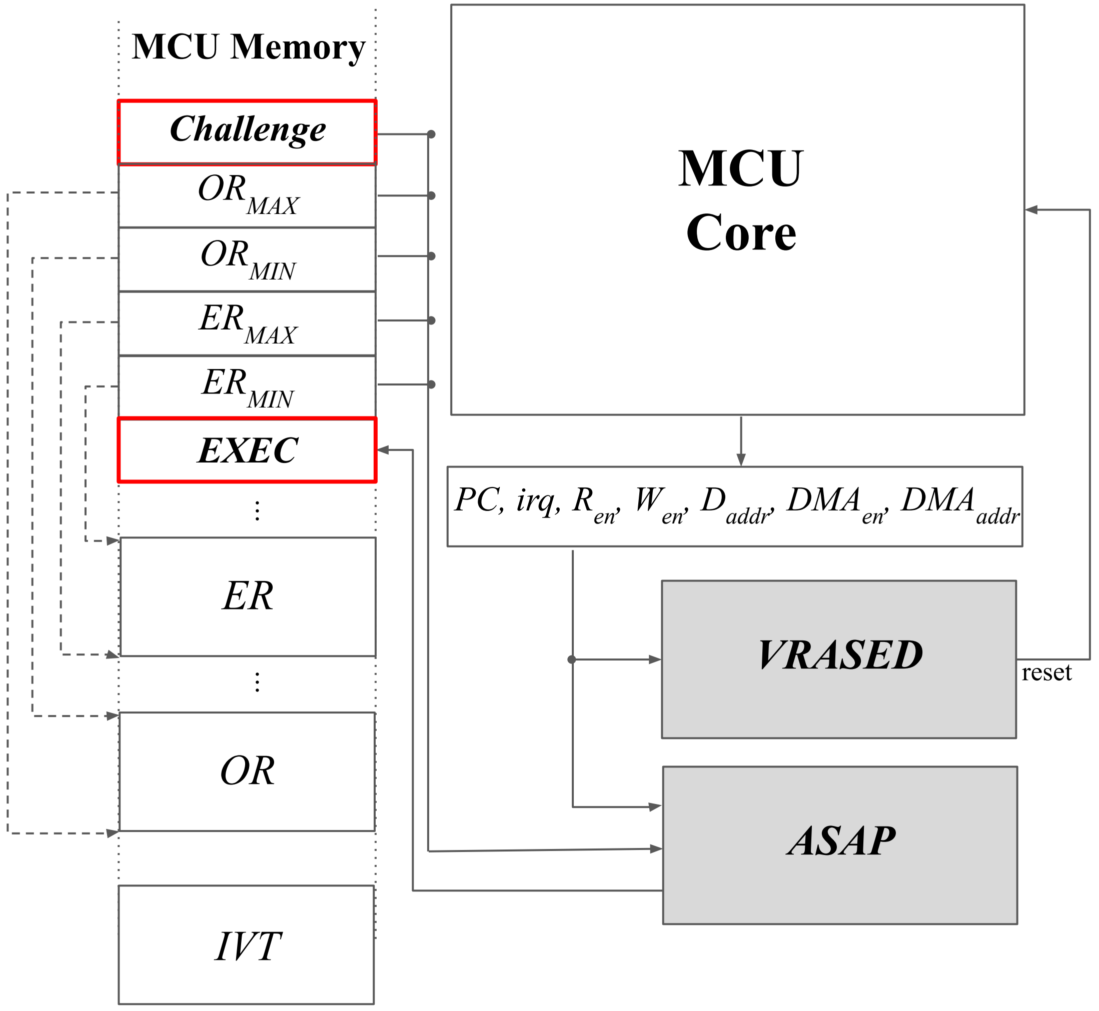
System Overview

- In MCU memory, regions are reserved for the
  - Executable Region (*ER*)
  - Output Region (*OR*)
- Their start and end addresses are stored for *ASAP* to monitor



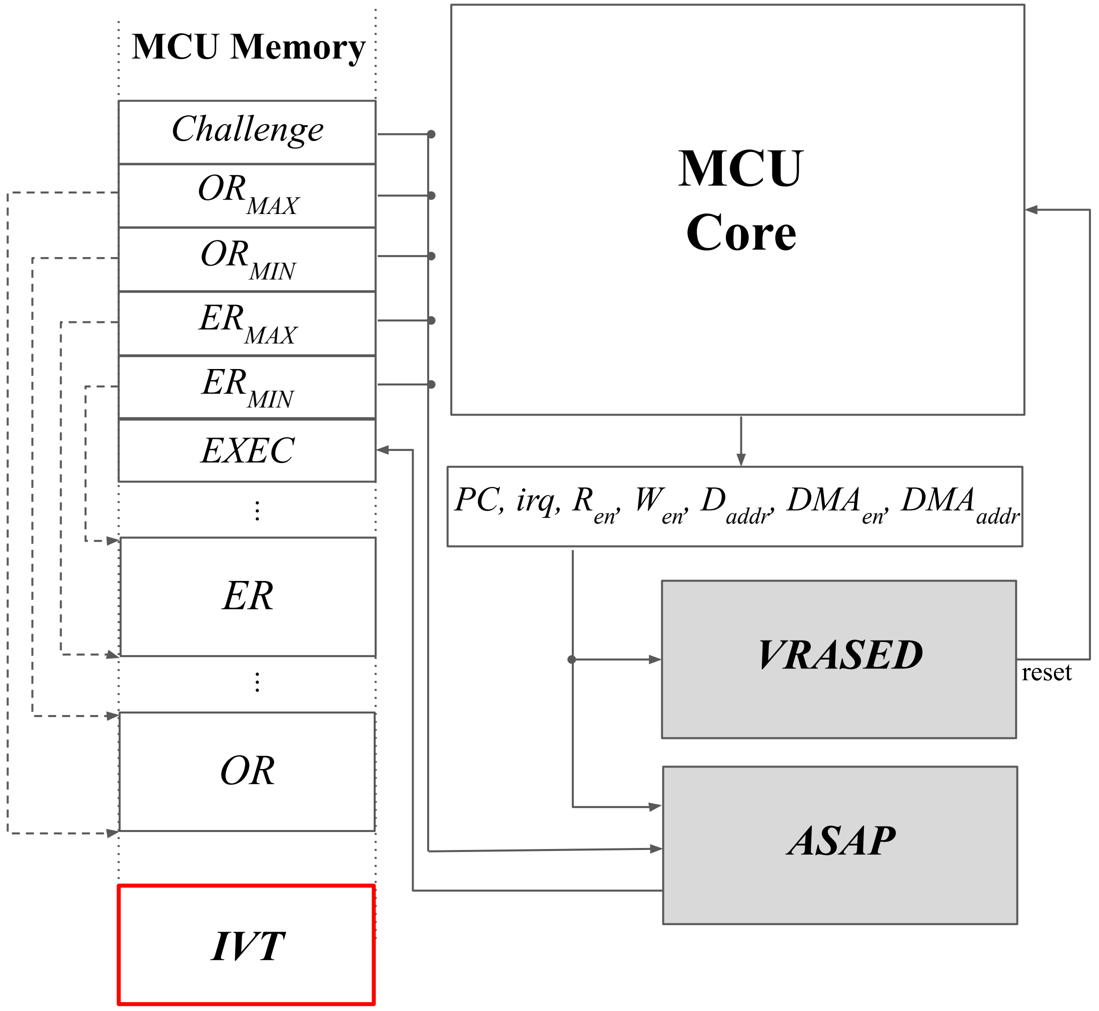
System Overview

- The *Challenge* from the verifier is stored & monitored
- *ASAP* sets *EXEC* to zero if any *PoX* violation occurs



System Overview

- The IVT is the last 16 entries in the address space
- Start and end addresses are known & monitored by *ASAP*



ASAP Security Properties

1) IVT Immutability & Integrity

- Monitor the CPU & DMA for write attempts to the IVT
- Set *EXEC* to zero if a write attempt occurs

IVT Immutability

$If\ (DMA_{en}\ \&\ DMA_{addr} \in IVT)\ |\ (W_{en}\ \&\ D_{addr} \in IVT) \rightarrow !EXEC$

Execution exits at ER_{max}

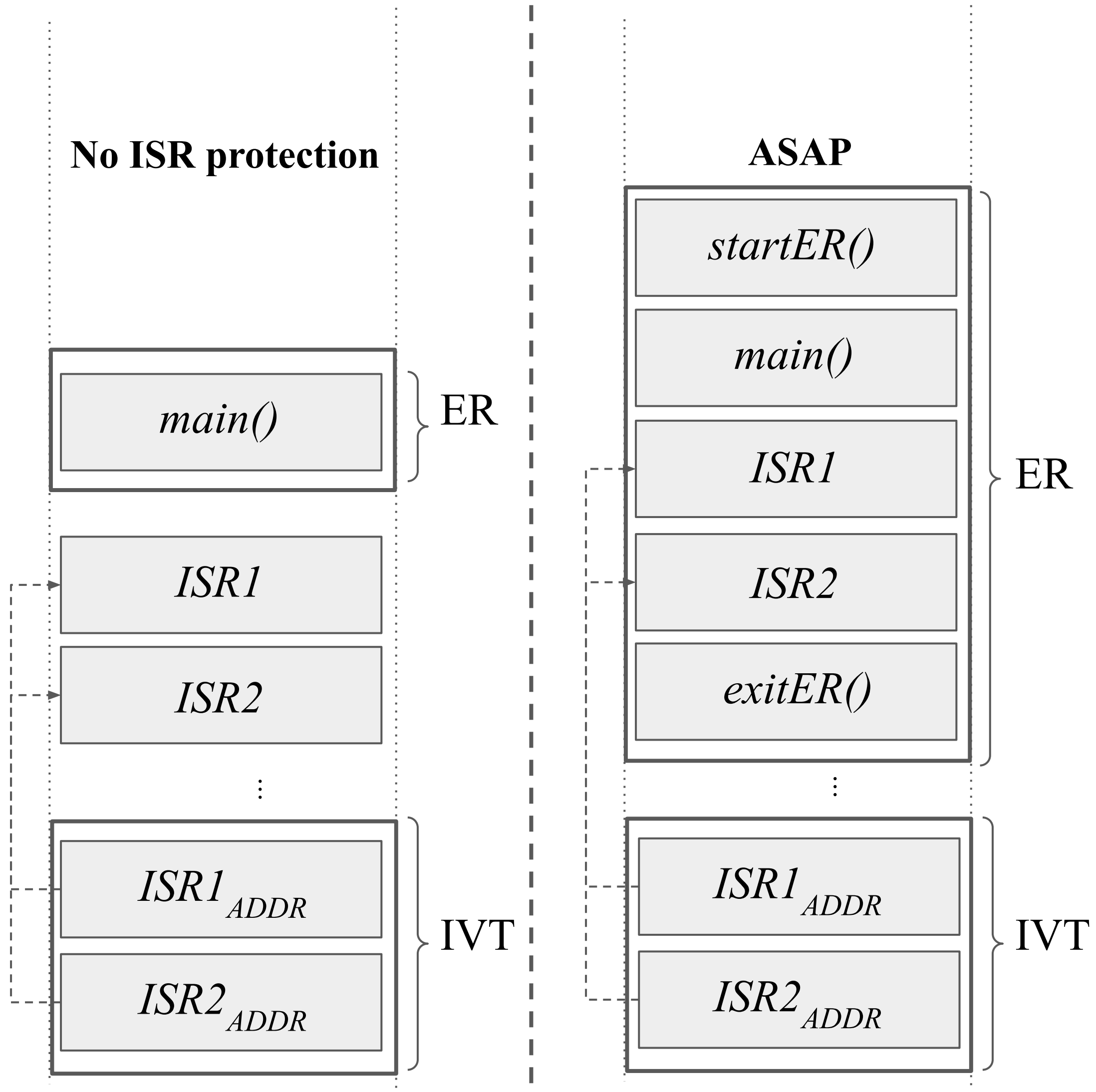
$If\ (PC \in ER)\ \&\ !(next\ PC \in ER) \rightarrow PC = ER_{max}\ \&\ !EXEC$

Execution starts at ER_{min}

$If\ !(PC \in ER)\ \&\ next\ PC \in ER \rightarrow next\ PC = ER_{min}\ \&\ !EXEC$

ASAP Security Properties

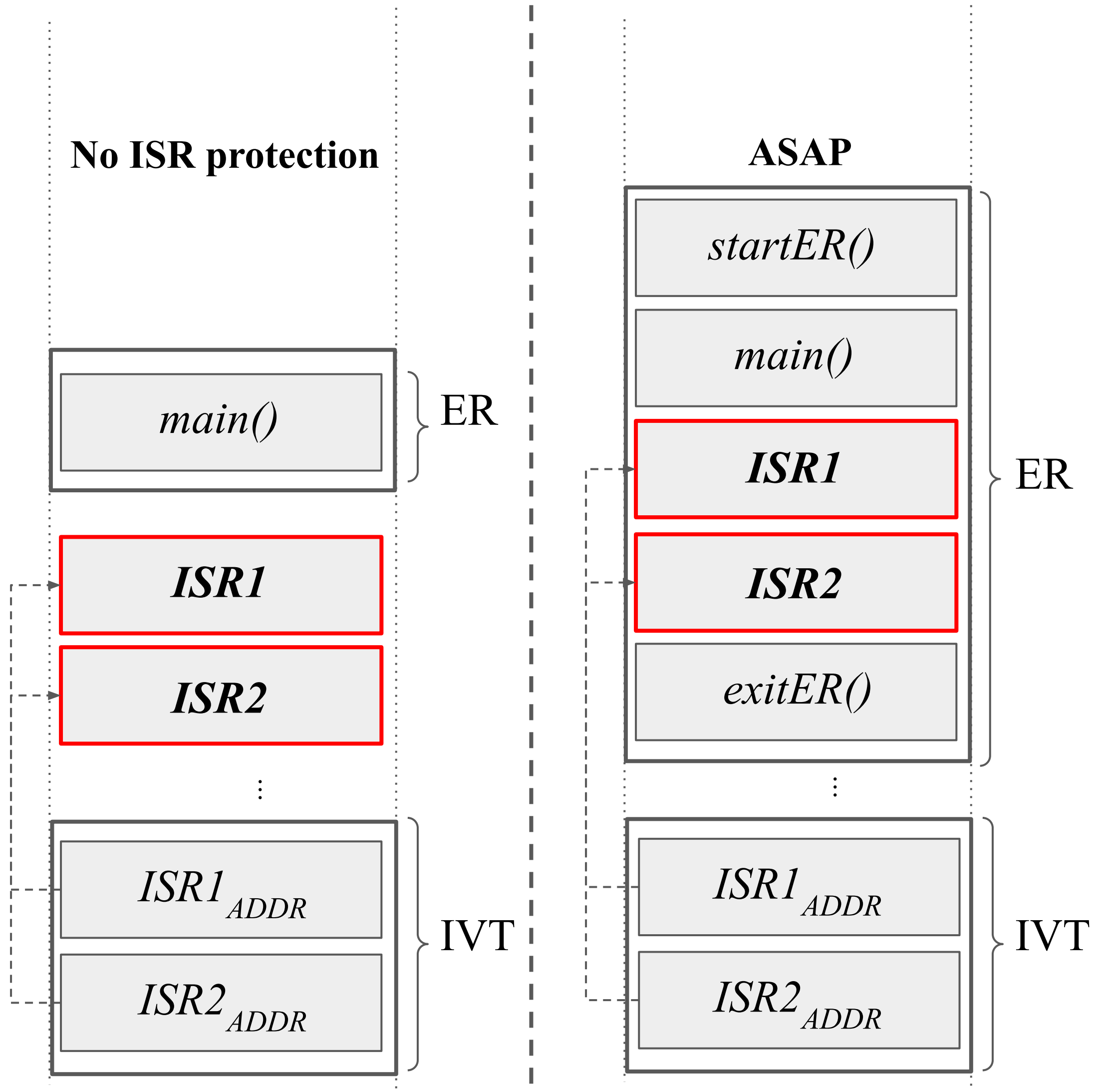
2) ISR Immutability & Integrity



ASAP Security Properties

2) ISR Immutability & Integrity

- Selective linking in ASAP to ensure all expected ISRs are captured in *ER*

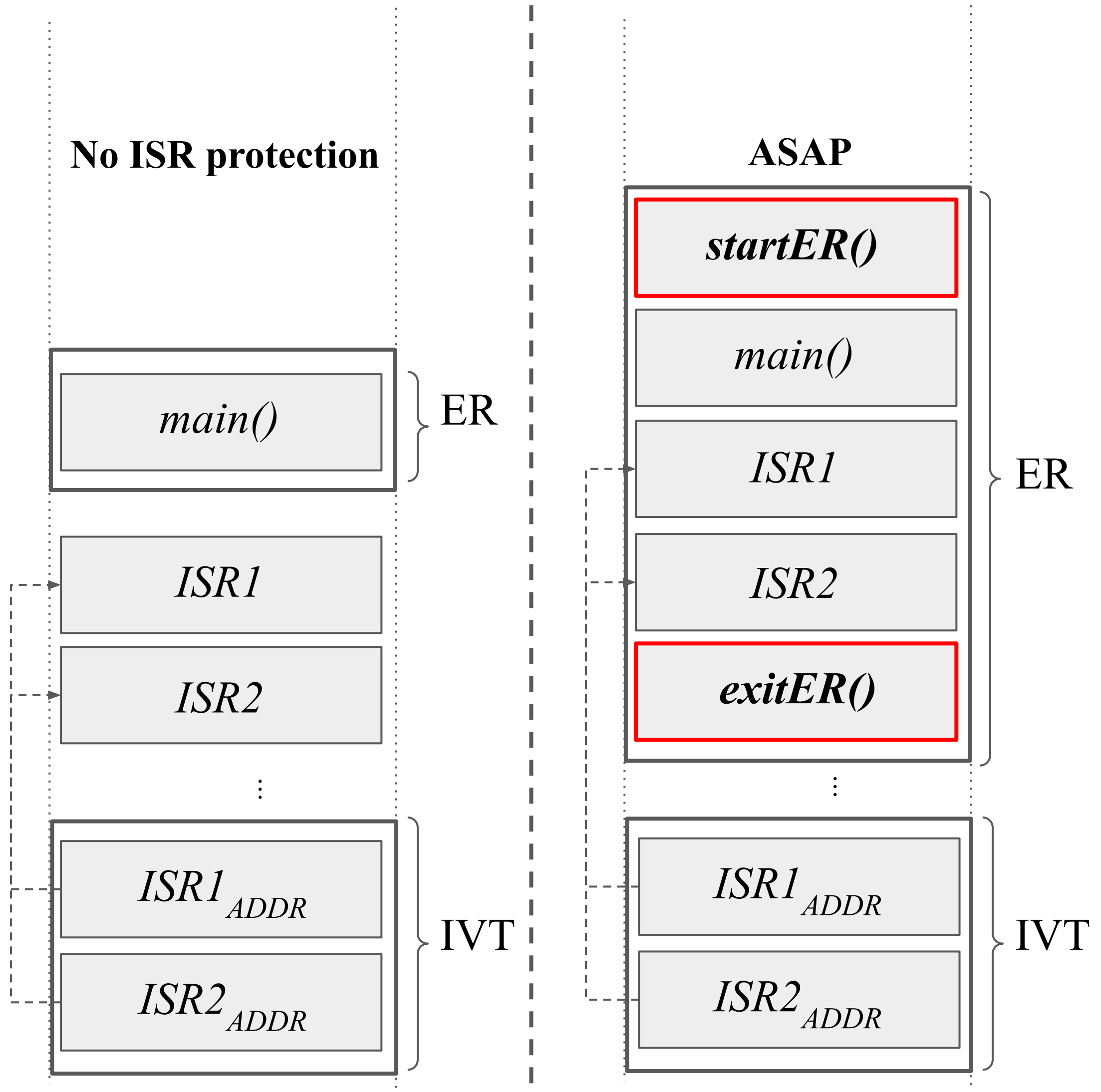


ASAP Security Properties

2) ISR Immutability & Integrity

- Selective linking in ASAP to ensure all expected ISRs are captured in *ER*

- Ensure entry and exit at fixed points



ASAP Security Properties

2) ISR Immutability & Integrity

- Selective linking in ASAP to ensure all expected ISRs are captured in *ER*

- Ensure entry and exit at fixed points

IVT Immutability

$If\ (DMA_{en}\ \&\ DMA_{addr} \in IVT)\ |\ (W_{en}\ \&\ D_{addr} \in IVT) \rightarrow !EXEC$

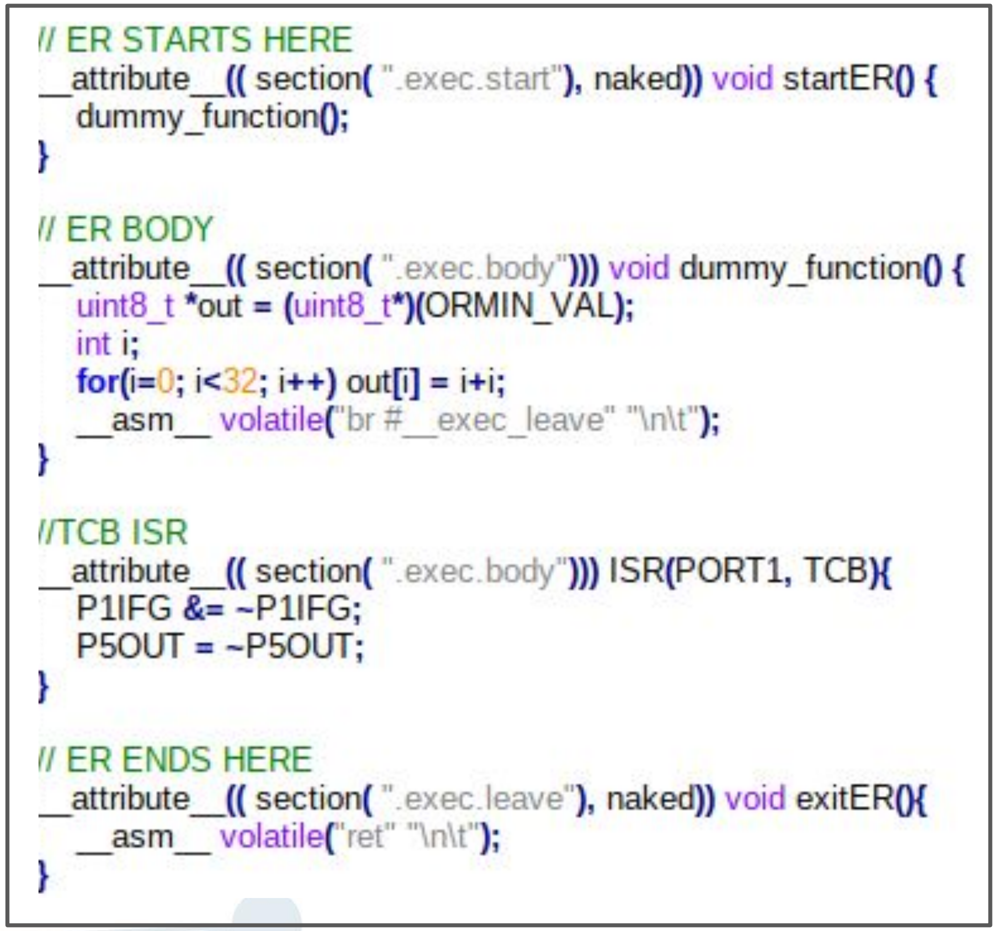
Execution exits at *ER*$_{max}$

$If\ (PC \in ER)\ \&\ !(next\ PC \in ER) \rightarrow PC = ER_{max}\ \&\ !EXEC$

Execution starts at *ER*$_{min}$

$If\ !(PC \in ER)\ \&\ next\ PC \in ER \rightarrow next\ PC = ER_{min}\ \&\ !EXEC$

Selective Linking example



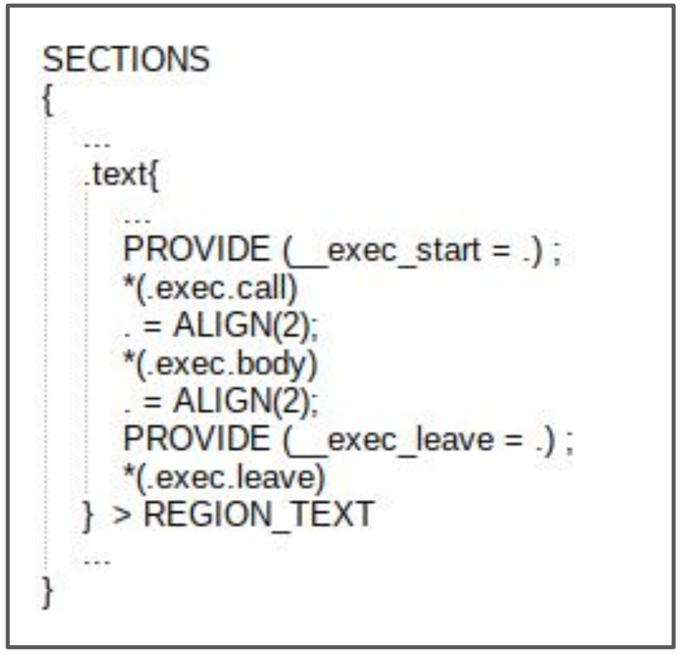
```
// ER STARTS HERE
__attribute__(( section( ".exec.start"), naked)) void startER() {
   dummy_function();
}

// ER BODY
__attribute__(( section( ".exec.body"))) void dummy_function() {
   uint8_t *out = (uint8_t*)(ORMIN_VAL);
   int i;
   for(i=0; i<32; i++) out[i] = i+i;
   __asm__ volatile("br #__exec_leave" "\n\t");
}

//TCB ISR
__attribute__(( section( ".exec.body"))) ISR(PORT1, TCB){
   P1IFG &= ~P1IFG;
   P5OUT = ~P5OUT;
}

// ER ENDS HERE
__attribute__(( section( ".exec.leave"), naked)) void exitER(){
   __asm__ volatile("ret" "\n\t");
}
```

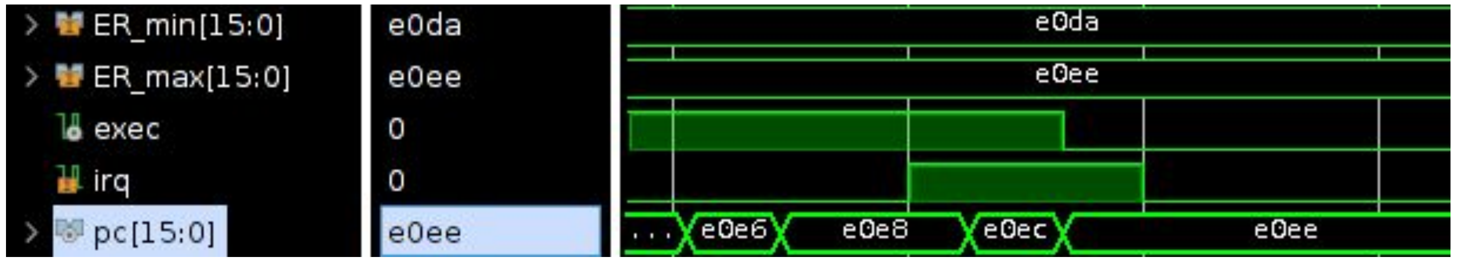
Example code



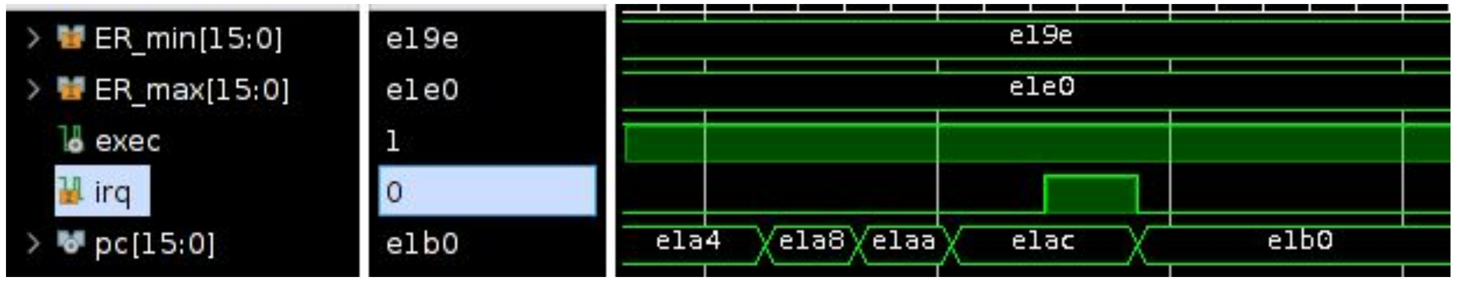
```
SECTIONS
{
   ...
   .text{
      ...
      PROVIDE (__exec_start = .) ;
      *(.exec.call)
      . = ALIGN(2);
      *(.exec.body)
      . = ALIGN(2);
      PROVIDE (__exec_leave = .) ;
      *(.exec.leave)
   } > REGION_TEXT
   ...
}
```

Linker Script Excerpt

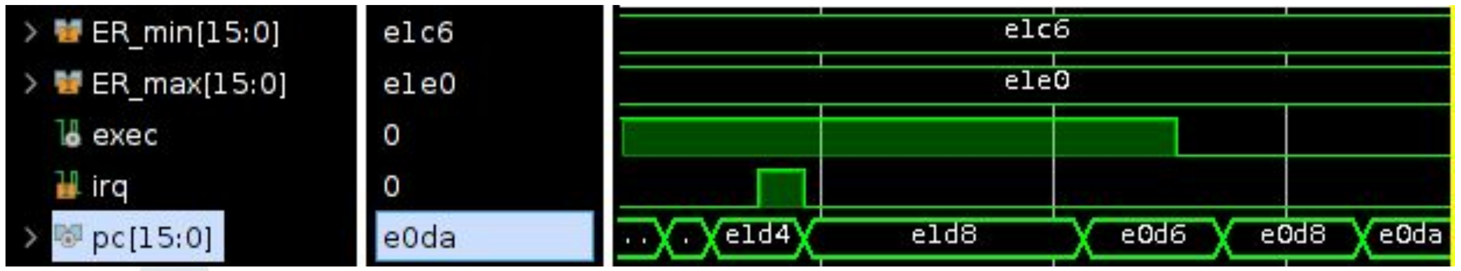
Proof of Concept



APEX interrupted



ASAP valid ISR



ASAP invalid ISR

Conclusion

- *ASAP*: architectural support for *PoX* in MCUs that operate in real time
- Builds upon existing PoX architecture
- Requires minimal hardware modifications
- Selective linking makes protected ISRs easily configurable
- Allows for all software to benefit from PoX security guarantees

DESIGN AUTOMATION CONFERENCE

Thank you