SYNOPSYS®
Silicon to Software™

Zephyr RTOS for ARC Processors: From “Nano Kernel" to Heterogeneous Cluster

Alexey Brodkin, Engineering Manager

Synopsys ARC® Processor Summit 2022

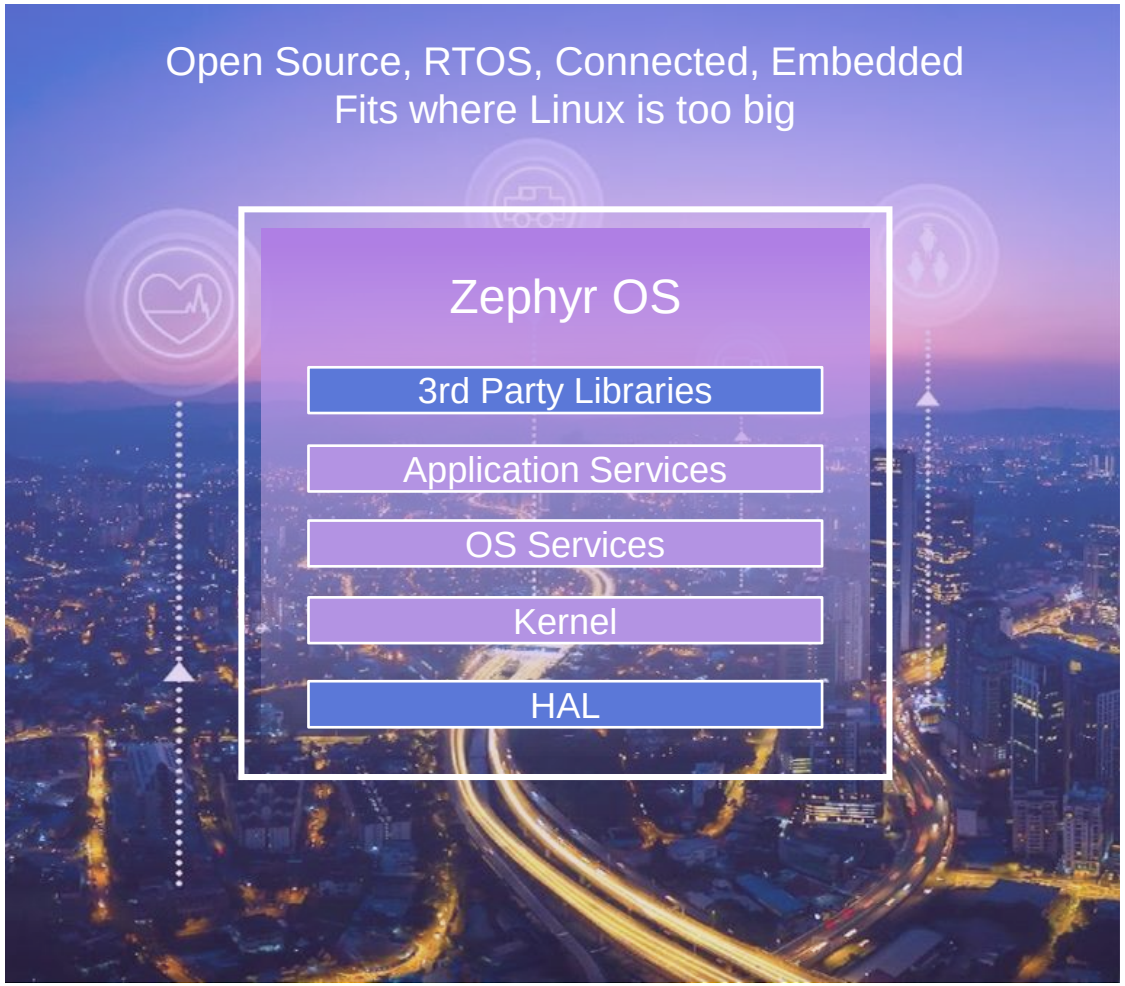
Agenda

- Zephyr RTOS evolution
- ARC processors in Zephyr RTOS
- Software features
- New hardware features

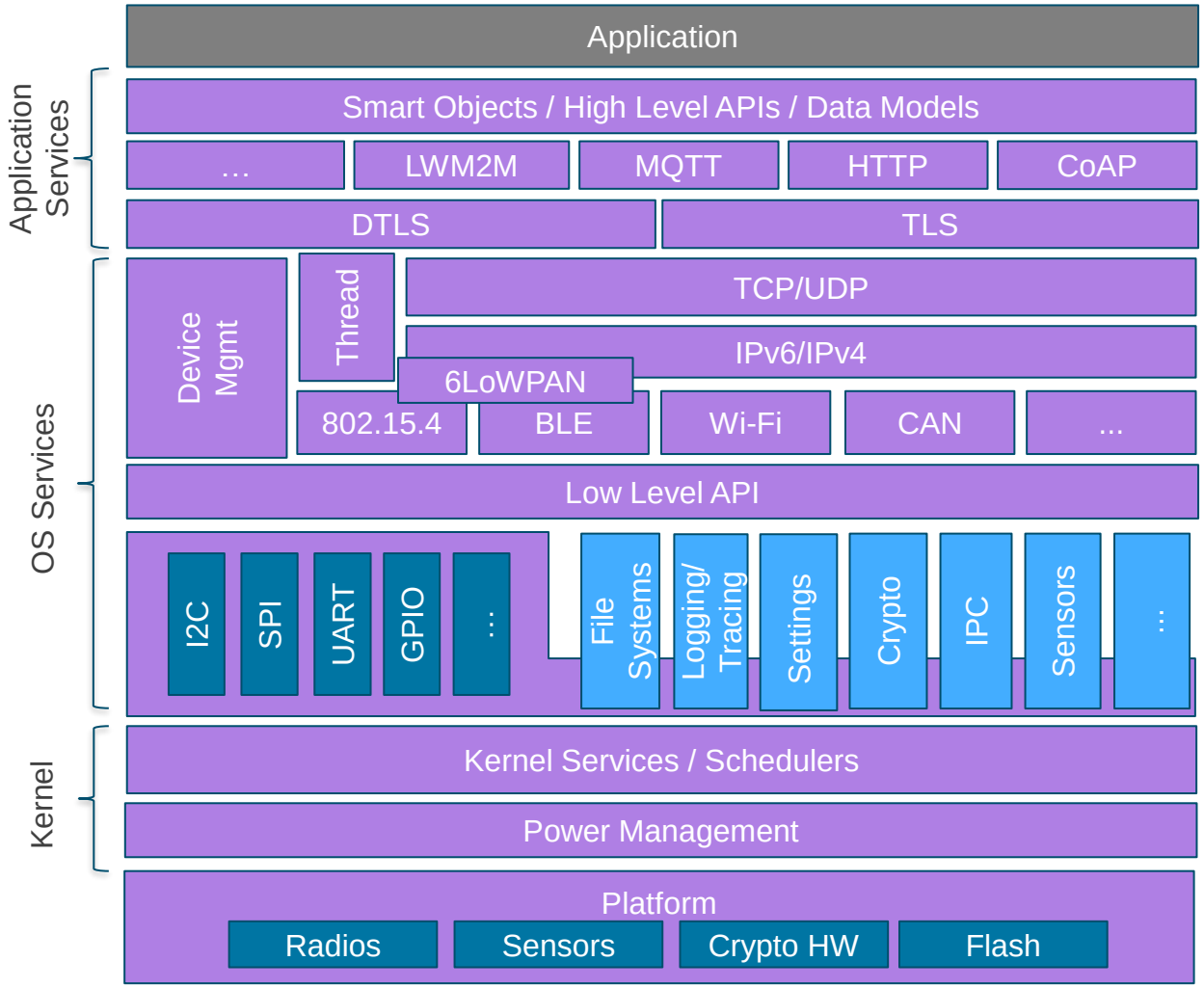
Zephyr RTOS overview

Vendor neutral open-source real time operating system

- Joint efforts of the vibrant community
- Vendor-friendly Apache 2.0 license
- LTS branches for product development
- For ARC EM, ARC HS 32- & 64-bit processors
- On all ARC development boards & simulators



Zephyr RTOS architecture

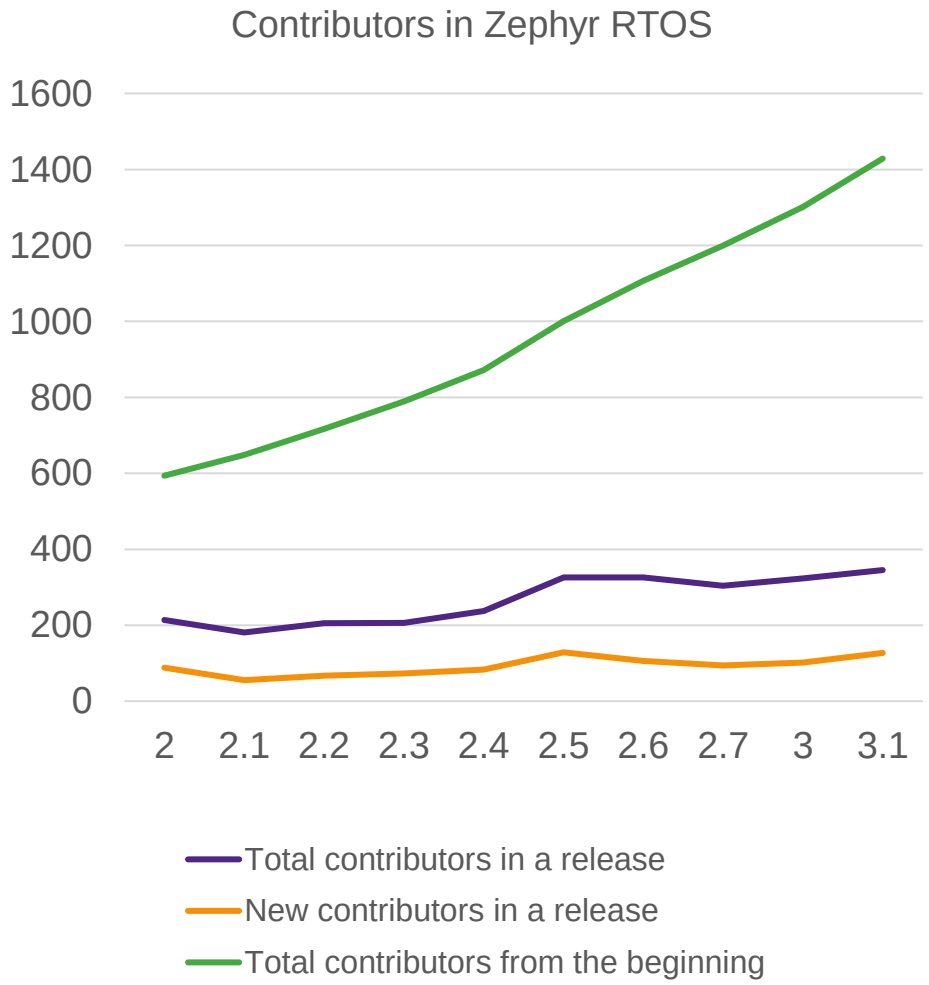


- Highly configurable, highly modular
- Cooperative and Preemptive Threading
- Memory and resources are typically statically allocated
- Integrated device driver interface
- Memory Protection: Stack overflow protection, Kernel object and device driver permission tracking, Thread isolation
- Bluetooth® Low Energy (BLE 5.1) with both controller and host, BLE Mesh
- 802.15.4 OpenThread
- Native, fully featured and optimized networking stack

Fully featured OS allows developers to focus on the application

Zephyr RTOS development

Community & major milestones



Feb 2016	• First public release v1.0.0
Apr 2019	• First Long-Term Support (LTS) release v1.14.0
Oct 2021	• Second LTS release v2.7.0
Jun 2022	• Latest release v3.1.0
Fall 2023	• Third LTS release planned

ARC processors in Zephyr RTOS

Support of ARC processors in Zephyr RTOS cont'd

From v2.0.0 onwards

v2.0.0	• ARC HS processors family • SMP (Simultaneous Multi-Processing) for ARC HS processors • ARC EM SDP & ARC HS Development Kit
v2.1.0	• "Direct IRQ" • Floating-point for selected tasks
v2.4.0	• MetaWare toolchain support • QEMU for ARCv2 processors: ARC EM & ARC HS • SMP enabled by default on quad-core ARC HS Development Kit
v2.5.0	• SMP improvements for ARC: scheduler fixes & multi-core configurations for nSIM
v2.6.0	• ARCv3 ISA: single-core ARC HS6x with nSIM & QEMU • MPU on ARCv2 QEMU platforms
v2.7.0	• Multi-core (SMP) for ARC HS6x • MetaWare toolchain: C library & basic C++ • QEMU for ARC HS6x in Zephyr CI
v3.1.0	• Single- & multi-core (SMP) ARC HS5x with GNU and MetaWare toolchains

	ARC processor families			
	EM	**HS3x/4x**	**EV/VPX**	**HS5x/6x**
Port status	upstreamed	upstreamed	WIP	upstreamed
Features				
Closely coupled memories (ICCM, DCCM)	Y	Y	TBD	TBD
Execution with caches - Instruction/Data, L1/L2 caches	Y	Y	Y	Y
Hardware-assisted unaligned memory access	Y	Y	TBD	Y
Regular interrupts with multiple priority levels, direct interrupts	Y	Y	TBD	Y
Fast interrupts, separate register banks for fast interrupts	Y	Y	TBD	N
Hardware floating point unit (FPU)	Y	Y	N	TBD
Symmetric multiprocessing (SMP) support, switch-based	N/A	Y	TBD	Y
Hardware-assisted stack checking	Y	Y	TBD	N
Hardware-assisted atomic operations	N/A	Y	TBD	Y
DSP ISA	Y	N	TBD	TBD
DSP AGU/XY extensions	N	N	TBD	TBD
Userspace	Y	Y	N	TBD
Memory protection unit (MPU)	Y	Y	TBD	N
Memory management unit (MMU)	N/A	N	N/A	N
SecureShield	Y	N/A	N/A	N/A
Toolchains				
GNU (open source GCC-based)	Y	Y	N	Y
MetaWare (proprietary Clang-based)	Y	Y	Y	Y
Simulators				
QEMU (open source)	Y	Y	N	Y
nSIM (proprietary, provided by MetaWare Development Tools)	Y	Y	Y	Y

https://docs.zephyrproject.org/latest/hardware/arch/arc-support-status.html

Interesting software features

“Single-thread” or “no threading” mode

Use cases
- Bootloaders
- Sample application
- Simple event-driven applications

Size reduction
- Removal of the scheduler
- Removal of thread-related routines
- Reduction of start-up & timing-related code

OS services
- IRQ handling
- Device drivers
- Timers
- Tracing & logging
- Memory management

https://docs.zephyrproject.org/latest/kernel/services/threads/nothread.html

“Hello world” sample in “no multithreading” mode

```
#include <zephyr/zephyr.h>

void main(void)
{
        printk("Hello World!\n");
}
```

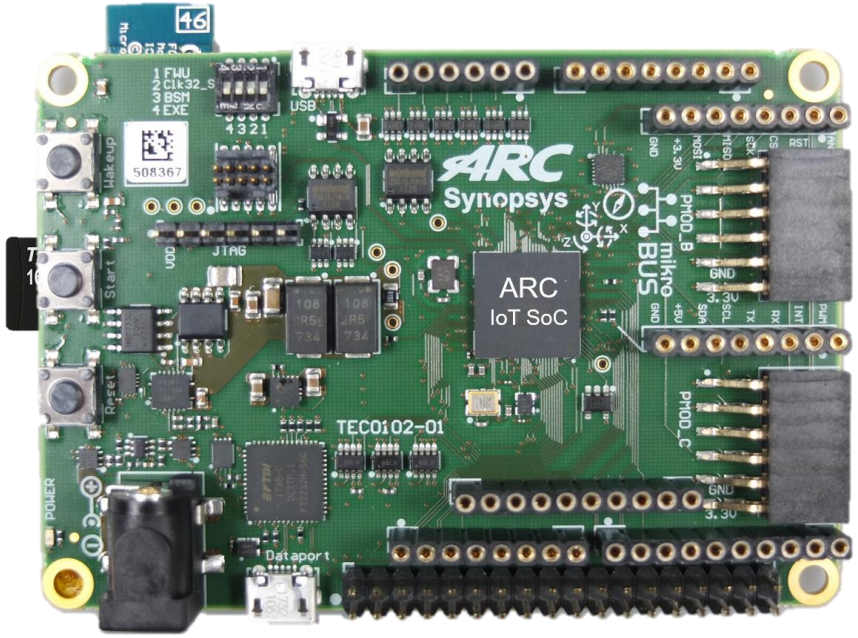
```
$ west build -b qemu_arc_hs
        samples/hello_world
        --
        -DCONFIG_MULTITHREADING=n


$ west build -t run

Hello World!
```

https://github.com/zephyrproject-rtos/zephyr/blob/main/samples/hello_world/src/main.c
https://docs.zephyrproject.org/latest/samples/hello_world/README.html

Single-threaded “Hello world” on IoTDK



Sample application	• “Hello world”
ROM	• 11.768 → 9.440 bytes
RAM	• 3.792 → 3.368 bytes
Footprint reduction	• 18%

Memory footprint of Zephyr RTOS on ARC nSIM platform

Depends on software and/or hardware features enabled in the build

Configuration	Path in source tree	Size, KiB	Comments
nsim_hs	samples/hello_world	**~8.7**	Single thread Zephyr (multithreading disabled, CONFIG_MULTITHREADING=n)
nsim_hs	samples/synchronization	**~15.8**	Multithreading Zephyr, several threads, no userspace
nsim_hs_smp	samples/synchronization	**~23.4**	SMP Zephyr, 2 cores, several threads, no userspace

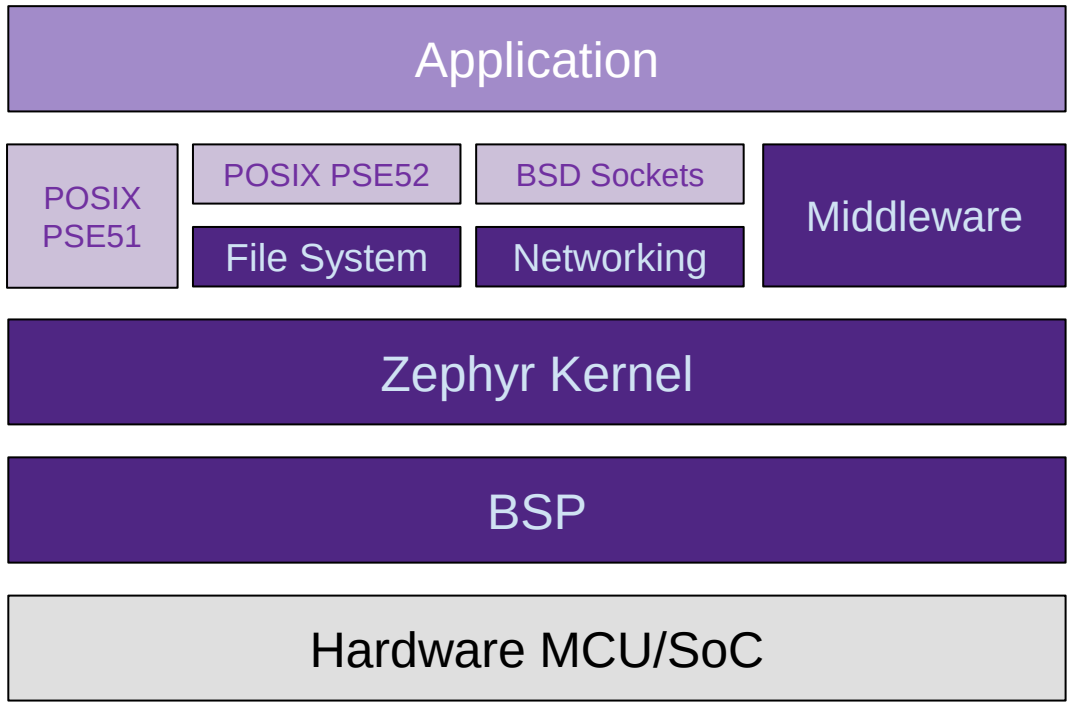
- Exception debug disabled
  - CONFIG_ARC_EXCEPTION_DEBUG=n
  - CONFIG_FAULT_DUMP=0

POSIX API on Zephyr

Provides familiar API to non-embedded programmers, especially to Linux developers

Enable re-use (portability) of existing libraries based on POSIX APIs

- Provides efficient subset appropriate for small (MCU) embedded systems
- POSIX API subset is increasingly popular operating system abstraction layer (OSAL) for IoT
- Supports subsets of PSE51, PSE52, and BSD sockets API



https://docs.zephyrproject.org/latest/guides/portability/posix.html

HTTP web-server example: Civet Web

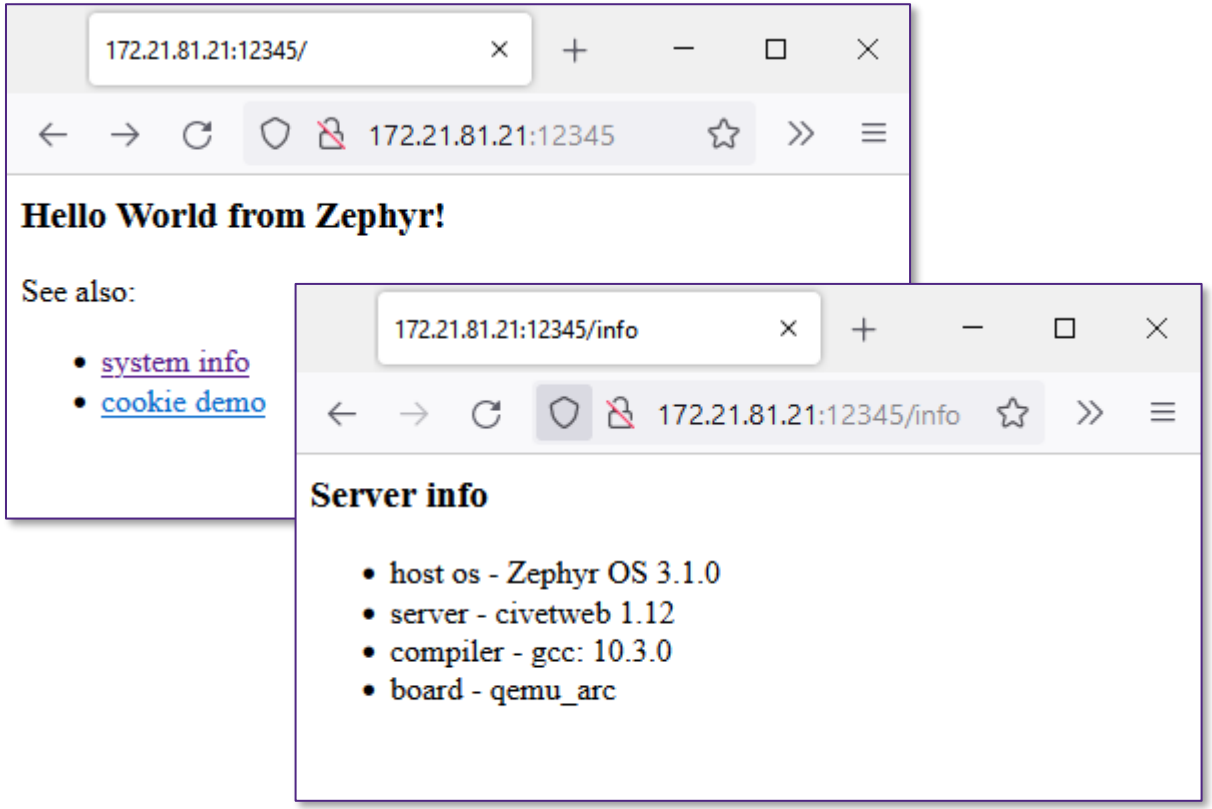

```
$ west build -b qemu_arc_hs
        samples/net/civetweb/http_server



$ west build -t run


*** Booting Zephyr OS build zephyr-v3.1.0 ***
<inf> net_config: Initializing network
<inf> net_config: IPv4 address: 192.0.2.1


<dbg> mg_cry_internal_impl: log_access: 192.0.2.2 - "GET /?null
HTTP/1.0" 200
```



Civet Web: Complex application on top of Zephyr RTOS

Services and API's available in Zephyr RTOS make it possible

Written mostly on C and uses:

- POSIX API
  - Pthreads
  - POSIX Message Queues
  - BSD Sockets
- DNS client
- IPv4
- TCP
- Ethernet device drivers

- Independent project developed on GitHub: https://github.com/civetweb/civetweb
- Very minimal changes for Zephyr support, see commit 67cdc
- Removed from Zephyr RTOS in v3.2.0 due to missing maintainer

EEMBC CoreMark Pro

Primer for easy porting of POSIX compliant application to Zephyr RTOS

Input data

- Official sources: https://github.com/eembc/coremark-pro
- Uses EEMBC's Multi-Instance Test Harness (MITH)
- MITH abstraction layer is configured to work with the POSIX pthread

Port to Zephyr

- Integrate Zephyr build system
  - CMakeLists.txt
  - Kconfig
  - prj.conf
- Pre-populate **argc** & ***argv[]**

CoreMark Pro: Integrate Zephyr build system

CMakeLists.txt

```
cmake_minimum_required(VERSION 3.20.0)

find_package(Zephyr REQUIRED HINTS $ENV{ZEPHYR_BASE})
project(coremark)

if(CONFIG_64BIT)
  target_compile_options(app PRIVATE "-D EE_SIZEOF_LONG=8")
  target_compile_options(app PRIVATE "-D EE_SIZEOF_PTR=8")
  target_compile_options(app PRIVATE "-D EE_PTR_ALIGN=8")
endif()

# MITH headers
include_directories(mith/include)
include_directories(mith/al/include)

# MITH sorces
FILE(GLOB mith_sources mith/al/src/*.c mith/src/*.c)
target_sources(app PRIVATE ${mith_sources})
```

```
# Input arguments
target_sources(app PRIVATE main-args.c)

# CoreMark Pro sources
if(CONFIG_CMP_COREMARK)
  FILE(GLOB app_sources benchmarks/core/*.c)
  target_sources(app PRIVATE ${app_sources})
  target_sources(app PRIVATE workloads/core/core.c)
endif()

set_ifndef(W_WORKERS_NUM 4)
target_compile_options(app PRIVATE "-D W_WORKERS_NUM=${W_WORKERS_NUM}")

set_ifndef(C_CONTEXT_NUM 4)
target_compile_options(app PRIVATE "-D C_CONTEXT_NUM=${C_CONTEXT_NUM}")

set_ifndef(I_ITERATION_NUM 1)
target_compile_options(app PRIVATE "-D I_ITERATION_NUM=${I_ITERATION_NUM}")
```

CoreMark Pro: Integrate Zephyr build system

Kconfig & prj.conf

 # Kconfig

```
mainmenu "CoreMark Pro benchmark"

choice
        prompt "CoreMark Pro benchmark"
        default CMP_COREMARK

config CMP_COREMARK
        bool "coremark"
        help
          core benchmark

endchoice

source "Kconfig.zephyr"
```

 # prj.conf

```
CONFIG_POSIX_API=y
CONFIG_PTHREAD_IPC=y
CONFIG_NEWLIB_LIBC=y
CONFIG_NEWLIB_LIBC_FLOAT_PRINTF=y
CONFIG_MAX_PTHREAD_COUNT=40
CONFIG_AUTO_PTHREAD_ATTRS=y
CONFIG_POSIX_MQUEUE=y
CONFIG_THREAD_NAME=y
CONFIG_INIT_STACKS=y
CONFIG_SPEED_OPTIMIZATIONS=y
```

CoreMark Pro: Pre-populate **argc** & ***argv[]**

Embedded application doesn't accept command line arguments, work it around

// main-args.c

```
#define _g_xstr(a) _g_str(a)
#define _g_str(a) #a

int argc = 5;
char *argv[] = {
                "core.exe",
                "-v0",
                "-w" _g_xstr(W_WORKERS_NUM),
                "-c" _g_xstr(C_CONTEXT_NUM),
                "-i" _g_xstr(I_ITERATION_NUM)
};
```

// workloads/core/core.c

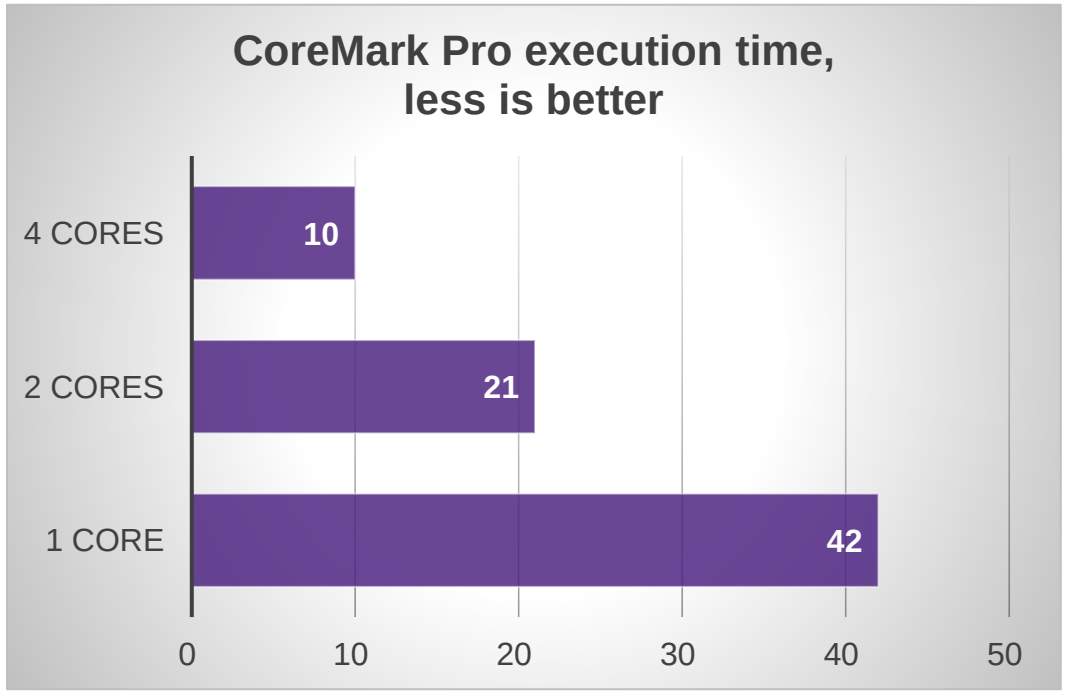
```
#ifdef __ZEPHYR__
extern int argc;
extern char *argv[];

void main(void)
#else
int main(int argc, char *argv[])
#endif
{
        char name[MITH_MAX_NAME];
        char dataname_buf[MITH_MAX_NAME];

...
```

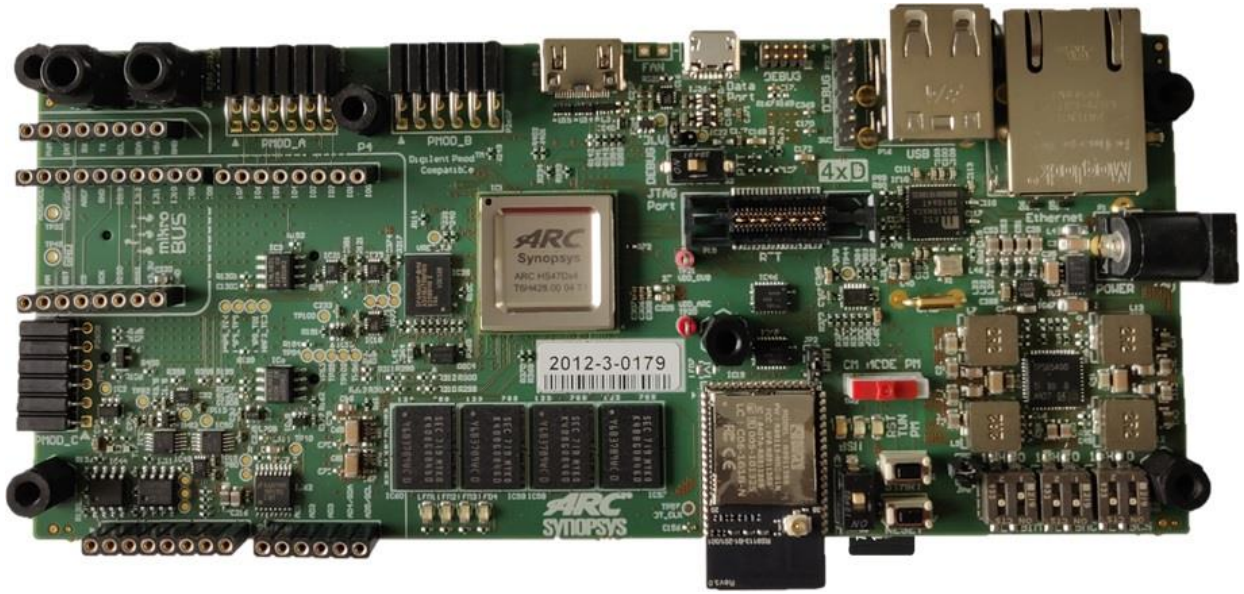
EEMBC CoreMark Pro on quad-core ARC HS48D

Linear scaling on a multi-core system



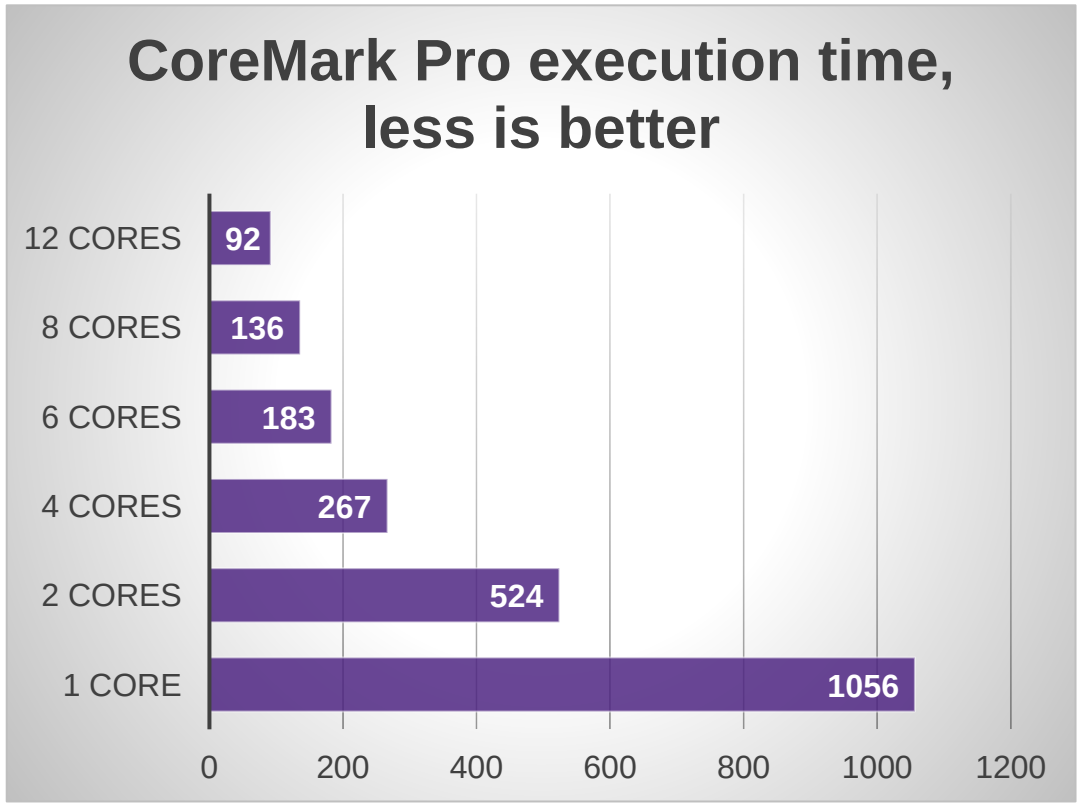
ARC HS4xD Development Kit

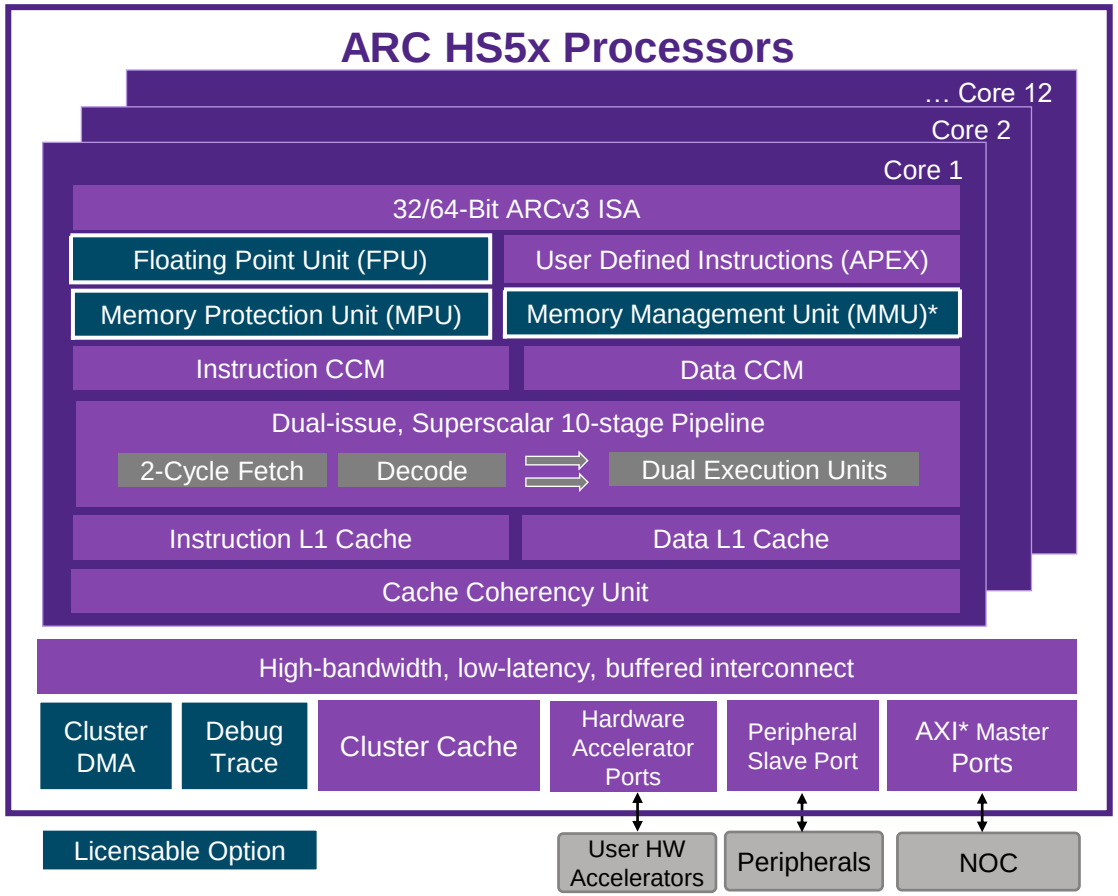
- Quad-core dual-issue ARC HS48D
- Core running at 1000 MHz



EEMBC CoreMark Pro on 12-core ARC HS58

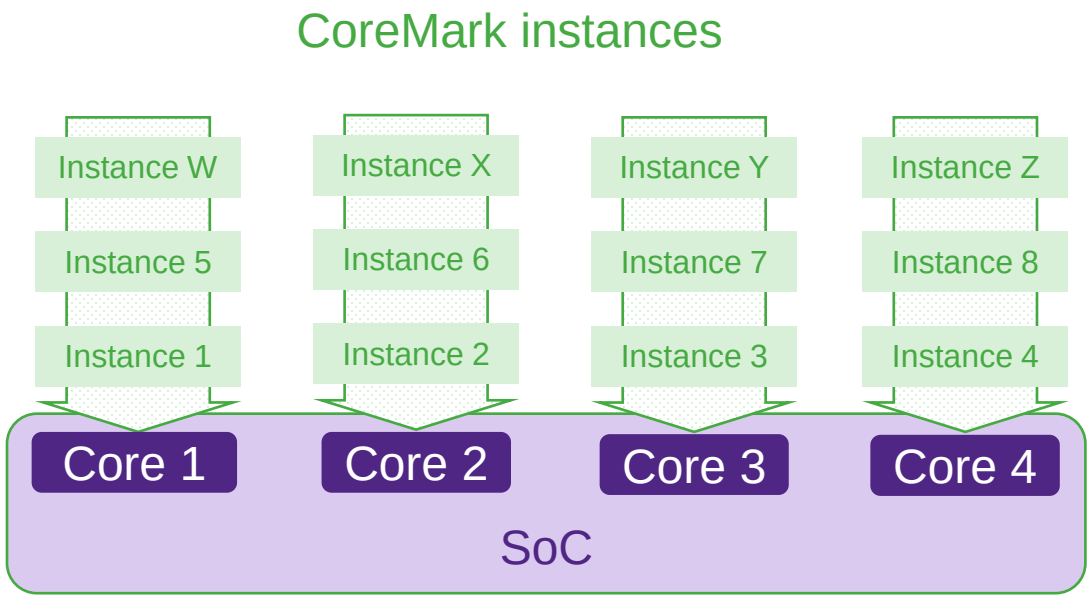
Linear scaling on a multi-core system





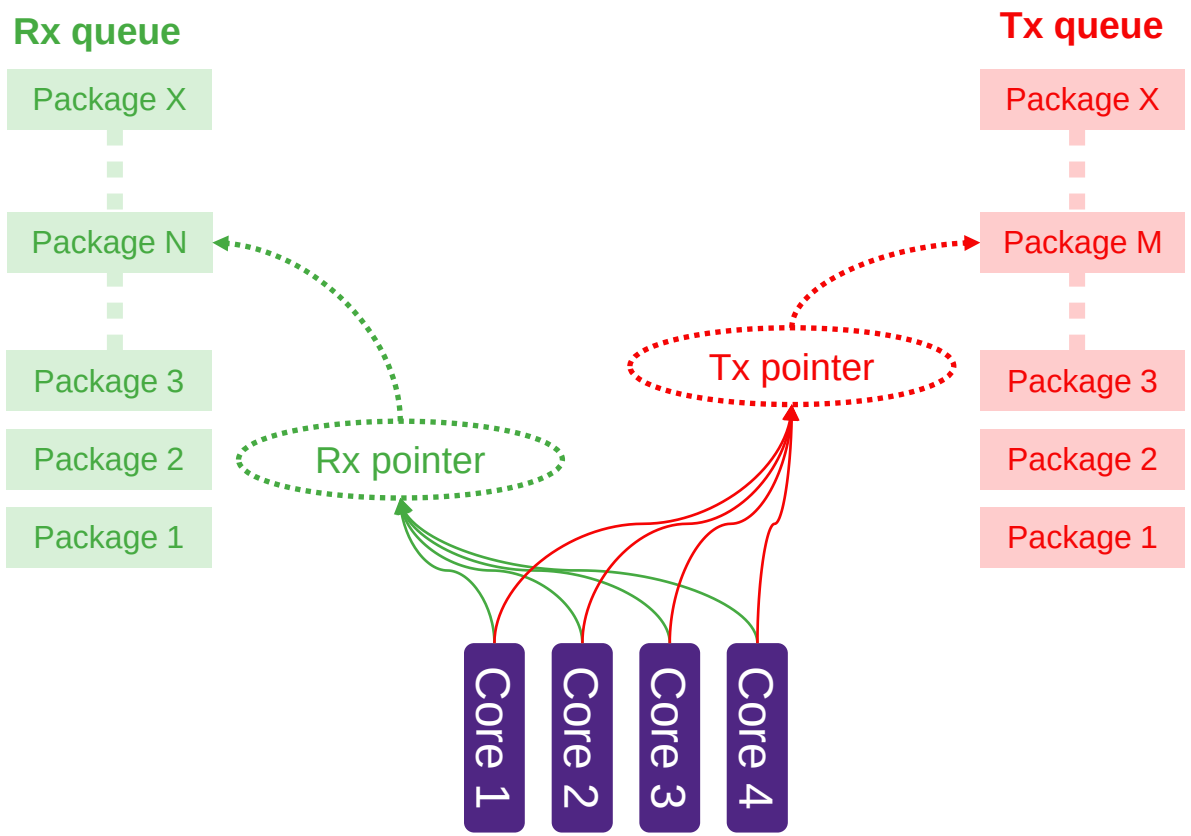
CoreMark Pro: Scales linearly

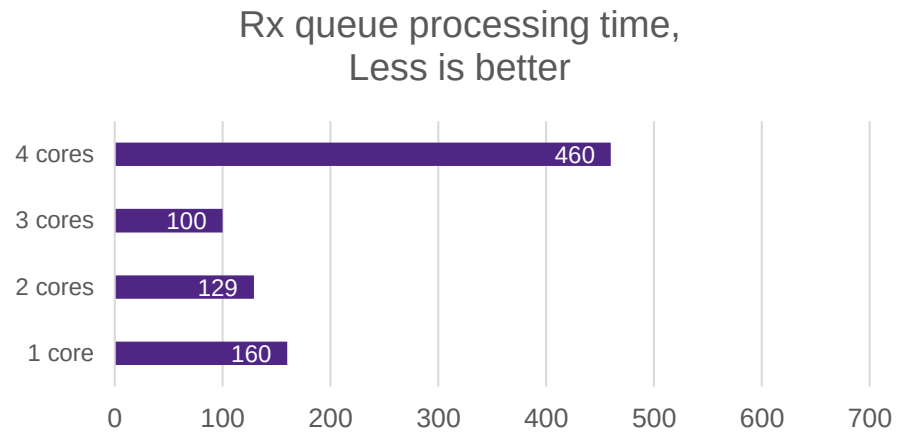
- CoreMark is inherently single-core benchmark
  - Linked-list manipulations
  - Simple state machine
  - Matrix manipulation
- No code or data dependencies
- Multiple instances perfectly run on all available cores or processors

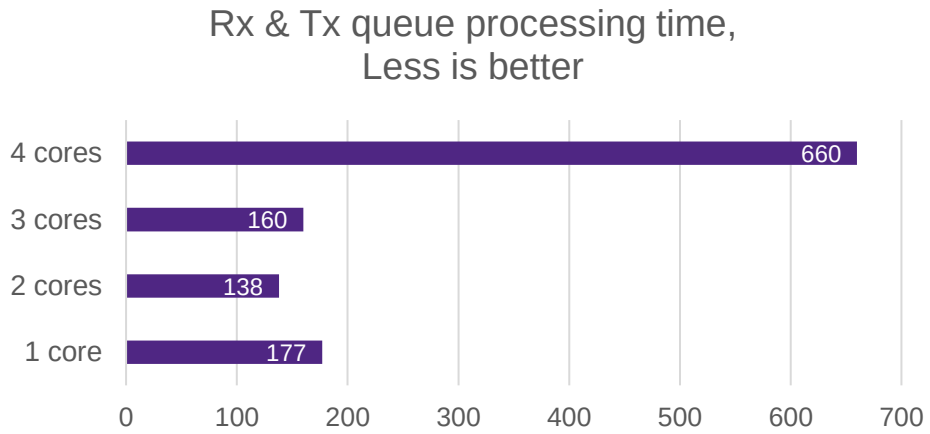


pktqueue: Counter-example

Wasting time on synchronization







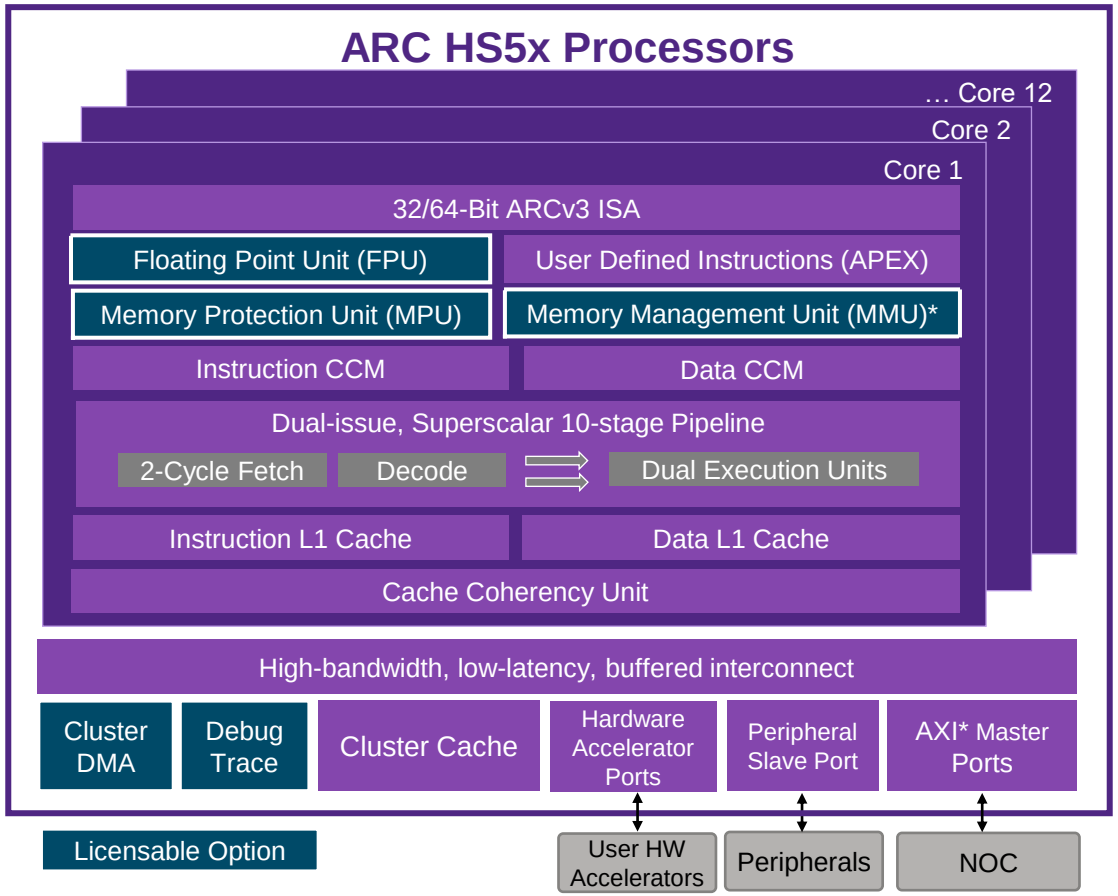
https://docs.zephyrproject.org/latest/samples/arch/smp/pktqueue/README.html

New hardware features

ARCv3 ISA processor support

Single-core and multi-core ARC HS5x & HS6x are now supported

- ARC HS6x support added in Zephyr v2.6.0
- ARC HS5x support added in Zephyr v3.1.0
- Single-core and multi-core configurations (up-to all 12 cores*) are supported
- Support building with both GNU & MetaWare toolchains

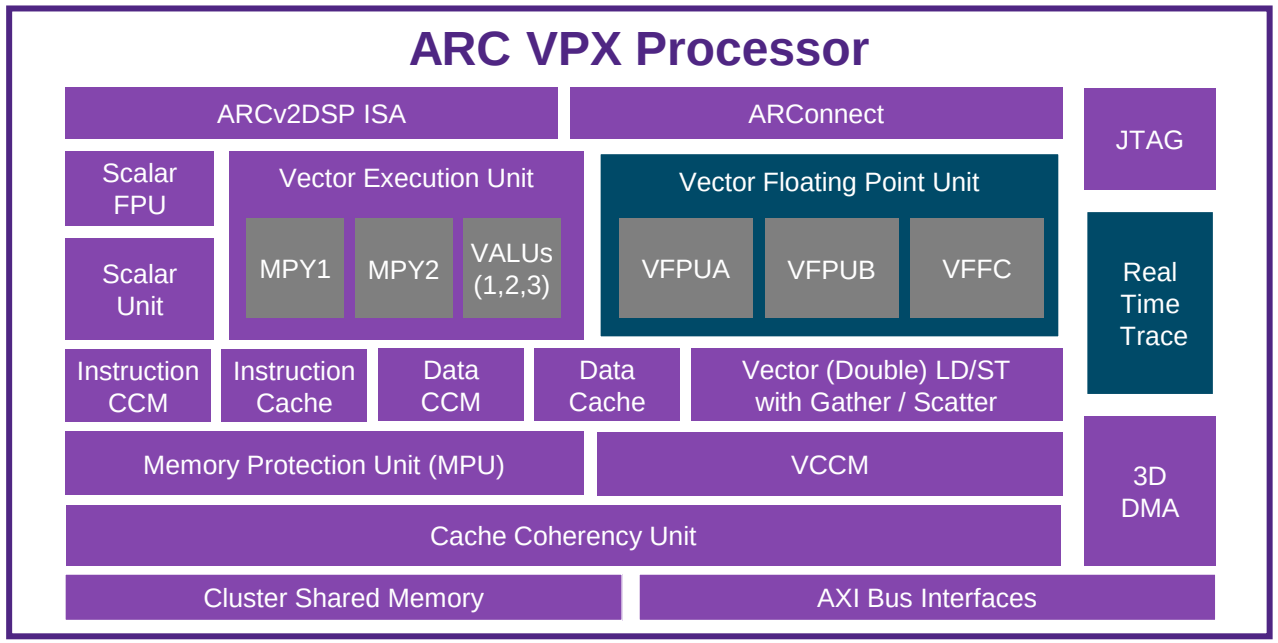


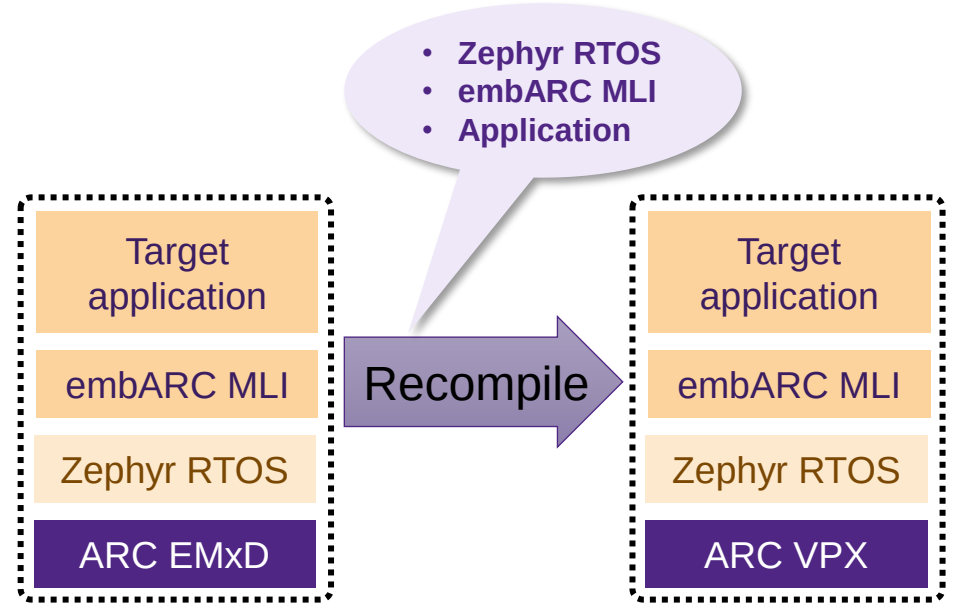
* 12-core SMP support is available in Synopsys Zephyr RTOS fork, upstreaming activities are ongoing

ARC VPX support

Zephyr port for ARC VPX in progress

- Scalar part of ARC VPX is close to ARCv2 ARC HS
- Smooth firmware transition from ARC EM to ARC VPX
- Uses MetaWare tools





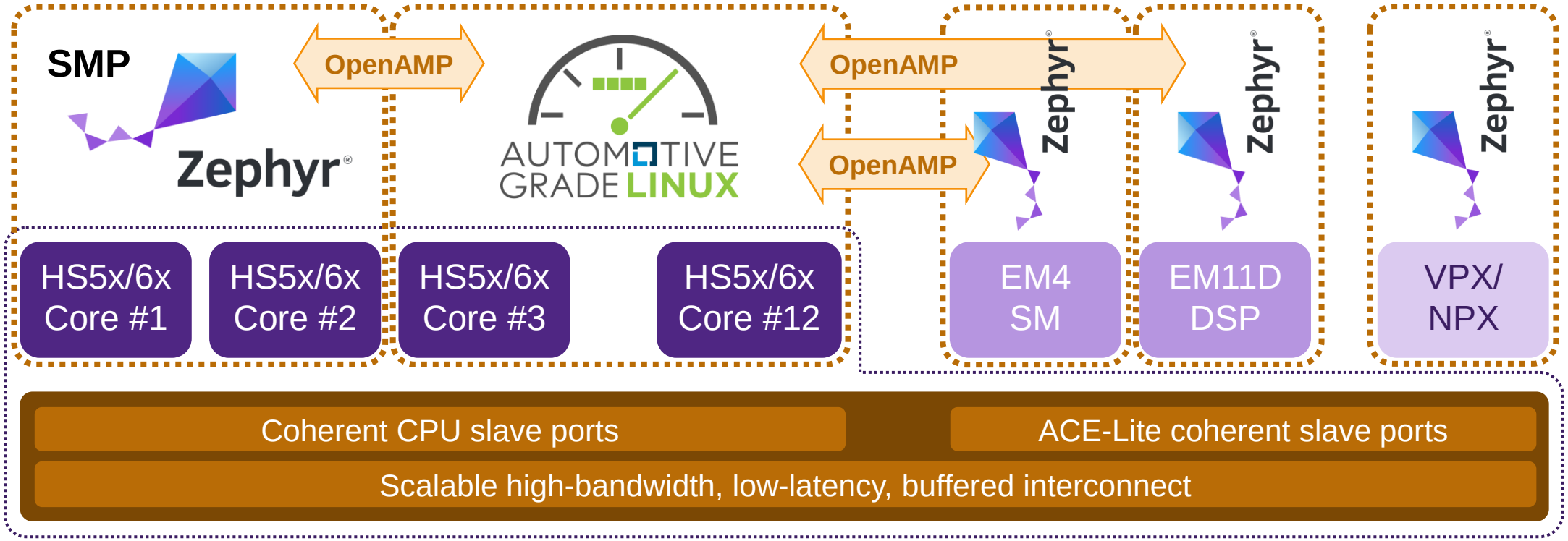
- Combined control & DSP functions
- Zephyr RTOS in multithreading mode

- More powerful only DSP functions
- Zephyr RTOS in no threading mode

Heterogeneous cluster

Zephyr RTOS as an abstraction layer

- Support of full range of up-to-date ARC processor families
- Uniform API to OS services



Conclusions

Zephyr RTOS for ARC processors is mature and powerful tool

- Zephyr RTOS supports
  - Most widely used families of ARC processors:
    - ARCv2 ISA processors: ARC EM & ARC HS3x/4x
    - ARCv3 ISA processors: 32-bit ARC HS5x & 64-bit ARC HS6x
  - Major features of ARC processors
    - MPU, SMP, DSP, FPU etc
- Software features of Zephyr RTOS:
  - Efficiently utilize hardware features of ARC processors
  - Help with development of complex real-life applications

SYNOPSYS®
Silicon to Software™

Thank You