Zephyr® Project

Developer Summit

Zephyr® Project
Developer Summit

Arm® core + DSP Co-simulation

With customized QEMU in Zephyr

Hake Huang, NXP Semiconductors
Hake.huang@nxp.com

Speakers

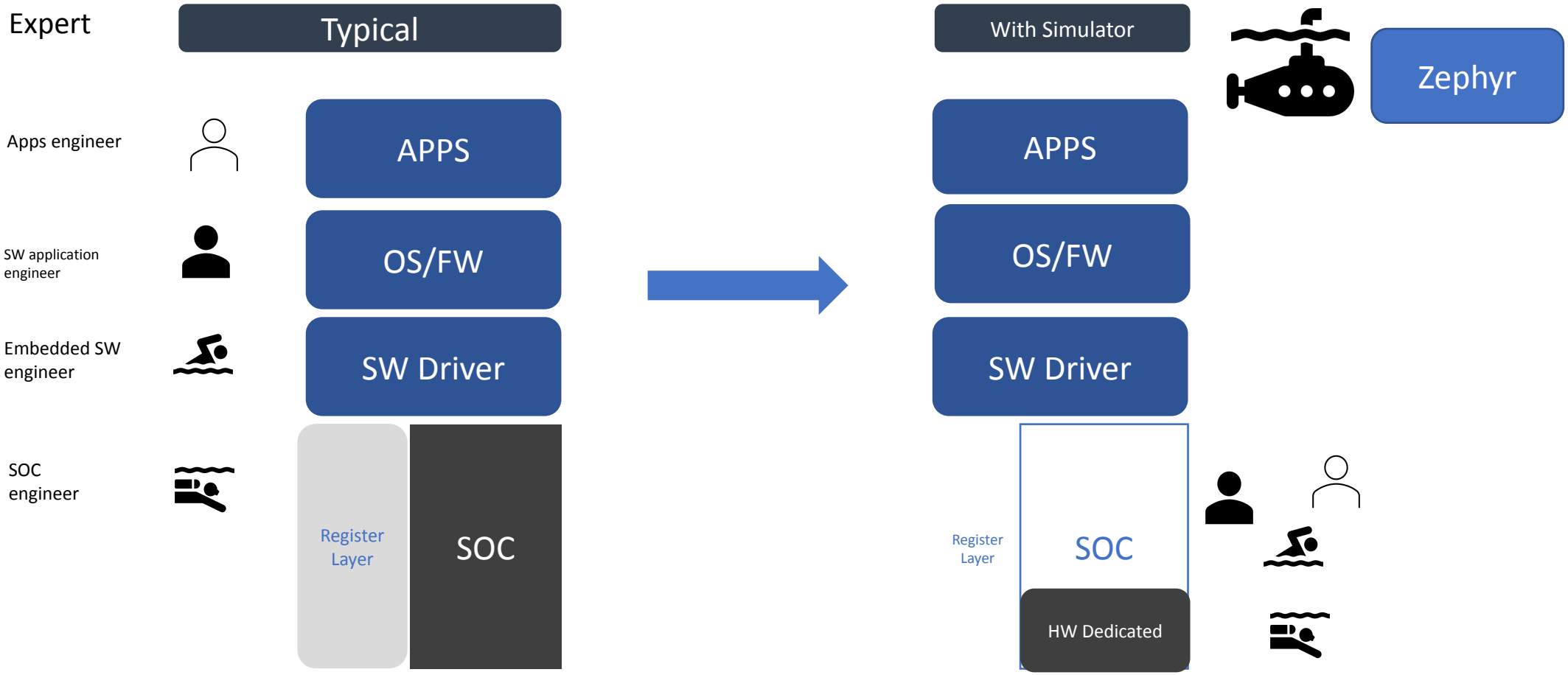
Hake Huang
Software Engineer, NXP Semiconductor

Software Engineer from NXP.member of Zephyr Testing Working group.Coordinator of Zephyr-RTOS project

- How a simulator can benefit
- A brief intro to QEMU
- SOC co-simulator design pattern
- i.MX RT595 QEMU co-simulator

- How a simulator can benefit

Simulator Benefits



- A brief intro to QEMU

- QEMU is a machine emulator: it can run an unmodified *target* operating system (such as Windows® or Linux®) and all its applications in a virtual machine
- QEMU is made of several subsystems:
  - CPU emulator (currently x86, PowerPC, ARM, Sparc, RISC-V, XTENSA etc.,)
  - Emulated devices (e.g. VGA display, 16450 serial port, PS/2 mouse and keyboard, IDE hard disk NE2000 network card, ...)
  - Generic devices (e.g. block devices, character devices, network devices) used to connect the emulated
  - devices to the corresponding host devices
  - Machine descriptions (e.g. PC, PowerMac, Sun4m) instantiating the emulated devices
  - Debugger
  - User interface

Consider the case where we must translate the following PowerPC instruction to x86 code:

```
addi r1,r1,-16    # r1 = r1 - 16
```

Guest Instructions

The following micro operations are generated by the PowerPC code translator:

```
movl_T0_r1          # T0 = r1
addl_T0_im -16      # T0 = T0 - 16
movl_r1_T0          # r1 = T0
```

When the code generator is run, the following host code is output:

```
# movl_T0_r1
# ebx = env->regs[1]
mov    0x4(%ebp),%ebx
# addl_T0_im -16
# ebx = ebx - 16
add    $0xfffffff0,%ebx
# movl_r1_T0
# env->regs[1] = ebx
mov    %ebx,0x4(%ebp)
```

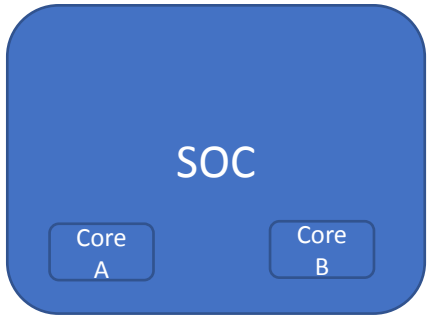
Host Instructions

Picture source Ref1

Key Concepts for Full-system emulation

- Object file is parsed to get its symbol table, its relocations entries and its code section.
- The micro operations are located in the code section using the symbol table.
- The relocations of each micro operations are examined to get the number of constant parameters.
- A memory copy in C is generated to copy the micro operation code..
- For some hosts such as ARM, constants must be stored near the generated code because they are accessed with PC relative loads with a small displacement.

- Co-simulator design



?

inter-core inter-operation?

Core A process

Core B process

1. A real time IPC
2. Easy to expand to complex scenario

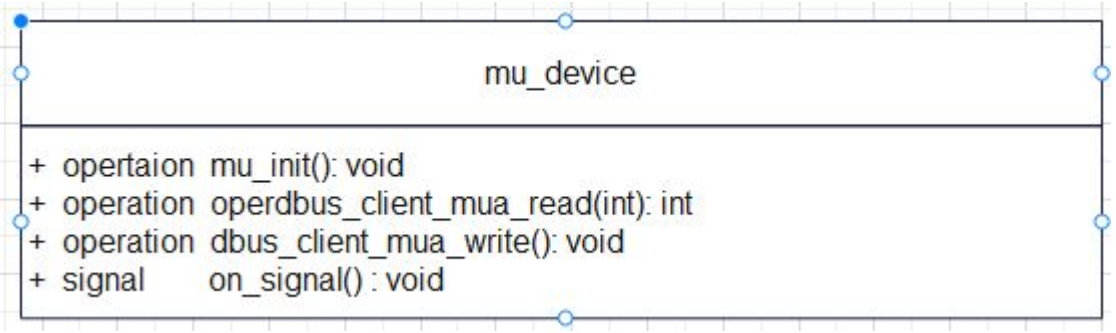
D-Bus https://www.freedesktop.org/wiki/Software/dbus/

- is a message bus system, a simple way for applications to talk to one another.
- helps coordinate process lifecycle
- heavily tested in the real world over several years

```
result = g_dbus_proxy_call_sync(mu->proxy, "Read",
                                g_variant_new("(t)", offset),
                                G_DBUS_CALL_FLAGS_NO_AUTO_START,
                                -1, NULL, &err);
```

```
@dbus.service.method("org.qemu.debus.mua",
                     in_signature='t', out_signature='u')
def Read(self, offset):
```

A QEMU process whose read/write to device address will translate to dbus operations in python script running as dbus-daemon process



4 routines to enable a QEMU dbus device (init, interrupt, write and read)

```c
static void dbus_client_mua_init(Object *obj)
{
    ......
    /*find the TYPE_DBUS_OBJ object*/
    s->dbus.dbus_conn = g_bus_get_sync(G_BUS_TYPE_SESSION, NULL, &err);
    s->dbus.proxy = g_dbus_proxy_new_sync(s->dbus.dbus_conn,
                    G_DBUS_PROXY_FLAGS_NONE,
                    NULL,      /* GDBusInterfaceInfo */
                    "org.qemu.client",       /* name */
                    "/org/qemu/client",      /* object path */
                    "org.qemu.client.mua", /* interface */
                    NULL,      /* GCancellable */
                    &err);

        g_signal_connect(s->dbus.proxy,
            "g-signal",
            G_CALLBACK (on_signal),
            s);
    ......
}
```

```c
static void
on_signal (GDBusProxy *proxy,
           gchar      *sender_name,
           gchar      *signal_name,
           GVariant   *parameters,
           gpointer    user_data)
{
      .......

      qemu_irq_raise(s->irq);

}
```

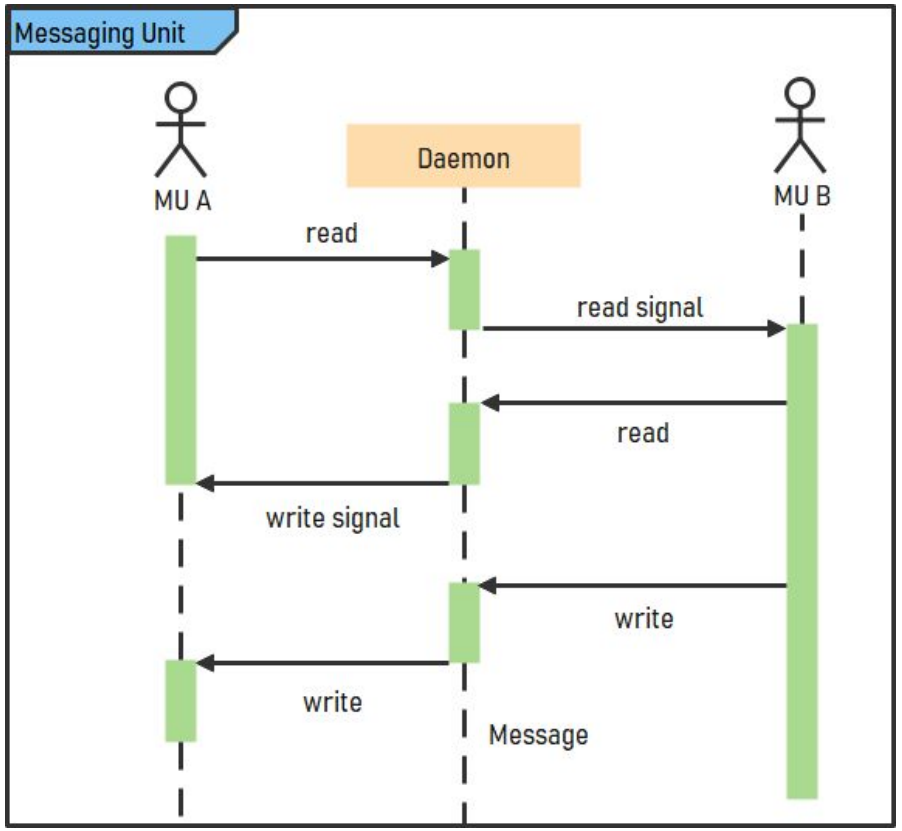
```c
static void dbus_client_mua_write(void *opaque, hwaddr offset,
        uint64_t value, unsigned size)
{
    ......
        /* write to dbus */
        result = g_dbus_proxy_call_sync(s->dbus.proxy, "MUAWrite",
                                        g_variant_new("(tt)",
                                        offset, value),
                                        G_DBUS_CALL_FLAGS_NO_AUTO_START,
                                        -1, NULL, &err);

}
```

```c
static uint64_t dbus_client_mua_read(void *opaque, hwaddr offset,
                                     unsigned size)
{
    ......
        result = g_dbus_proxy_call_sync(s->dbus.proxy, "MUARead",
                                        g_variant_new("(t)", offset),
                                        G_DBUS_CALL_FLAGS_NO_AUTO_START,
                                        -1, NULL, &err);


        g_variant_get(result, "(u)", &ret);

    return ret;
}
```



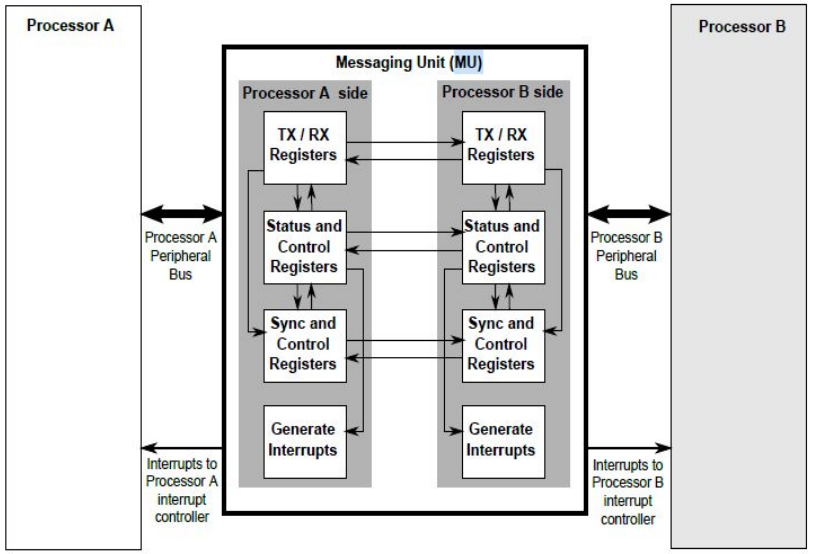
A Python® dbus daemon is written here for easy transaction implement

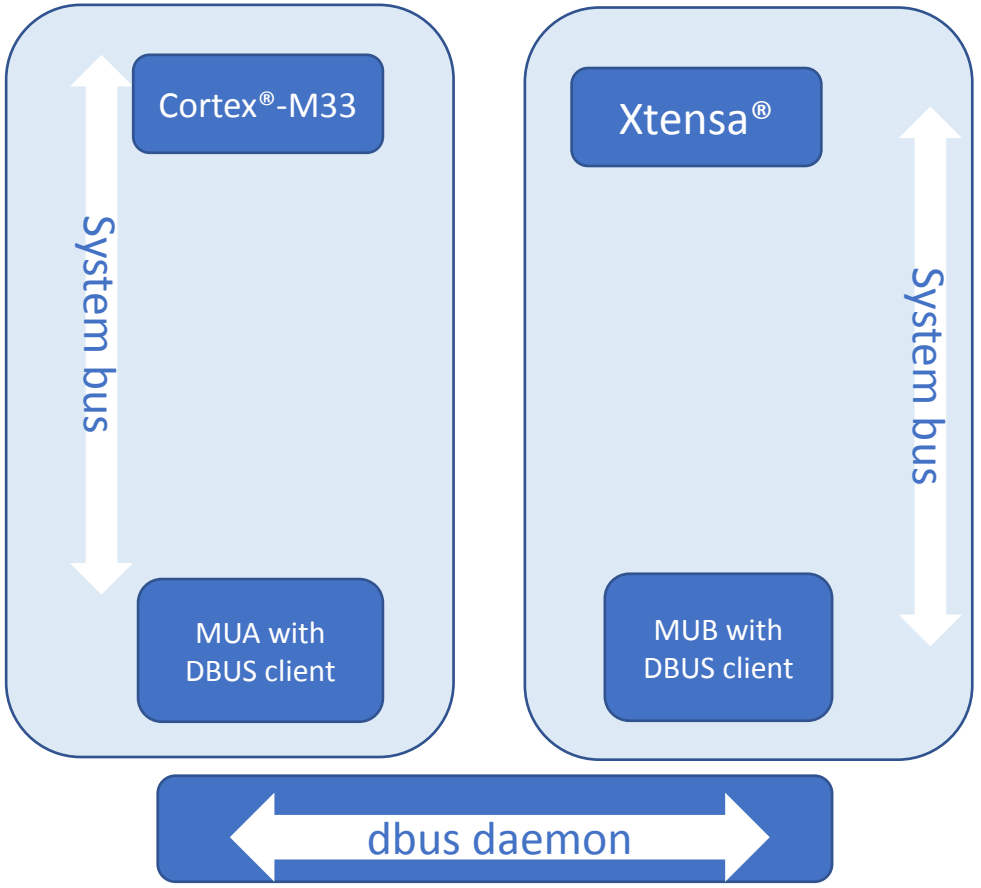
We call it dbus_mu_service.py

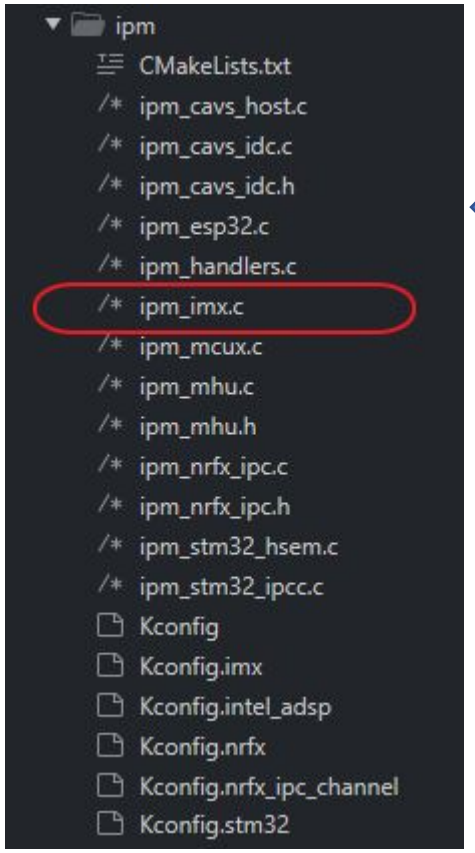
```
import dbus
import dbus.service
import dbus.mainloop.glib
```

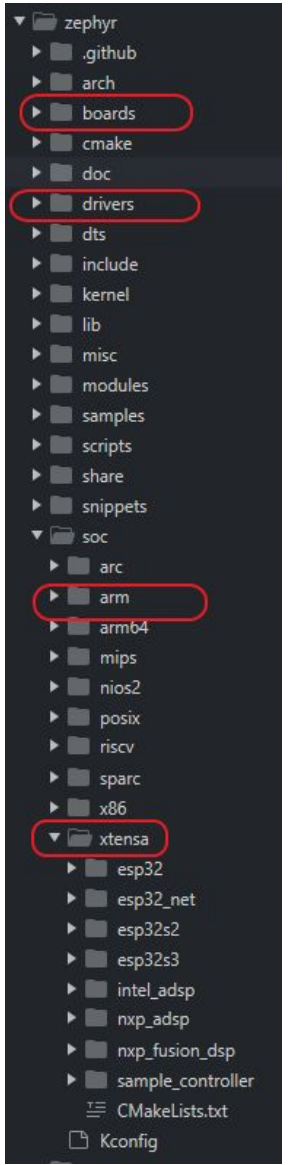
https://dbus.freedesktop.org/doc/dbus-python/

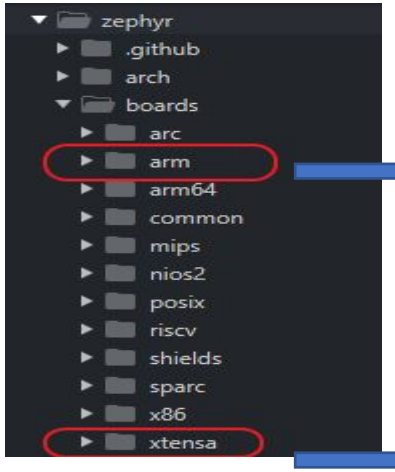
Real Hardware

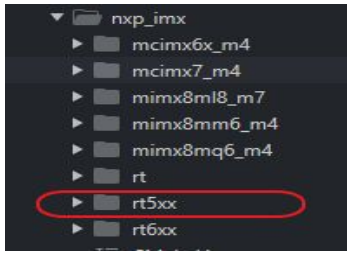


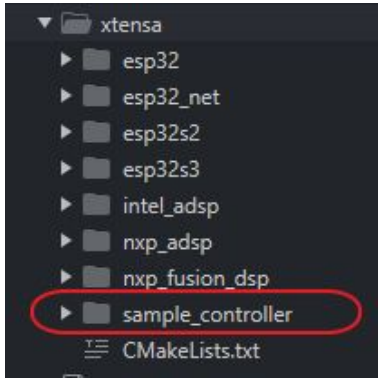


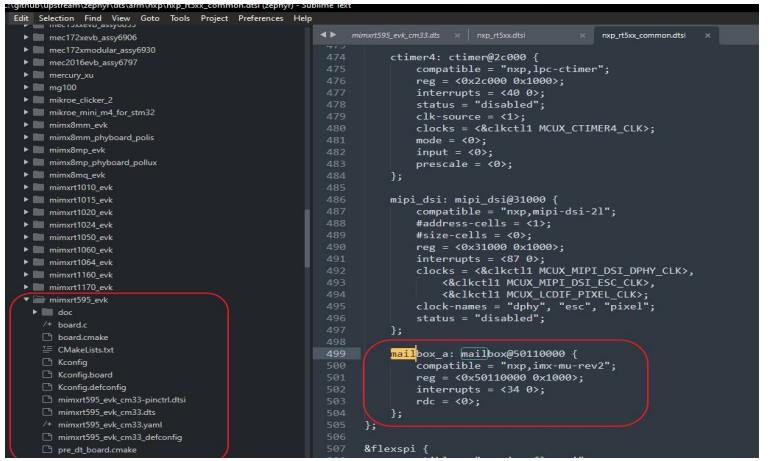


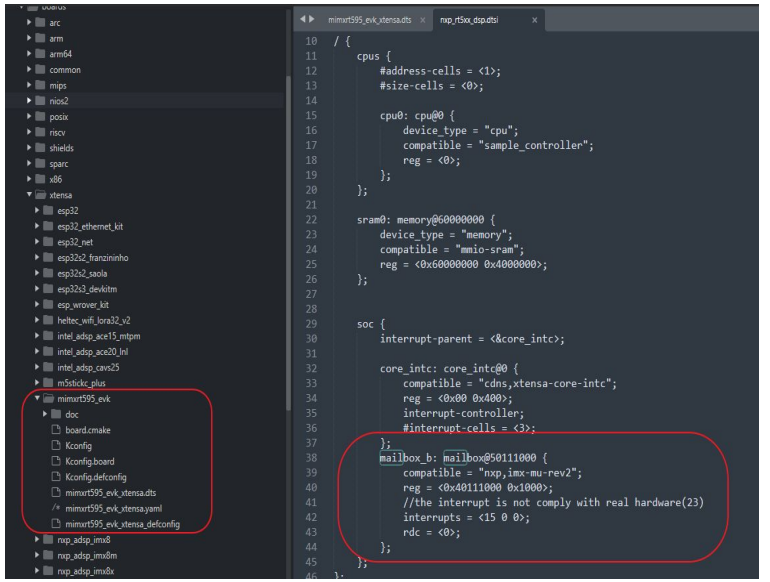






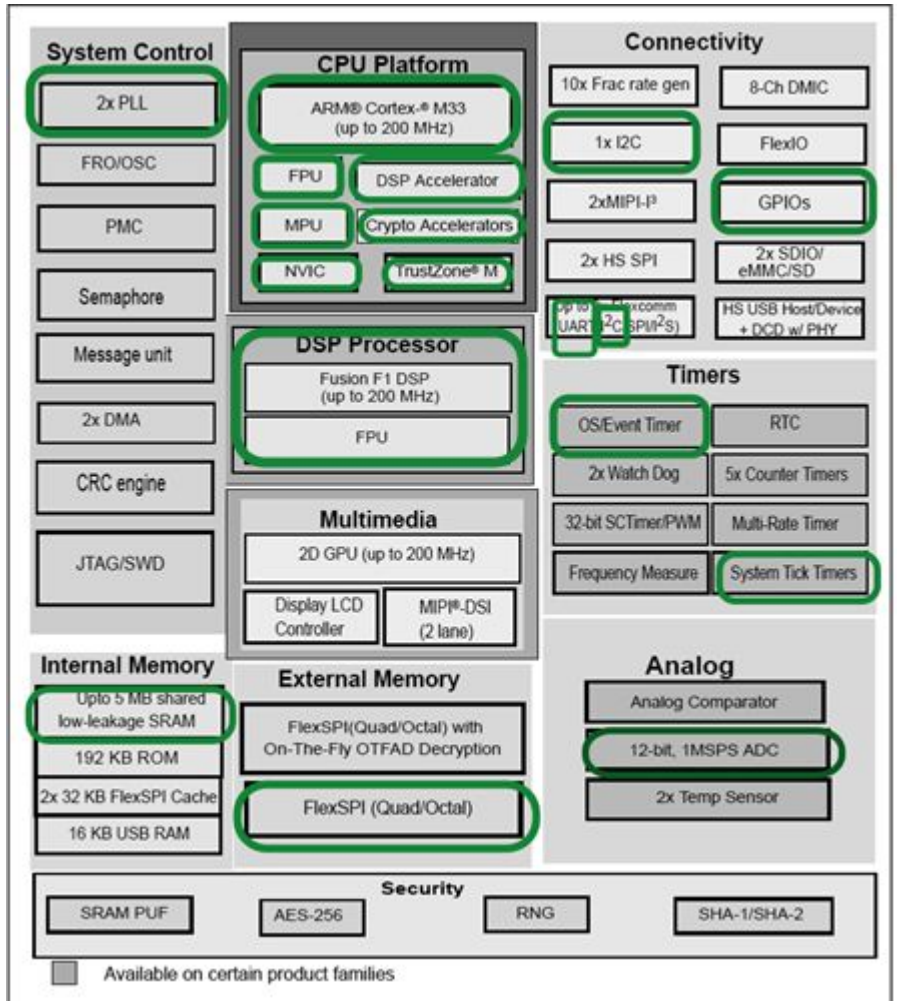




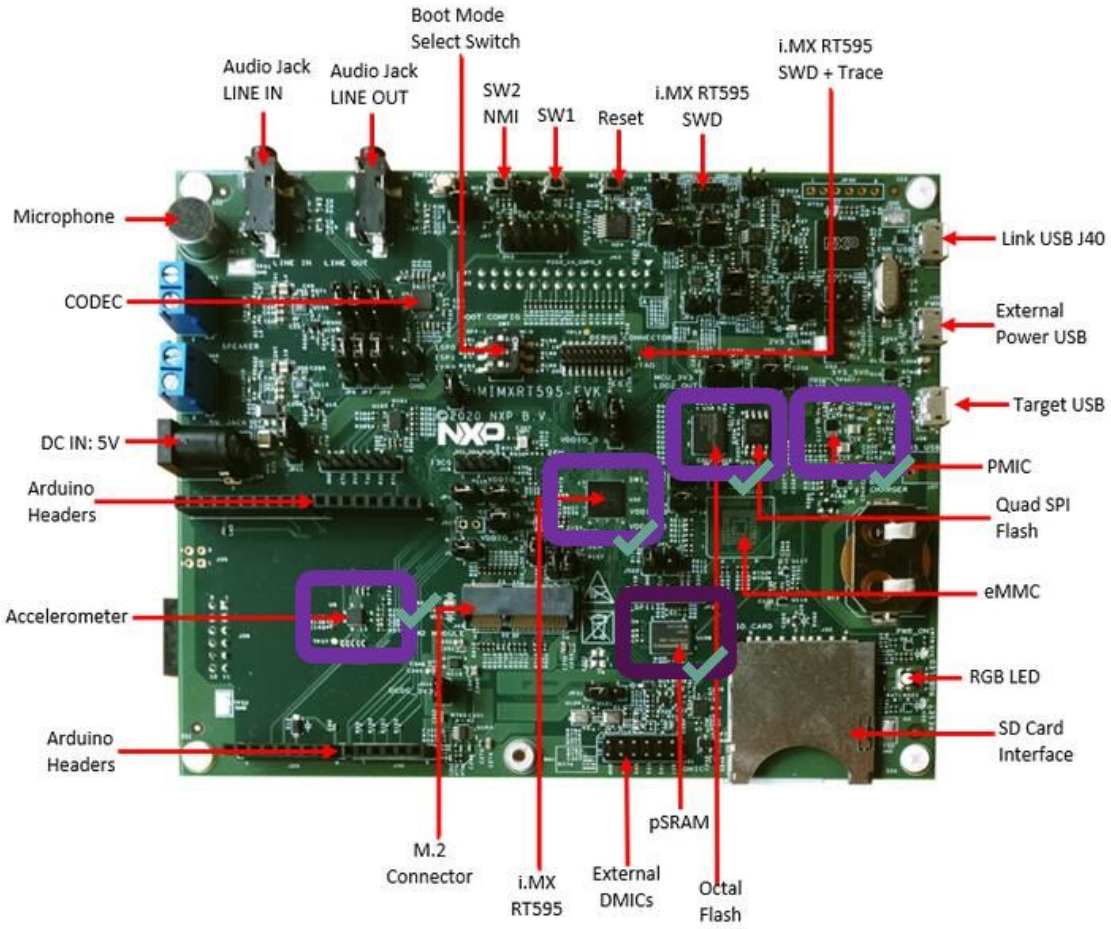


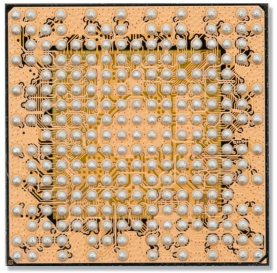
Zephyr-RT
OS Magics

- i.MX RT595 QEMU Co-simulator with DSP



- ✔ Share the same binary with real i.MX RT595 board on Zephyr
- ✔ Re-use Zephyr framework for DSP development
- ✔ Inter-operation from i.MX RT595 Cortex-M33 core to Fusion® DSP

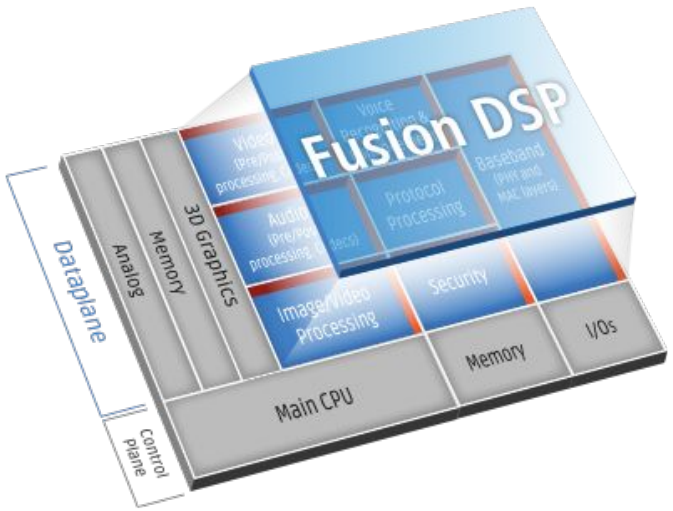




EDGELOCK ASSURANCE

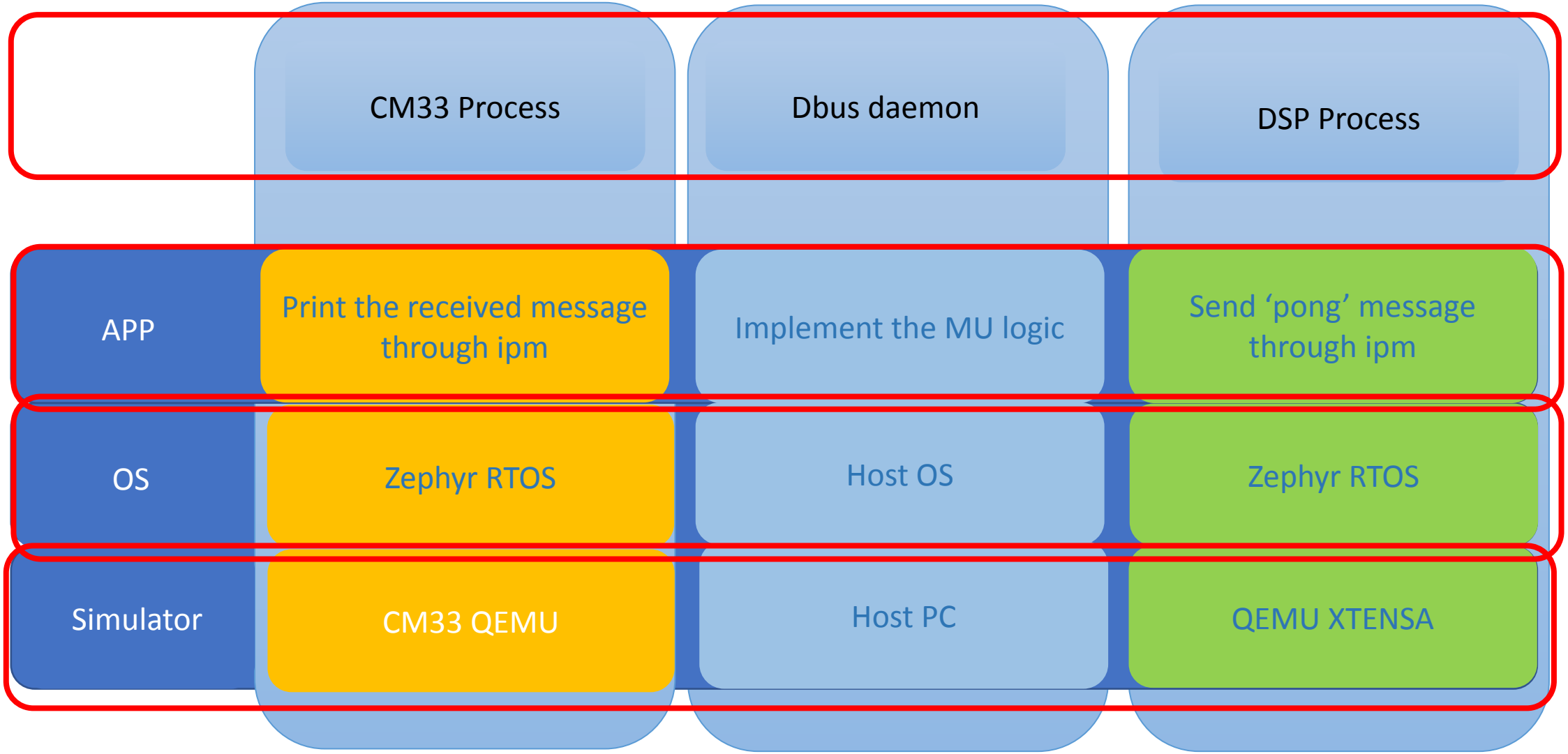
- $ git clone git://git.qemu.org/qemu.git
- $./configure --target-list=arm-softmmu
- $ make
- **$./build/**qemu-system-arm -nographic -machine rt595-m33,boot-base-addr=0x18001000 -kernel /home/shared/temp/zephyr.elf

Tensilica is known for its customizable Xtensa microprocessor core, which is used in i.MX RT595



- $ **git clone git://git.qemu.org/qemu.git** $ **mkdir qemu-xtensa ; cd qemu-xtensa**
- $ **../qemu/configure --prefix=`pwd`/root --target-list=xtensa-softmmu,xtensaeb-softmmu**
- $ **make install**
- **$ ./qemu-system-xtensa -nographic -machine xt-rt595-nommu -semihosting -cpu sample_controller**

Note: the implementation of this part is only for demo purpose, thus the underlying DSP settings are reused from simple controller supported in Zephyr


CM33 Process
Dbus daemon
DSP Process
APP
Print the received message through ipm
Implement the MU logic
Send ‘pong’ message through ipm
OS
Zephyr RTOS
Host OS
Zephyr RTOS
Simulator
CM33 QEMU
Host PC
QEMU XTENSA

Cortex_m33

```
0000000040000000-0000000040000fff (prio 0, i/o): rt_rstctl0
    0000000040001000-0000000040001fff (prio 0, i/o): rt_clkctl0
    0000000040002000-0000000040002fff (prio 0, i/o): rt_sysctl0
    0000000040020000-0000000040020fff (prio 0, i/o): rt_rstctl1
    0000000040021000-0000000040021fff (prio 0, i/o): rt_clkctl1
    0000000040022000-0000000040022fff (prio 0, i/o): rt_sysctl1
    0000000040025000-0000000040025fff (prio 0, i/o): RT_PINT
    0000000040080000-0000000040080fff (prio 0, i/o): iotkit-secctl-ns-regs
    0000000040106000-0000000040106fff (prio 0, i/o): rt.flexcomm
    0000000040110000-0000000040110fff (prio 0, i/o): dbus_client_mua
    0000000040113000-0000000040113fff (prio 0, i/o): rt-ostimer
    0000000040122000-0000000040122fff (prio 0, i/o): rt.flexcomm.i2c
    0000000040127000-0000000040127fff (prio 0, i/o): rt.flexcomm.i2c
    0000000040134000-0000000040134fff (prio 0, i/o): rt.flexspi
    0000000040135000-0000000040135fff (prio 0, i/o): rt_pmc
    0000000040136000-00000000401360ff (prio 0, i/o): sdhci
    0000000040137000-00000000401370ff (prio 0, i/o): sdhci
    000000004013a000-000000004013afff (prio 0, i/o): rt-lpadc
    000000004013c000-000000004013cfff (prio 0, i/o): rt.flexspi
    000000004020d000-000000004020dfff (prio 0, i/o): rt.flexcomm
  0000000040004000-0000000040004fff (prio 0, i/o): IOPCTL
  0000000040026000-0000000040026fff (prio 0, i/o): PERIPHERAL_MUXES
  0000000040033000-0000000040033fff (prio 0, i/o): CACHE_Control_0
  0000000040034000-0000000040034fff (prio 0, i/o): CACHE_Control_1
  0000000040100000-0000000040102fff (prio 0, i/o): HS_GPIO
  0000000040204000-0000000040206fff (prio 0, i/o): SEC_HS_GPIO
0000000040012000-0000000040012fff (prio 0, i/o): armsse-cpu-pwrctrl
  000000004001f000-000000004001ffff (prio 0, i/o): armsse-cpuid
  0000000050000000-000000005fffffff (prio -1500, i/o): alias alias 3 @arm-sse-cpu-container0
0000000040000000-000000004fffffff
  0000000050010000-0000000050010fff (prio 0, i/o): cachectrl0
  0000000050011000-0000000050011fff (prio 0, i/o): CPUSECCTRL0
```

dbus_mu

Fusion DSP
(RT595_XTENSA)

```
0000000000000000-ffffffffffffffff (prio 0, i/o): system
  0000000000800000-0000000000807fff (prio 0, ram): sram0
  0000000000808000-000000000080ffff (prio 0, ram): sram1
  0000000000810000-0000000000817fff (prio 0, ram): sram2
  0000000000818000-000000000081ffff (prio 0, ram): sram3
  0000000000820000-0000000000827fff (prio 0, ram): sram4
  0000000000828000-000000000082ffff (prio 0, ram): sram5
  0000000000830000-0000000000837fff (prio 0, ram): sram6
  0000000000838000-000000000083ffff (prio 0, ram): sram7
  0000000000840000-000000000084ffff (prio 0, ram): sram8
  0000000000850000-000000000085ffff (prio 0, ram): sram9
  0000000000860000-000000000086ffff (prio 0, ram): sram10
  0000000000870000-000000000087ffff (prio 0, ram): sram11
  0000000000880000-000000000089ffff (prio 0, ram): sram12
  00000000008a0000-00000000008bffff (prio 0, ram): sram13
  00000000008c0000-00000000008dffff (prio 0, ram): sram14
  00000000008e0000-00000000008fffff (prio 0, ram): sram15
  0000000000900000-0000000000093fff (prio 0, ram): sram16
  0000000000940000-0000000000097fff (prio 0, ram): sram17
  0000000000980000-00000000009bffff (prio 0, ram): sram18
  00000000009c0000-00000000009fffff (prio 0, ram): sram19
  0000000000a00000-0000000000a3ffff (prio 0, ram): sram20
  0000000000a40000-0000000000a7ffff (prio 0, ram): sram21
  0000000000a80000-0000000000abffff (prio 0, ram): sram22
  0000000000ac0000-0000000000afffff (prio 0, ram): sram23
  0000000000b00000-0000000000b3ffff (prio 0, ram): sram24
  0000000000b40000-0000000000b7ffff (prio 0, ram): sram25
  0000000000b80000-0000000000bbffff (prio 0, ram): sram26
  0000000000bc0000-0000000000bfffff (prio 0, ram): sram27
  0000000000c00000-0000000000c3ffff (prio 0, ram): sram28
  0000000000c40000-0000000000c7ffff (prio 0, ram): sram29
  0000000000c80000-0000000000cbffff (prio 0, ram): sram30
  0000000000cc0000-0000000000cfffff (prio 0, ram): sram31
  000000003ffc0000-000000003ffdffff (prio 0, ram): xtensa.dataram1
  000000003ffe0000-000000003fffffff (prio 0, ram): xtensa.dataram0
  0000000040000000-0000000040001fff (prio 0, ram): xtensa.instram0
  0000000040111000-0000000040111fff (prio 0, i/o): dbus_client_mub
  0000000050000000-0000000053ffffff (prio 0, ram): xtensa.sysrom0
  0000000060000000-0000000067ffffff (prio 0, ram): xtensa.sysram0
```

dbus_mu

- Show time


Microsoft Teams
zds2023 recording meeting
2023-05-23 12:52 UTC
Recorded by
Hake Huang
Organized by
Hake Huang


Microsoft Teams
zds2023 recording meeting
2023-05-23 13:31 UTC
Recorded by
Hake Huang
Organized by
Hake Huang

- Extension work

1. Create several customized QEMU for CI testing
2. Use those QEMU system for code coverage analysis
3. Extend user cases with co-simulation

Refences:

- QEMU introduce https://www.usenix.org/legacy/event/usenix05/tech/freenix/full_papers/bellard/bellard.pdf
- i.MX RT595 Reference manual

https://www.nxp.com/webapp/Download?colCode=IMXRT500RM

Thank you

For any issues or questions, please feel free to email me or

Zephyr Test Working Group testing-wg@lists.zephyrproject.org