Software Flow for Complex SoC-FPGA

Cyril Jean
Director, IP Engineering
Microchip
What is an SoC-FPGA?
SoC

SoC FPGA

MSS Processor cores
- hart0
- hart1
- hart2
- hart3
- hart4

Memory
- DDR Controller
- Scratchpad(s)

MSS Peripherals
- UART
- Ethernet
- SPI
- eMMC
- I2C
- ...

Reset Ctrl
Clock Ctrl
SoC Programming

SoC FPGA

MSS Processor cores
- hart0
- hart1
- hart2
- hart3
- hart4

Clock Ctrl
Reset Ctrl

MSS Peripherals
- UART
- Ethernet
- SPI
- eMMC
- I2C

Memory
- DDR Controller
- Scratchpad(s)

Code:
```c
sw $0,$0
bnez $a1,0x004ae6
sw $1,$0
bnez $a1,0x004ae6
li $a5,0
addi $a1,$a1,-1
addi $a1,$a1,2
lbu $a1,$0($s1)
sh $a1,$0($a1)
divu $a1,$a5,$a4
bgeu $a5,4,$0004b310 $<print_i+0x210>
li $a5,8
bne $4,$a5,0x004ae6 $<print_i+0x174>
lw $a5,$0($a0)
addi $a5,$5,1
bnez $a5,0x004ae6 $<print_i+0x174>
lw $a4,$4($a0)
lw $a5,$4($a0)
blt $a5,$a4,0x004ae6 $<print_i+0x174>
lw $a5,4($a0)
addi $a5,-1($s1)
addi $s1,$s1,-1
sub $a2,$a2,24
sw $a2($s0)
sw $a4,$4
sw $a5,$5
addi $a2,$a2,12
sw $a1,$0
sw $a5,$6
j $a1,0x004ae6 $<print_common>
lw $a5,-1
bne $a0,$a5,0x004ae4 $<print_i+0x272>
li $a5,-1
id $a2,72($sp)
id $a0,64($sp)
id $a1,48($sp)
id $a2,48($sp)
id $a3,40($sp)
id $a4,32($sp)
id $a5,24($sp)
id $a6,16($sp)
addi $gp,$sp,80
jeq $a5,0($a5)
```
FPGA

- Memory
- Logic
- Math
- I/Os
- SERDES
FPGA Programming

#RISCVSUMMIT | tmt.knect365.com/risc-v-summit/
SoC-FPGA

MSS Processor cores
- hart0
- hart1
- hart2
- hart3
- hart4

MSS Peripherals
- DDR Controller
- Scratchpad(s)
- UART
- Ethernet
- SPI
- eMMC
- I2C
- ...

Memory
- DDR Controller
- Scratchpad(s)

FPGA
- Memory
- I/Os
- Logic
- Math
- SERDES
Interacting With The FPGA

- The FPGA exposes interfaces visible in the processors memory map
- Interrupts can be generated by FPGA logic
SoC-FPGA Development Board

#RISCVSUMMIT | tmt.knect365.com/risc-v-summit/
Typical System Start-up Flow

Zero stage boot loader → First stage boot loader → Second stage boot loader → Operating System

Memory Map
- Memory
  - Peripheral A registers
  - Peripheral B registers
  - Peripheral C registers

Interrupt Table
- Interrupt A
- Interrupt B
- Interrupt C
Simpler Start-up Flow

Zero stage boot loader → First stage boot loader → Bare Metal or RTOS

Memory Map
- Memory
- Peripheral A registers
- Peripheral B registers
- Peripheral C registers

Interrupt Table
- Interrupt A
- Interrupt B
- Interrupt C
Why stop at one context?

Zero stage boot loader → First stage boot loader → Second stage boot loader → Operating System → Bare Metal or RTOS

Memory Map
- Memory Region A
  - Peripheral A registers
  - Peripheral B registers
  - Peripheral C registers

Interrupt Table
- Interrupt A
- Interrupt B
- Interrupt C

Memory Map
- Memory Region B
  - Peripheral D registers
  - Peripheral E registers
  - Peripheral F registers

Interrupt Table
- Interrupt D
- Interrupt E
- Interrupt F

#RISCVSUMMIT | tmt.knect365.com/risc-v-summit/
Development Flow

MSS "C" Configuration Data

Libero® MSS Component

Compile

Libero IDE

Third Party IDEs

SoftConsole

Memory Clients

eNVM via JTAG

sNVM, eNVM

Application Image (Linux®, other)

Software IDE

Libero® MSS Configurator

Standalone MSS Configurator

#RISCVSUMMIT | tmt.knect365.com/risc-v-summit/
PolarFire SoC Configurator

#RISCVSUMMIT | tmt.knect365.com/risc-v-summit/
Configuration meta-data

```
# RISCV-SUMMIT | tmt.knect365.com/risc-v-summit/
```
Resources

https://www.microsemi.com/polarfire-soc

https://github.com/polarfire-soc