CO503 - 2020

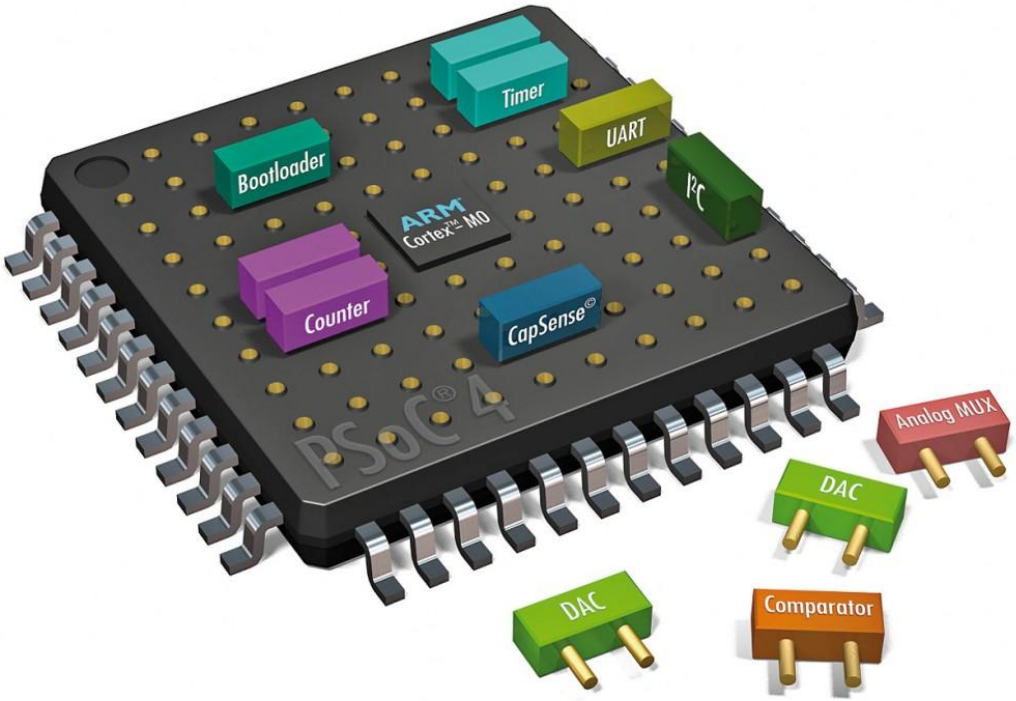
Advanced Embedded Systems

Processor Customizations

ISURU NAWINNE

RECAP …

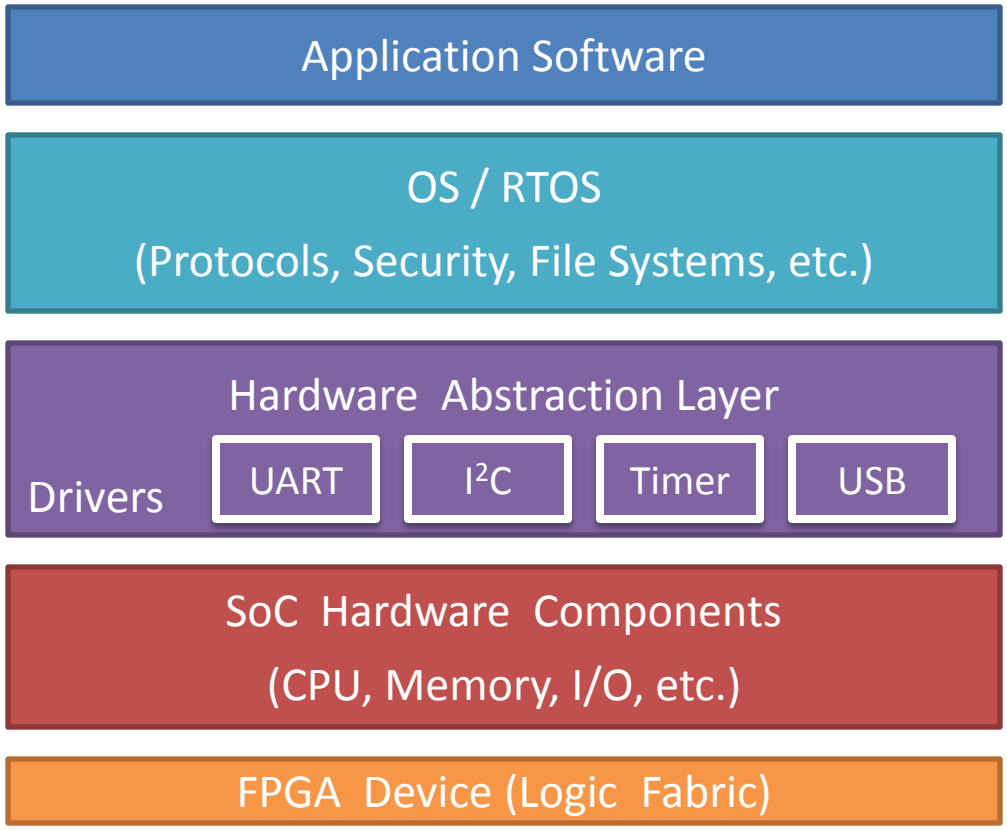
SoC Design & Prototyping using FPGA Tools



- FPGA → logic blocks
- MIPS CPU *(Nios II)*
- Other hardware components *(IP blocks)*
- Hardware software co-design
- Memory-mapped I/O

- We used a general purpose CPU

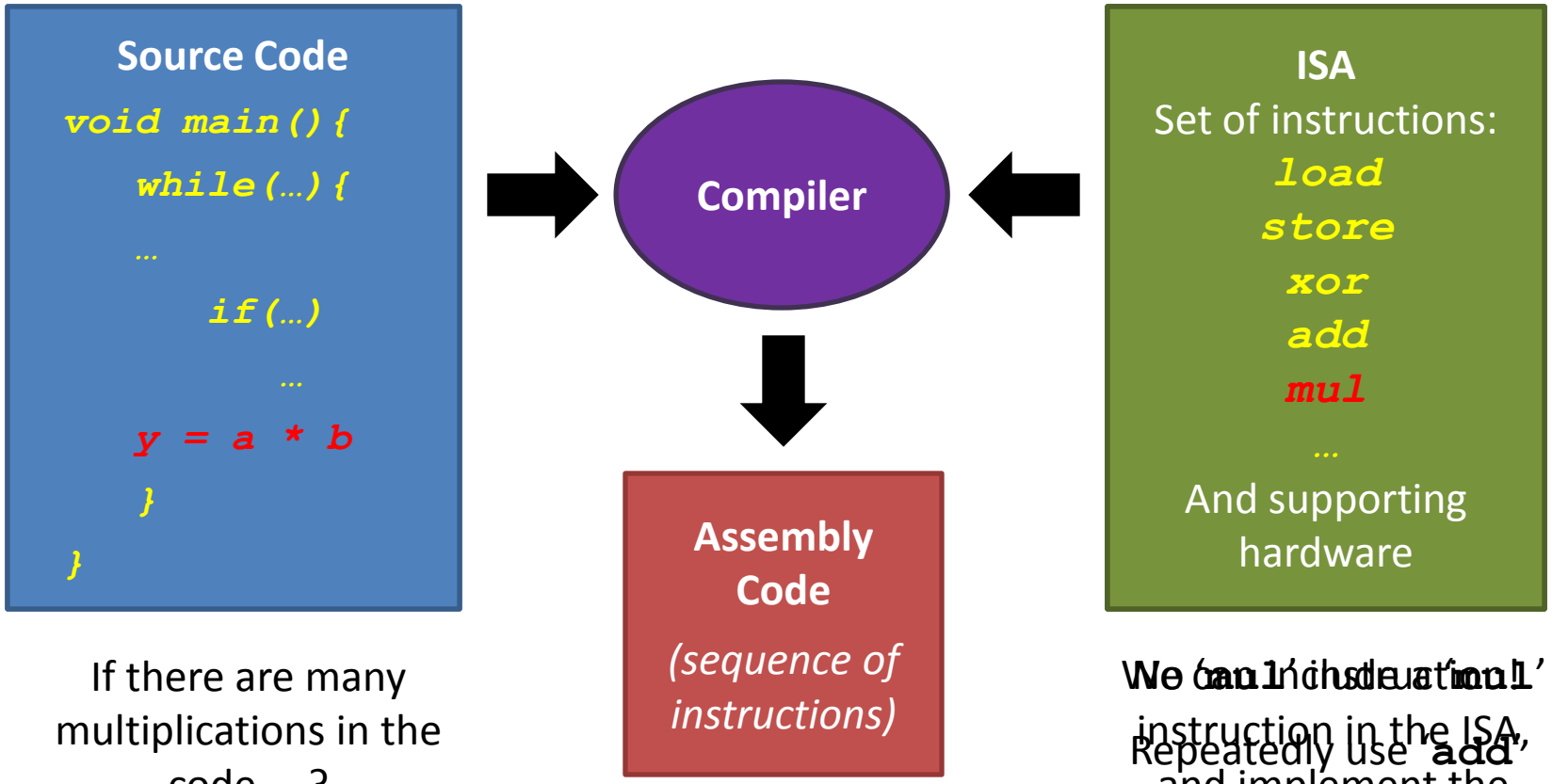
System Overview



Customizing a CPU for a specific application

- Why customize?
- General-purpose processors vs. ASIP
- What type of customizations are available?
  - ISA extensions / custom instructions
  - Memory sub-system optimizations (cache memory)

Source Code, Compiler & ISA



If there are many multiplications in the code … ?

We can include a ‘mul’ instruction in the ISA, and implement the required hardware

Repeatedly use ‘add’

CUSTOM INSTRUCTIONS

The process of adding a custom instruction...

1) Analyse the *source-code* along with the *assembly code*

2) Identify *time consuming blocks*
(repeating, multiple instructions)

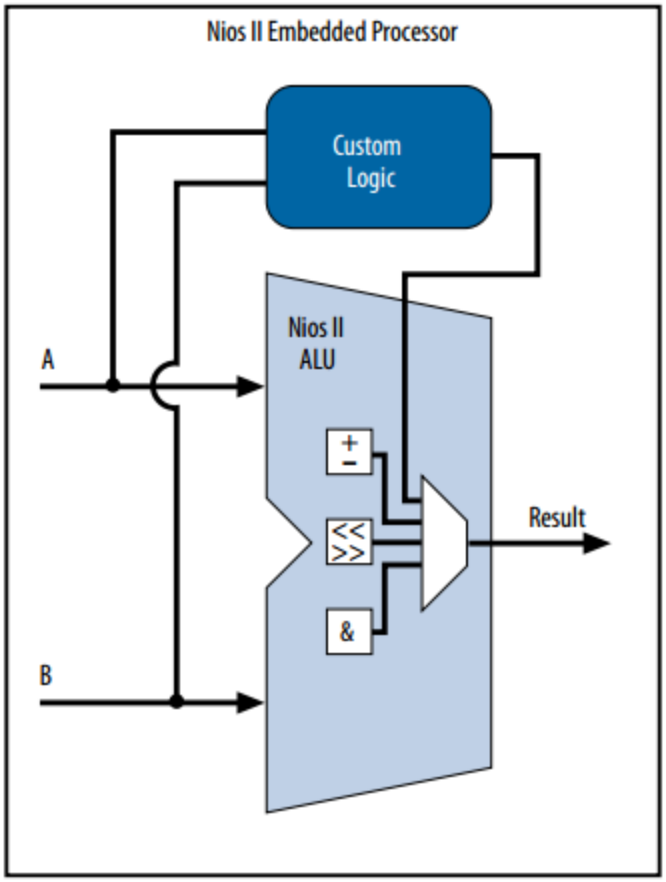
3) Define a *single instruction* to replace such blocks

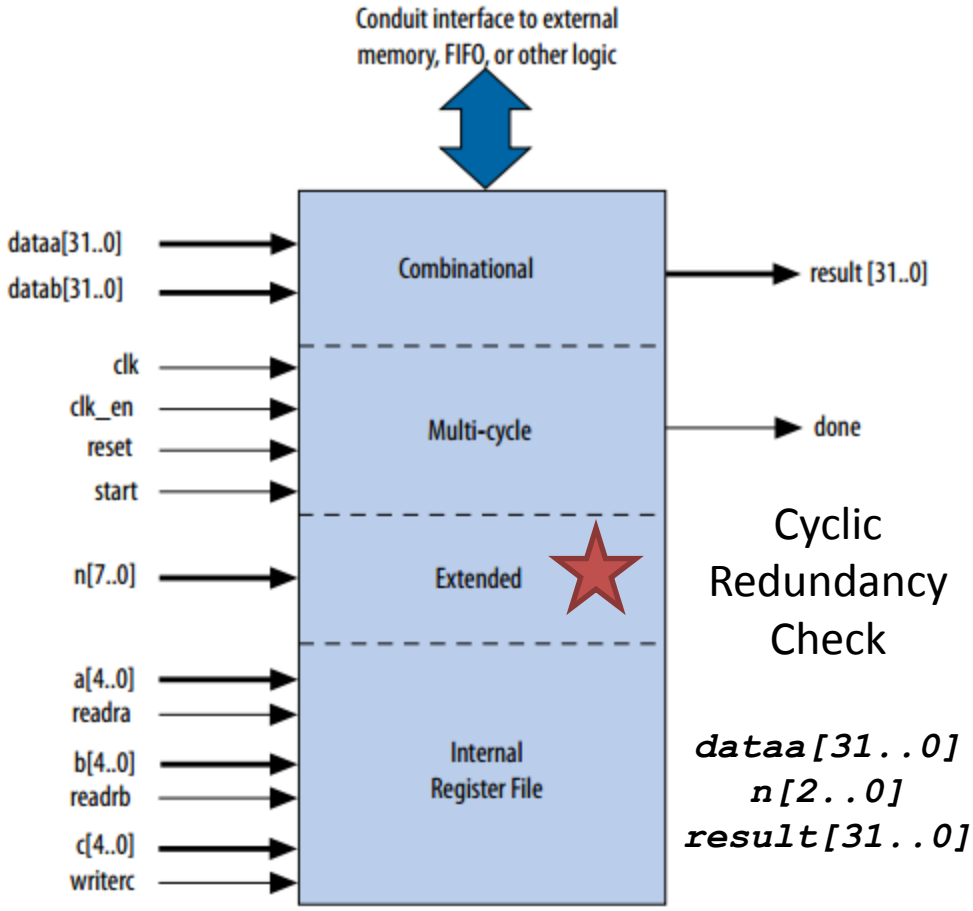
4) Design the *hardware* required to implement that instruction

5) Integrate the new hardware into the datapath !

6) How to use the custom instruction?
   - Integrate with system libraries, and use explicitly
   - Implement compiler support

Custom instructions in NIOS II processor





Cyclic Redundancy Check

```
dataa[31..0]
n[2..0]
result[31..0]
```

Read the given user guide

CUSTOM INSTRUCTIONS

Adding a custom instructions to a NIOS II processor (Hardware)

1) Create the hardware unit for the custom instruction *(HDL)*
2) Add as a New Component to the library in Qsys
   a. give a name "CRC_CUSTOM"
   b. provide the HDL files and denote the top-level, then analyze
   c. set up the interfaces and signals *(guide!)*
3) Add an instance of CRC_CUSTOM and name it "crc"
4) Assign a selection index base value (0) *(what is this?)*
5) Connect "crc" to the *custom_instruction_master* port of the CPU
6) Add a second timer (at *us* scale) and name it "high_resolution_timer"
7) Generate and compile

CUSTOM INSTRUCTIONS

Adding a custom instructions to a NIOS II processor (Software)

1) *system.h* -> `custom x, n, A`

   `__built_in_custom_ini(x, n, A)`

2) Study the provided source code
   a. CRC algorithm needs to calculate the remainder of a division
   b. method 1: iterative modulo 2 division in S/W
   c. method 2: using a look-up table (optimized)
   d. method 3: using the custom instruction
      (XOR and shift in H/W, in parallel)
3) Assign the "high_resolution_timer" for time-stamping *(BSP)*
4) Build and run. Compare the performance of the three methods

SoC from Lab 1 ...