Tokyo Tech

RISC-V AI Vision Accelerator Design Methodology on C2RTL System-Level Design Verification Framework

Tsuyoshi Isshiki, Hansen Wang

Dept. of Information and Communications Engineering
Tokyo Institute of Technology

MPSoC ‘24
July 8th, 2024

Presentation Outline

✧ RISC-V System Design Platform

- ➢ C++ based RISC SoC test design using C2RTL
- ➢ RISC-V HW/SW Integrated Development Environment (HW/SW IDE)
- ➢ 5-stage RV32/64 IMAFD (Linux-enabled)
- ➢ 3-stage RV32 IMAFD (for low-end RTOS)

✧ RISC-V Embedded AI Vision Accelerator Design Using C2RTL

- ➢ CNN accel.(Resnet/VGG): 576 MACs, 622 GOPs (peak) @ 540MHz
- ➢ YOLOv8 accel. (classify, seg, pose): 1152 MACs, 1.37 TOPs (peak) @ 595MHz

RISC-V Based AI SoCs

- **25 billion RISC-V Based AI SoCs by 2027 (Semico Research)**
  - **RISC-V Based AI SoC revenue : $291B (2027)**
  - **RISC-V CPU Semiconductor IP (SIP) 34.9% CAGR through 2027**
  - **RISC-V SW ecosystem expands adoption in consumuer/enterprise SoC markets**
  - **RISC-V ratifications of 15 new specifications : vector, scalar cryptography, hypervisor**
- **Design opportunities and challanges of RISC-V architecture**
  - **RISC-V ISA specifications help build SW ecosystems for RISC-V based SoCs**
  - **Large opportunities for drastic improvements in compute/power efficiencies with *custom instruction extensions* and *HW accelerations***

→ ***Need System-Level SW/HW Design Environment for efficiently building customized RISC-V cores, HW accelerators and SoC architectures***

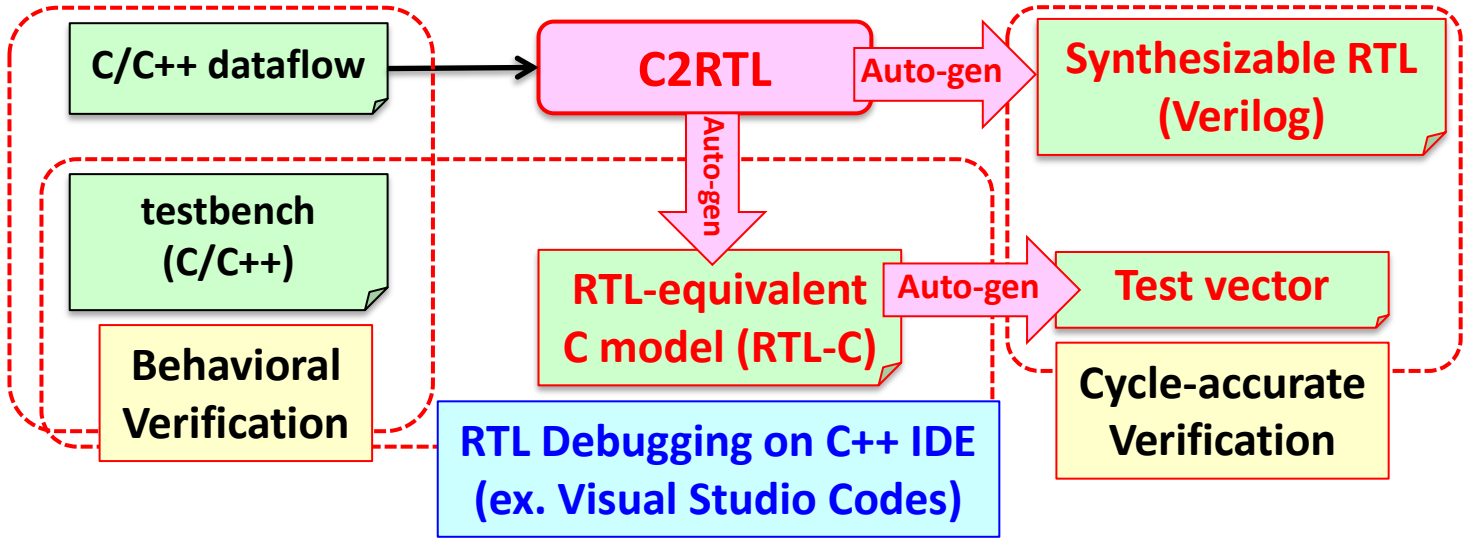
RISC-V System Design Platform

(funded by NEDO : 2022 - 2025)

- **C/C++ based RISC SoC test design using C2RTL**
  - **RISC-V core: 32/64-bit IMAFD ISA-subset configurable from a single source**
  - **I/D caches + MMU (TLB + page walk logic)**
  - **AXI4: master devices, slave devices, bus arbiters + interconnect**
  - **C++ based system-level integration of C++ components (UART, memory controller)**
  - **RISC-V SoC test design verified on FPGA, logic emulator, 28nm gate-level simulator → confirmed competitive performance and power efficiency with existing RISC-V cores**
- **RISC-V HW/SW Integrated Development Environment (HW/SW IDE)**
  - **HW design : RISC-V core + instruction extension + HW accelerator**
  - **SW design : (LLVM-based) compiler (ext. support), applications, middleware, OS**
  - **HW/SW co-verification : debugger probes (reg-file, memory) extending to RTL signals (pipeline registers, feedback wires)**

C2RTL Design Framework [MPSoC '15 – '19, '22 – '23]

- **C/C++ dataflow model : directly describe RTL behavior in C/C++**
  - HW attributes (bit-width/register/memory) with *GCC-attributes*
  - No language extensions, no built-in classes : *easy & intuitive coding*
  - Single-cycle behavior (register/memory updates) : *the only design constraint*
  - Object-Oriented RTL modeling : *C++ classes & templates*
  - SoC modeling : *RTL-IP generation & interconnect*



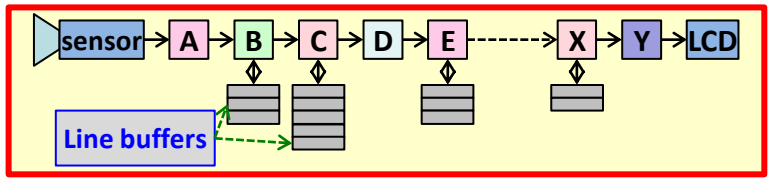
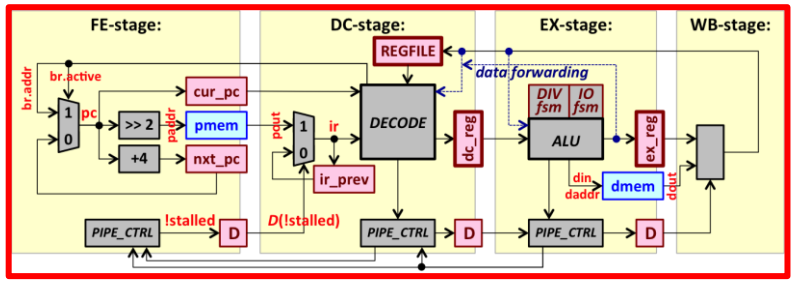
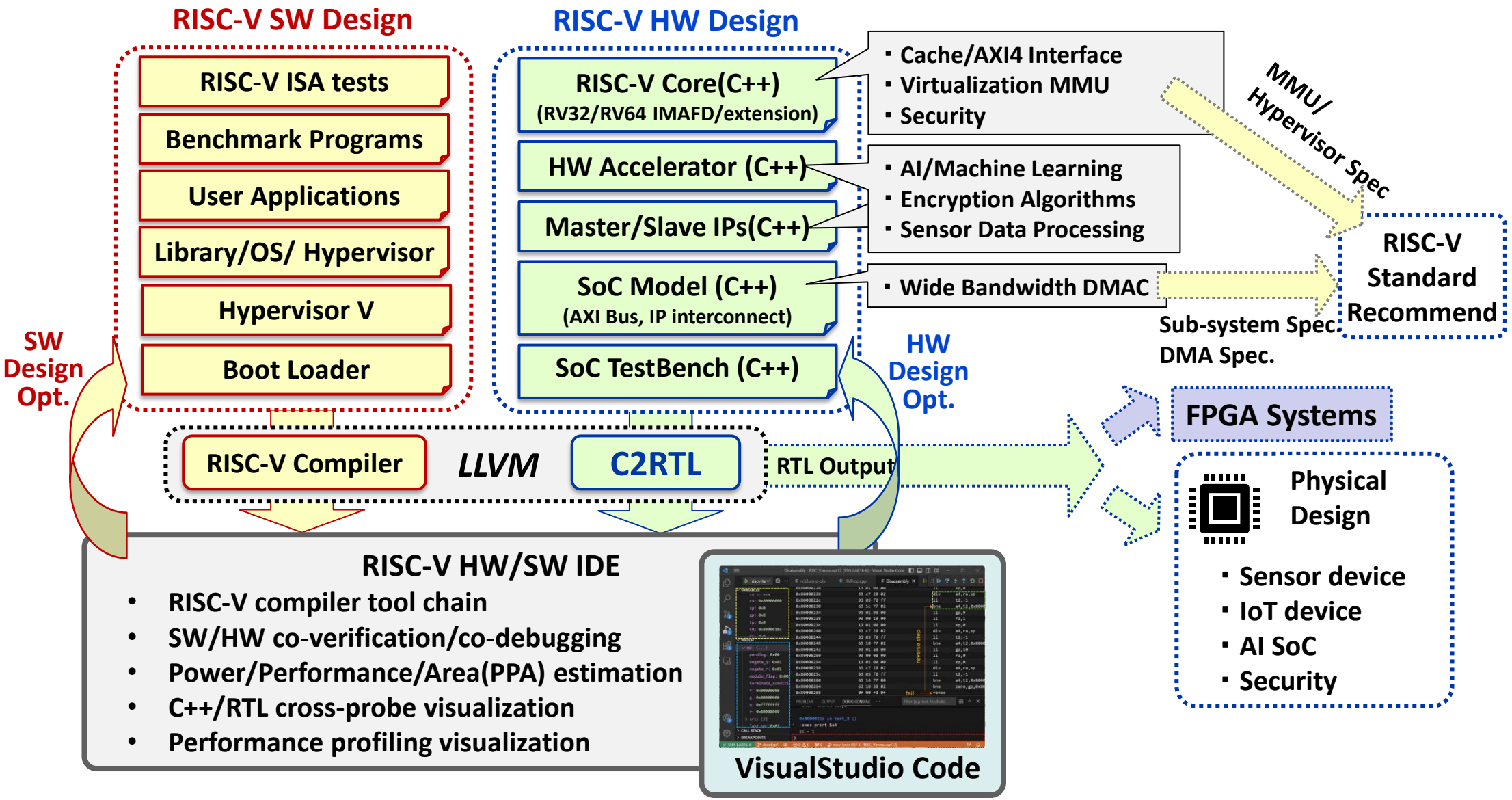


Image processing pipeline, deep learning

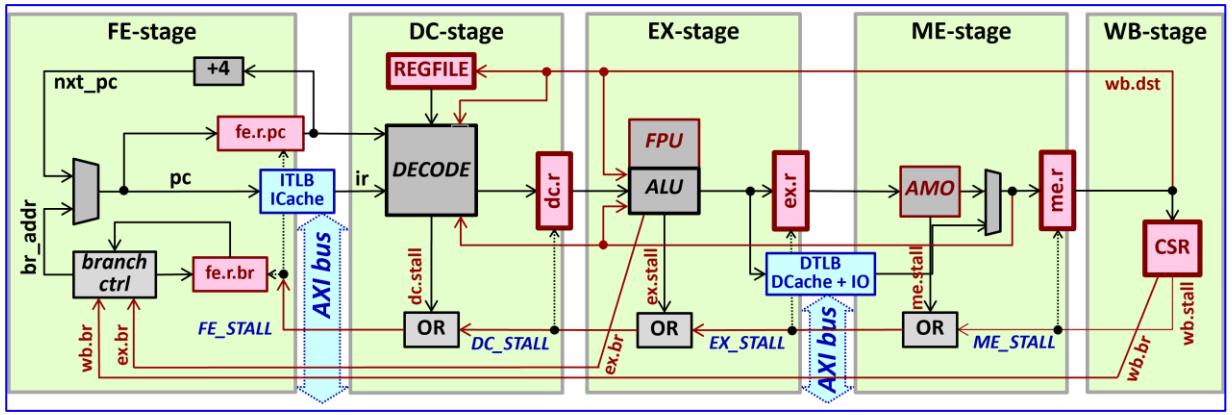


Processor pipeline

RISC-V HW/SW Integrated Design Environment (IDE)

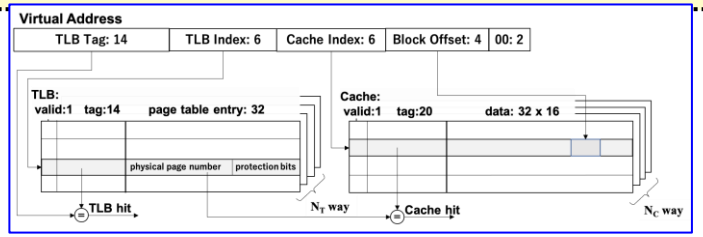


RISC-V Scalar Core Models From C++ Descriptions

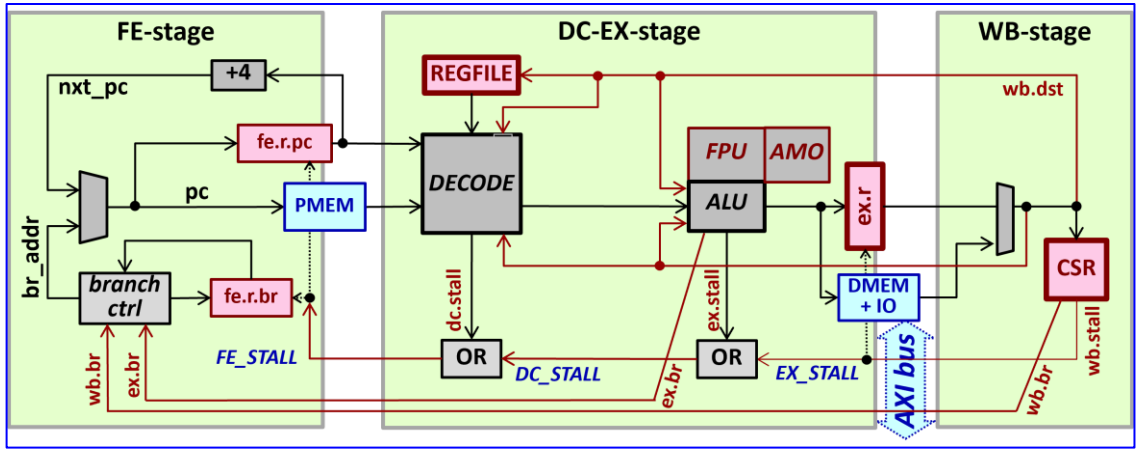


5-stage (Linux-OS enabled, MMU + Caches) : 740MHz @ 28nm

- MMU = TLB + page-walk logic (VIPT)
- I-Cache/I-TLB → AXI-Master port
- D-Cache/D-TLB/IO → AXI-Master port
- ISA : RV32/64-IMAFD (M/F/D : optional)
- Priviledge modes : U/M/HS/VU/VS



VIPT : Virtually-indexed physically tagged

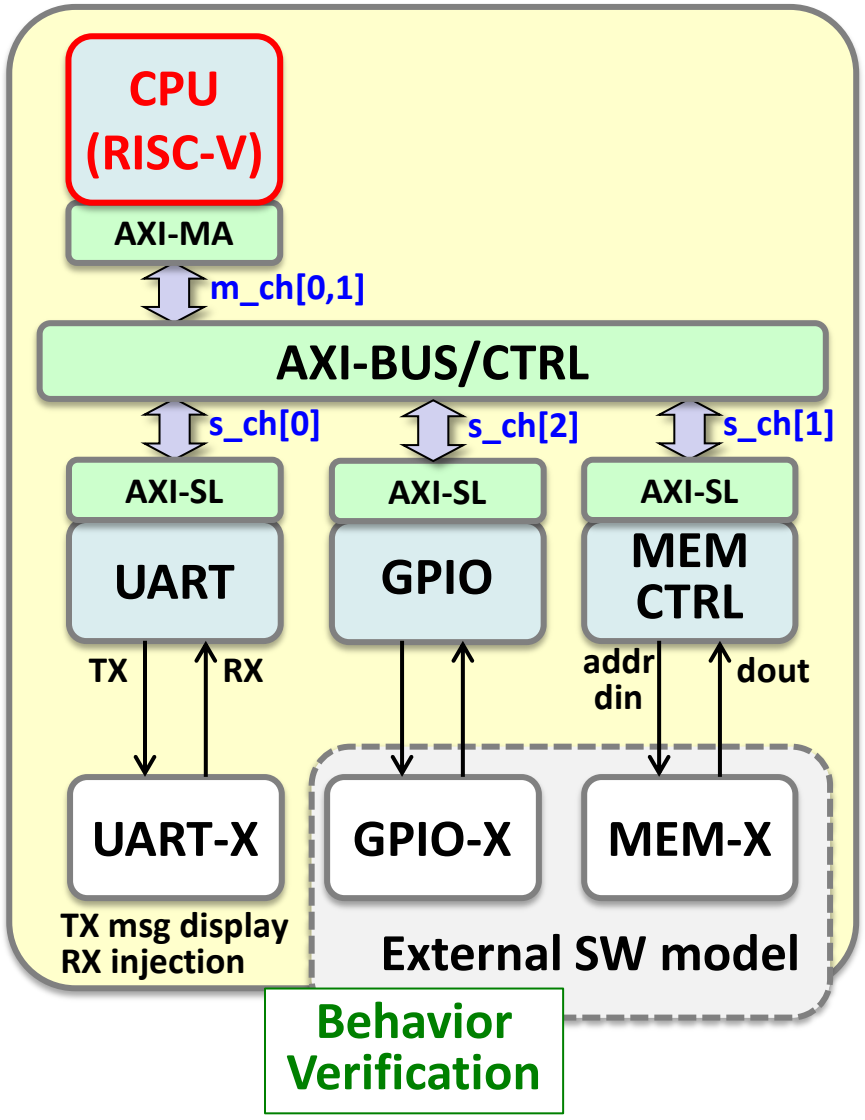


3-stage cache-less : 650MHz @ 28nm (est.)

- For embedded apps: no cache, no MMU
- ISA : RV32-IMAFD (M/F/D : optional)
- Priviledge modes : U/M
- Higher IPC : no branch stalls

All RISC-V scalar cores designed in C++ and translated to RTL by C2RTL

System-Level SoC Modeling in C++




```
#define _C2R_MODULE  __attribute__((C2R_module))
```

System-level top-function (contains IP-level top-function calls)

```
_C2R_MODULE
int RVProcAXI (IOPin *io, AXI4L::BUS<2, 4> *axi) {
  axi_uart.step (&axi->s_ch[0], &io->uart);
  axi_memctl.step (&axi->s_ch[1], &io->mpin);
  axi_gpio.step (&axi->s_ch[1], &io->gpio);
  val = cpu1.step (&axi->m_ch[0], &axi->m_ch[1], io->intr);
  axi_bus_ctrl.connectChannel (axi);
  io->uart.rx = uart_ext (io->uart.tx);
  return val;
}
```

single-cycle behavior description

Fully user-defined data structures (not built-in classes)

```
struct IOPin { /// SoC IO pins
  struct UARTPin { BIT tx, rx; } uart ; /// uart pins
  struct MEMCTLPin {  /// memory pins
    UINT32 addr, din, dout;
    UINT3  size;
    UINT4  cs;
    BIT    we, ras, cas;
  } mpin ;
  struct GPIOPin { UINT32 dout, din; } gpio ; /// gpio pins
  UINT32 intr; /// external interrupts
};
```

Presentation Outline

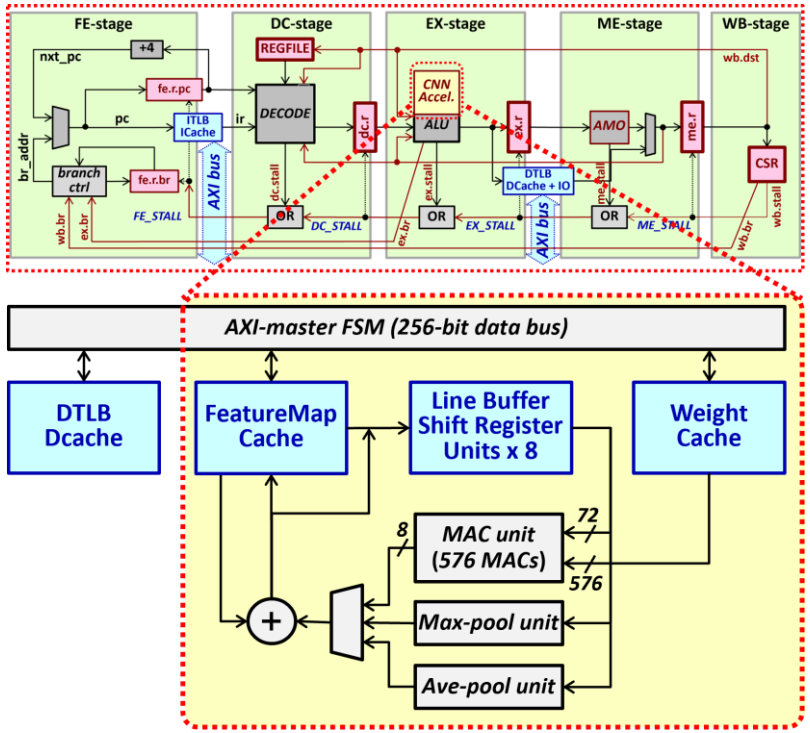
✧ RISC-V System Design Platform

- ➢ C++ based RISC SoC test design using C2RTL
- ➢ RISC-V HW/SW Integrated Development Environment (HW/SW IDE)
- ➢ 5-stage RV32/64 IMAFD (Linux-enabled)
- ➢ 3-stage RV32 IMAFD (for low-end RTOS)

✧ RISC-V Embedded AI Vision Accelerator Design Using C2RTL

- ➢ CNN accel.(Resnet/VGG): 576 MACs, 622 GOPs (peak) @ 540MHz
- ➢ YOLOv8 accel. (classify, seg, pose): 1152 MACs, 1.37 TOPs (peak) @ 595MHz

RISC-V Embedded CNN Accelerator [1][2]



- CNN accelerator embedded inside 5-stage RISC-V pipeline
- MAC Array : 576 MACs → 622 GOPs (peak) @ 540MHz
- Core Layout @ 28nm → 137.7mW (4.51 TOPs/W peak), 8.3mm²
- AXI-BUS : 256-bit data width
- External memory bandwidth : 17.2GB/s (dedicated DMAC)
- On-chip memory bandwidth : 354.2GB/s

CNN type	Speedup vs. SW	Ave. MAC utilization	GOPs	TOPs/W (actual)
Resnet34	3808 x	57.4%	357.5	2.58
Resnet50	7897 x	43.4%	270.5	1.95
Resnet101	7939 x	50.4%	314.3	2.27

Layer types	#MACs per kernel	Inputs	Outputs	Total #MACs
CONV 7x7	49	1	8	392 (68%)
CONV 3x3	9	8	8	576 (100%)
CONV 1x1	1	64	8	512 (88%)
Fully-connected	1	64	8	512 (88%)

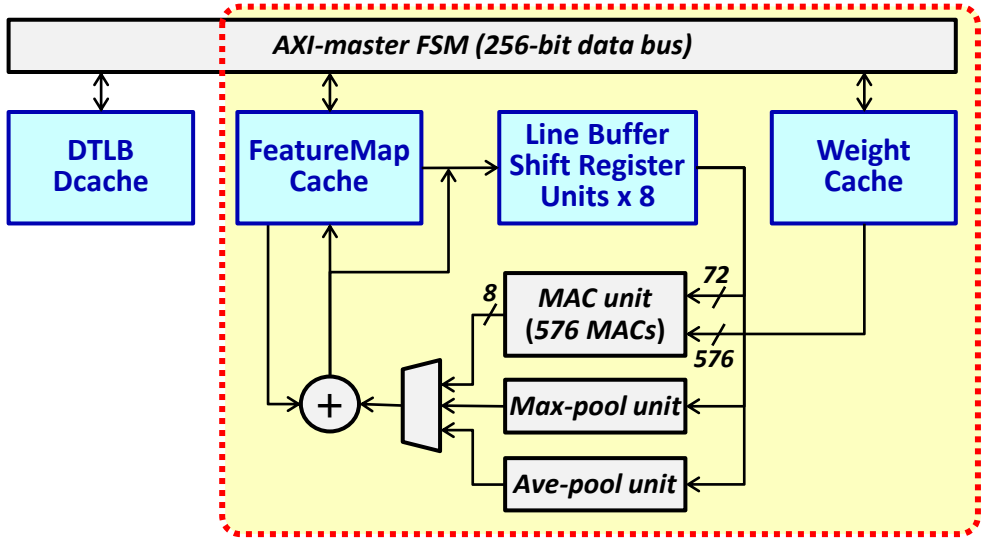
vs. GTX1080Ti	Throughput	Energy consumption
Resnet101	1.04 x	1/1897 x

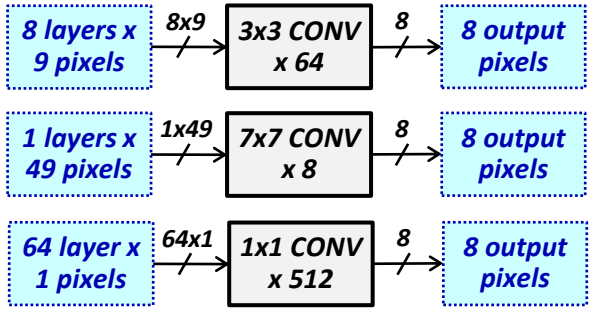
[1] H. Wang, D. Li, T. Isshiki, "Reconfigurable CNN Accelerator Embedded in Instruction Extended RISC-V Core", ICET 2023

[2] H. Wang, D. Li, T. Isshiki, "A Low-Power Reconfigurable DNN Accelerator for Instruction-Extended RISC-V", IPSJ Trans. SLDM (2024)

CNN Datapath & On-Chip Memory Subsystem

- **MAC array unit :**
  - 8-bit 576 MAC units (configurable to various CONV kernel sizes)
  - Dynamic Fixed-point format → (8 channels per exponent) adjusted in batch norm module
- **On-chip memory subsystem :**
  - Line-buffer + shift-register + feature-map frame buffer → 354.2GB/s on-chip bandwidth
  - Dedicated DMAC + 256-bit wide AXI bus → 17.2GB/s external bandwidth





configurable to various CONV kernel sizes

Layer types	#MACs per kernel	Inputs	Outputs	Total #MACs
CONV 7x7	49	1	8	392 (68%)
CONV 3x3	9	8	8	576 (100%)
CONV 1x1	1	64	8	512 (88%)
Fully-connected	1	64	8	512 (88%)

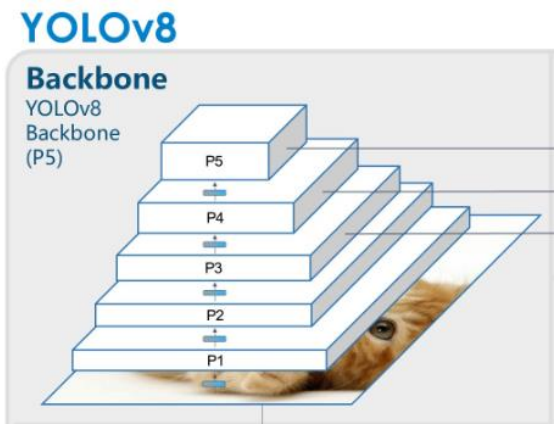
RISC-V Instruction Extensions for CNN Accelerator

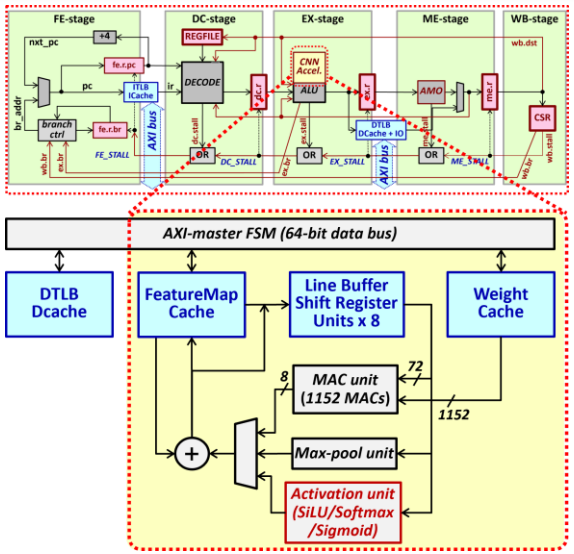
- **Fine-grain instruction-level SW control of the accelerator engine**
  - Allows *flexible SW implementation* of different CNN layer parameters
  - *SW-based design verification* and *debugging* made easier compare to hardwired state-machine control
  - CNN accelerator operates on virtual address space allowing easy user application development (no device drivers required)
- **Custom instructions added to RV32-IMA instruction-set**
  - DMAC-based data transfer instructions : weights, feature-map data

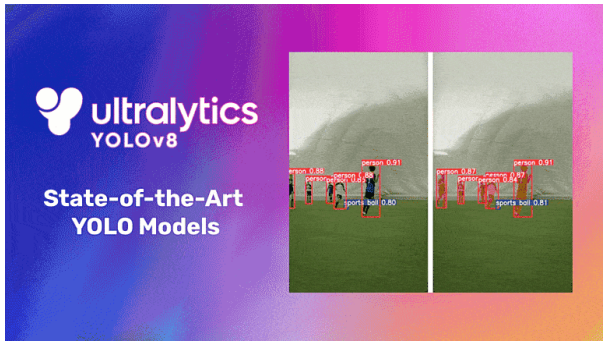
  → *DMA cycles hidden behind compute instruction cycles*
  - CNN compute instructions : convolution, max-pool, average-pool, linear
  - CNN accelerator setup instructions
  - Zero-overhead loop instructions

YOLOv8 Acceleration on RISC-V ISA Extension

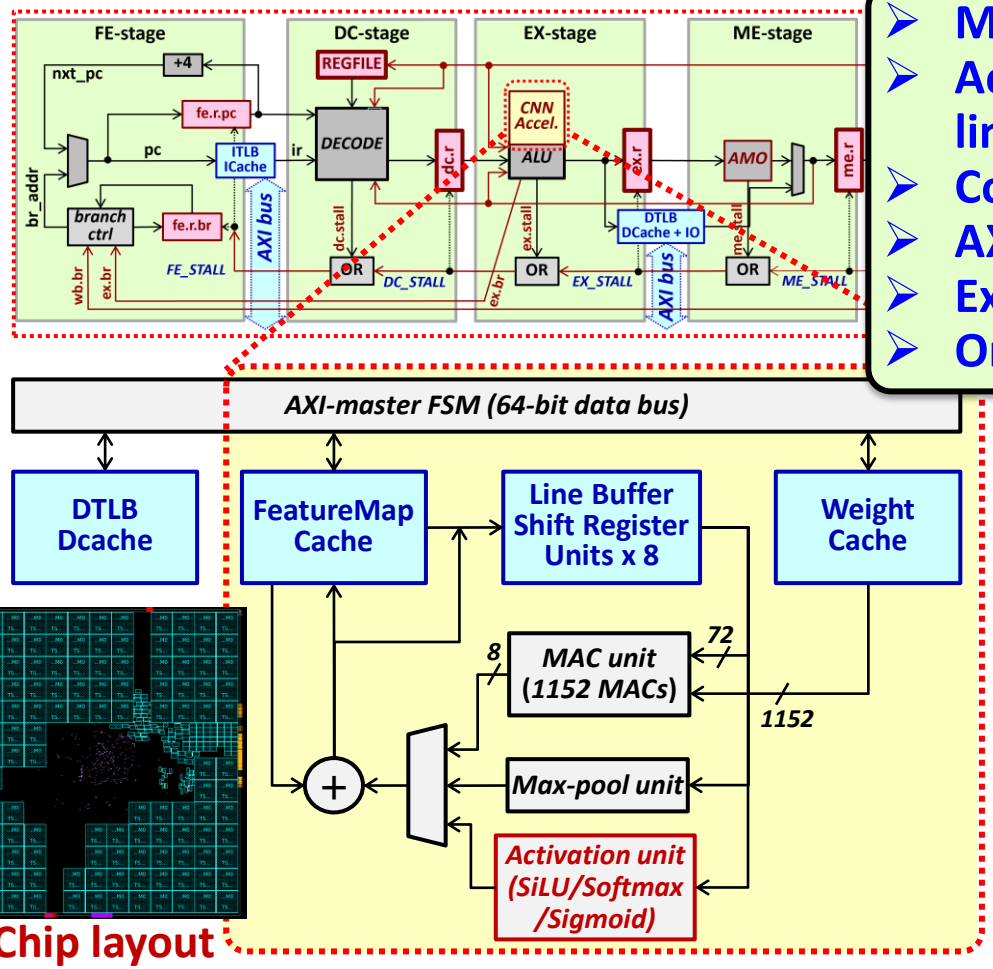
- ✧ **YOLOv8 : enhanced backbone network and features**
  - ➢ Features : detection, segmentation, pose estimation, tracking, classification
- ✧ **RISC-V ISA extension for YOLOv8 implementation**
  - ➢ HW accelerator controlled by ISA extensions (96.5% of total workload)
  - ➢ SW workload (3.5%) : Non-maximum suppression, anchor box, etc.







RISC-V Embedded Yolo.v8 Accelerator [3][4]



Chip layout

- MAC Array : 1152 MACs → 1.37 TOPs (peak) @ 595MHz
- Activation unit (SiLU, softmax, sigmoid) → hardwired piecewise linear approximation
- Core Layout @ 28nm → 202.5mW (6.76 TOPs/W peak), 7.9mm^2
- AXI-BUS : 64-bit data width
- External memory bandwidth : 4.76 GB/s (dedicated DMAC)
- On-chip memory bandwidth : 733 GB/s

Application	Ave. MAC util.	TOPs	TOPs/W
Yolo.v8s	42.3%	0.58	2.85
Yolo.v8s-seg	51.8%	0.71	3.50
Yolo.v8s-pose	43.0%	0.59	2.90

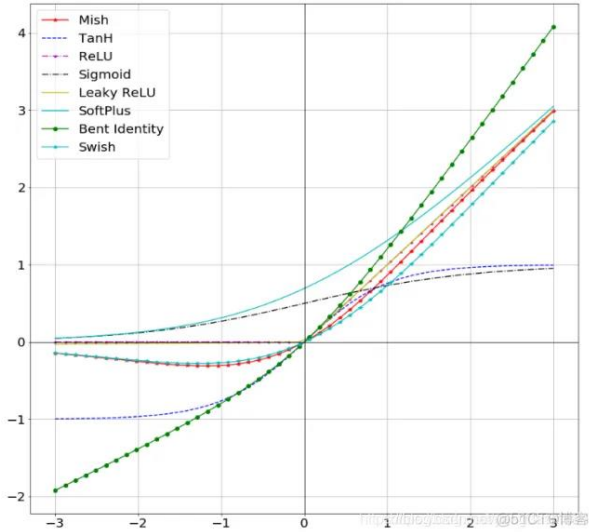
vs. GTX1080Ti	Throughput	Energy consumption
Yolo.v8s	0.133 x	1/166 x
Yolo.v8s-seg	0.132 x	1/165 x
Yolo.v8s-pose	0.136 x	1/170 x

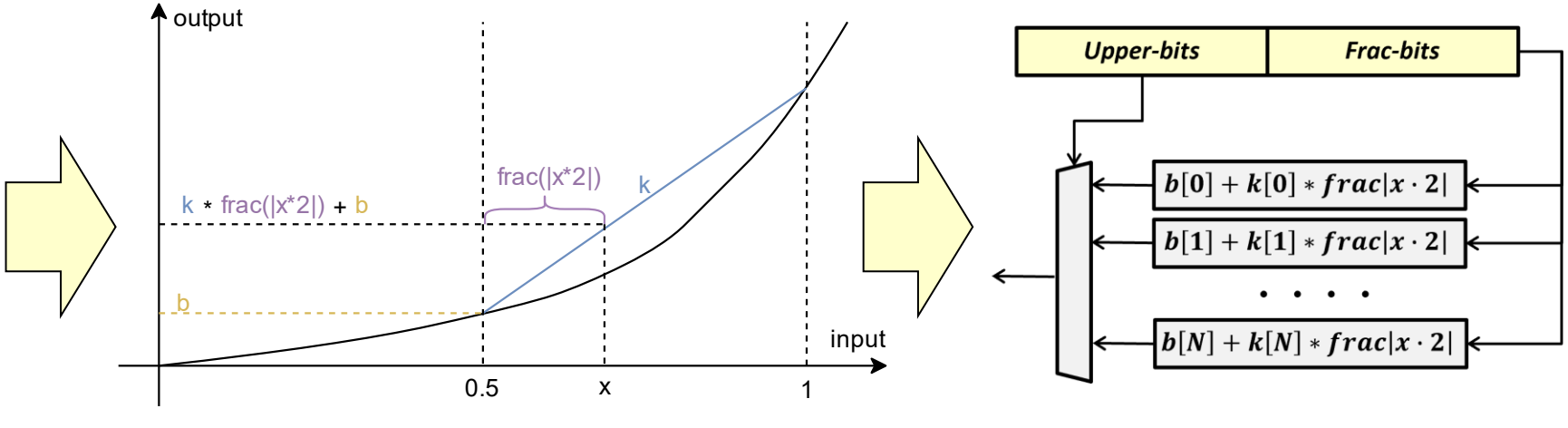
[3] H. Wang, D. Li, T. Isshiki, "A power-efficient end-to-end implementation of YOLOv8 based on RISC-V", CAIT 2023

[4] H. Wang, D. Li, T. Isshiki, "Energy-Efficient Implementation of YOLOv8, Instance Segmentation, and Pose Detection on RISC-V SoC", IEEE Access (2024)

Hardwired Piecewise Linear Approximator (Yolo.v8 : SiLU, Softmax, Sigmoid)

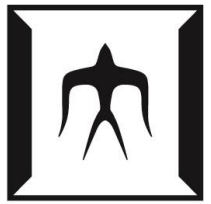
- Non-linear activation functions → hardwired piecewise linear approximator
- Slope (k) and intercept (b) values stored in LUTs → on-the-fly approximation





Summary

- **RISC-V System Design Platform on C2RTL framework**
  - **Directly describe RTL structures on C++ dataflow description**
  - **System-level modeling : IPs (processors, busses, peripherals) + system integration (IP connections)**
  - **RISC-V core: 32/64-bit IMAFD ISA-subset configurable from a single source**
  - **RISC-V SoC test design verified on FPGA, logic emulator, 28nm gate-level simulator → confirmed competitive performance and power efficiency with existing RISC-V cores**
- **RISC-V AI Vision Accelerator Architectures :**
  - **CNN accel.(Resnet/VGG): 576 MACs, 622 GOPs (peak) @ 540MHz**
  - **YOLOv8 accel. (classify, seg, pose): 1152 MACs, 1.37 TOPs (peak) @ 595MHz**



Thank You for Your Attention!

Tsuyoshi Isshiki

isshiki@ict.e.titech.ac.jp

Dept. of Information and Communications Engineering

Tokyo Institute of Technology

Portion of the contents are based on results obtained from a project, JPNP16007, commissioned by the New Energy and Industrial Technology Development Organization (NEDO)