
Dept. Electrical and Electronic Information Engineering
Toyohashi University of Technology

Custom Computing Systems Laboratory

March 2024

Shuichi Ichikawa, Prof.

`ichikawa@tut.jp`

Prof. Shuichi Ichikawa

Methods

- Computer Architecture
 - Microprocessors, Parallel processors
- Custom computing
 - Embedded systems, Co-processing
- Reconfigurable Computing
 - FPGA, Hardware partial evaluation
- High Performance Computing
 - Parallel Processing, Load Balancing
- Security
 - Crypto-circuits, RNG, Secure system
 - Information hiding, Digital signature

Applications

Based on

Performance
Tuning



Spectrum

- Hardware
 - Reconfigurable computing systems
 - Application-specific accelerators
 - Embedded systems
 - Secure systems
- Software / Application
 - Parallel processing
 - High Performance Computing
 - Security of information and systems

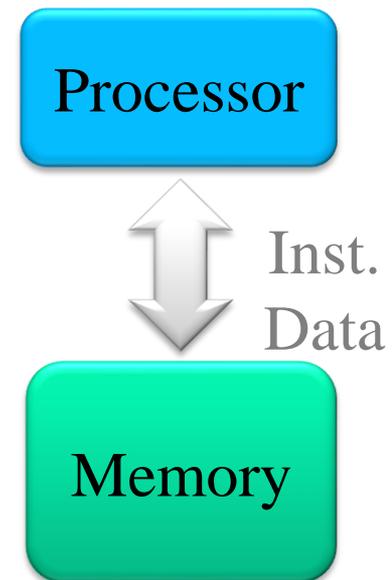


Custom Computing System

- When the performance of software is unsatisfactory...
 - Accelerate it by hardware implementation
 - Very natural or commonplace
- There have been many implementations
 - Signal processing, Image processing
 - Symbolic computation, Database
 - Scientific computing
- Not always successful !!

Why hardware is faster than software?

- Software is executed by a processor (hardware)!
- Von-Neumann bottleneck
 - Instructions are executed sequentially
- Hardwired sequence control
 - No need to execute an instruction sequence
 - Conditional branches limits the performance
- Parallel execution of arithmetic
 - Physical parallelism (many units)
 - Temporal parallelism (pipelining)
- More memory bandwidth
 - Arbitrary number of memory banks
 - Dedicated wires between units
 - Dedicated memory units for operation



Merits of Custom Computing

(What are impossible with general purpose computers)

- Optimal arithmetic with optimal data type
 - There are pre-defined data types in general computing
 - 8-bit, 16-bit, 32-bit, 64-bit, IEEE single float, double float, ...
 - You can adopt arbitrary length of data in custom computing
 - Arbitrary expression of data (redundant form, etc.)
 - Reduction of logic scale → More parallelism, more performance
 - Application-specific = fixed algorithm
 - Reduction of resources: Arithmetic, Memory, Wire
 - More parallelism
 - More performance / cost
-

Problems of Custom Computing

(What are drawbacks of its merits)

- High cost for design and implementation
 - You have to design and implement it by yourselves
 - Much effort and time are required for design, implementation, and debug.
 - One-off means expensive; no cost reduction by mass production
 - General purpose system is very easy and cheap to adopt
- Long period for development
 - General purpose hardware evolves very quickly
 - Its cost also decreases rapidly
 - Performance advantage is offset

Target of Custom Computing

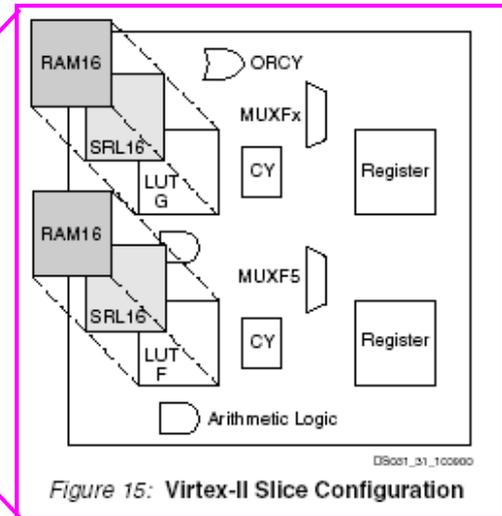
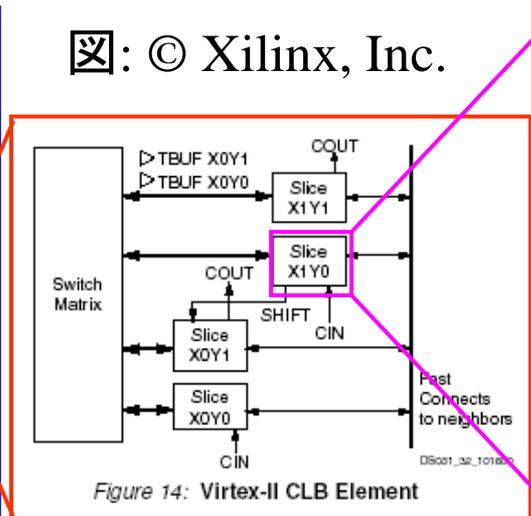
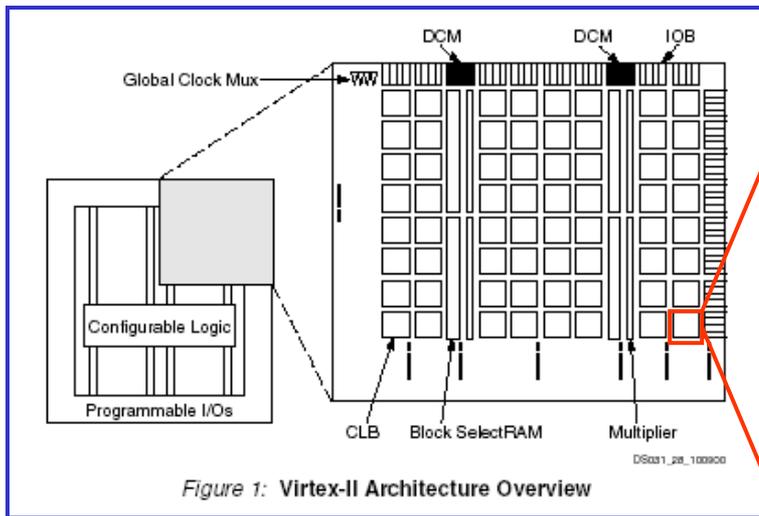
- Custom computing system is `niche`
 - Its function is limited. Its cost is higher than PC.
 - It does not replace general purpose computers
 - It is used for a limited variations of applications, where the merits of custom computing pay for the drawbacks.
- Custom computing must realize something that cannot be done by a general-purpose system
 - Performance per energy consumption
 - Real-time systems, Embedded systems
 - Game, Virtual Reality, ...
- Add-in or Accelerator to a general system
 - Graphics board, GPU

Supports to Custom Computing

- FPGA (Field Programmable Gate Array)
 - Is replacing traditional ASICs
 - ASIC = Application Specific Integrated Circuits
 - Can implement various circuit by downloading configuration
 - Might be re-configured on-site
 - Is suitable to implement one-off LSI
 - General purpose, mass-produced part
 - Produced by a cutting-edge process technology
 - Reasonable performance at reasonable cost
- Rapid progress in CAD and PC technology
 - Enabled to design an FPGA chip in a modest time
 - Enabled to design a large system at a reasonable time and cost

Background: Field Programmable Gate Array (FPGA)

- Piles of RAMs and switches
- Suited for prototyping, one-off system for a specific application
- Logic can be reconfigured on-site



Past Project: Custom Hardware for Embedded and Control Applications

- Converts a PLC instruction sequence into logic circuit
 - Co-operation with Yashima Netsugaku Corp. in Toyohashi City.



Perfect Layer Winder (prototype)



FPGA controller demo

Background: control application

- PLC (Programmable Logic Controller)
 - A kind of computer
 - Used for various sequence control applications
- Problems
 - Performance
 - Not fast enough for large control systems
 - Intellectual property
 - PLC program is easy to duplicate and to analyze

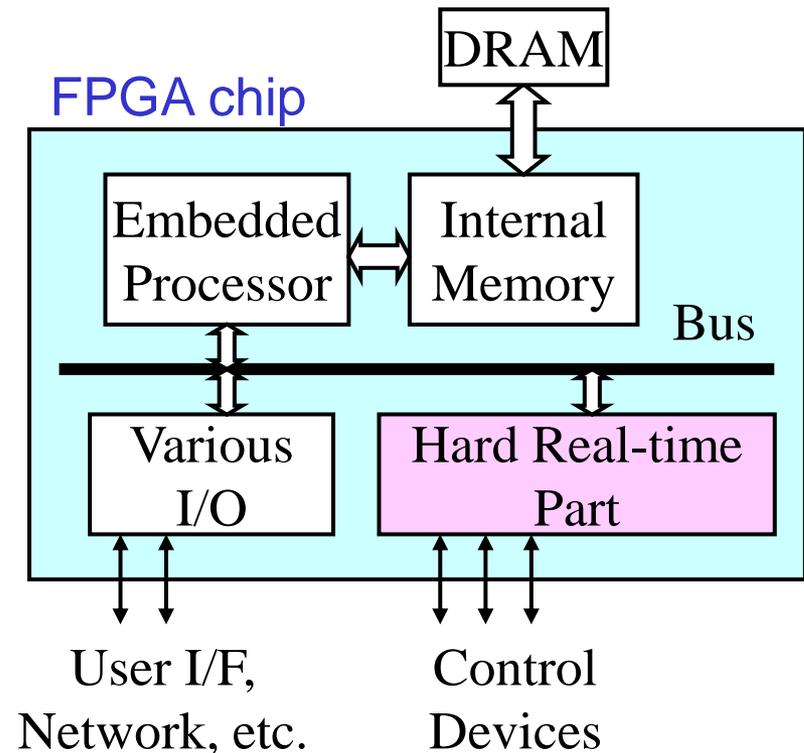


https://commons.wikimedia.org/wiki/File:A_series_PLC.jpg

Supposed System Configuration

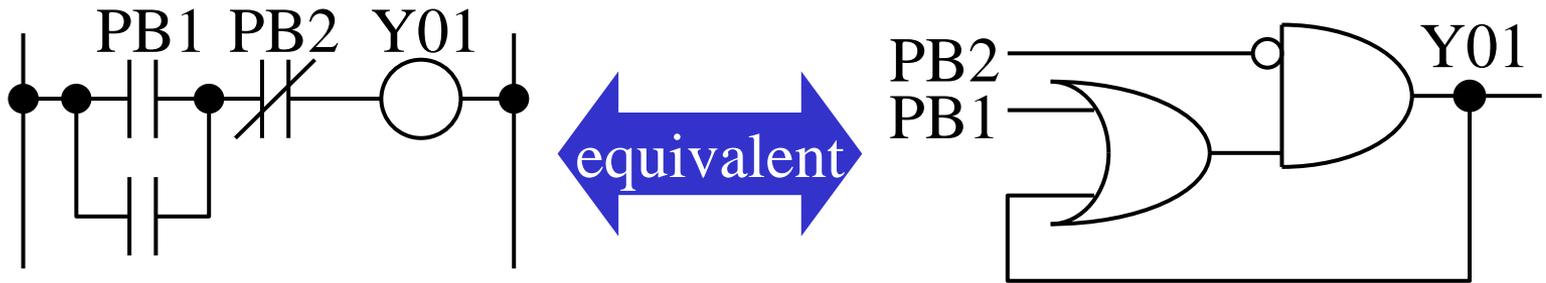
- “Legacy” systems are built with PLC
- Embedded processors are not fast enough
- Custom circuit is effective for hard real-time control
 - Automatic generation is desired
- Converting PLC program to hardware description

NOT for ALL systems



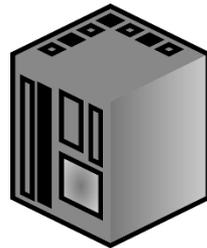
Goal

- To implement PLC programs with FPGA



LOD	PB1
OR	Y01
ANDN	PB2
OUT	Y01

PLC



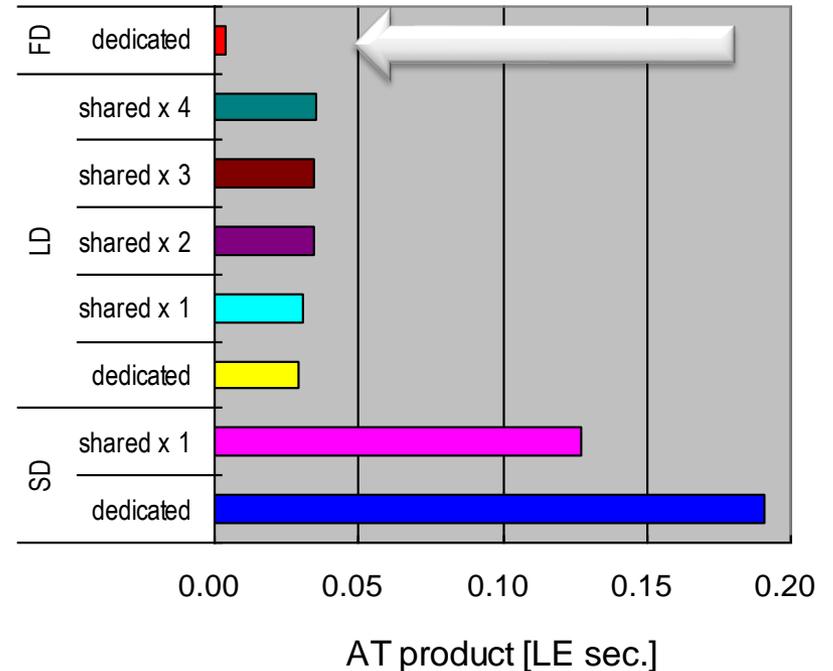
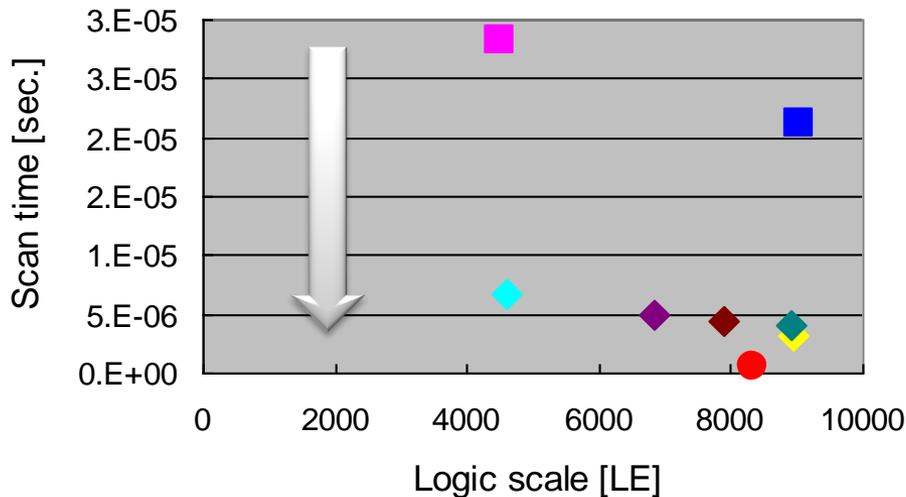
Small, fast, reliable,
low-cost, flexible,
resistant to
tampering and
duplication



Results

- Ladder program of productive industrial machinery
 - 165 instructions (incl. 6 add/sub, 12 multiplication, and 9 division)
- $T_{PLC} : T_{SD} = 76 : 1$, $T_{PLC} : T_{FD} = 3380 : 1$

- SD dedicated
- ◆ LD shared x 1
- ◆ LD shared x 2
- ◆ LD shared x 4
- SD shared x 1
- ◆ LD shared x 2
- ◆ LD shared x 3
- ◆ LD shared x 4
- FD dedicated
- ◆ LD dedicated
- ◆ LD shared x 3

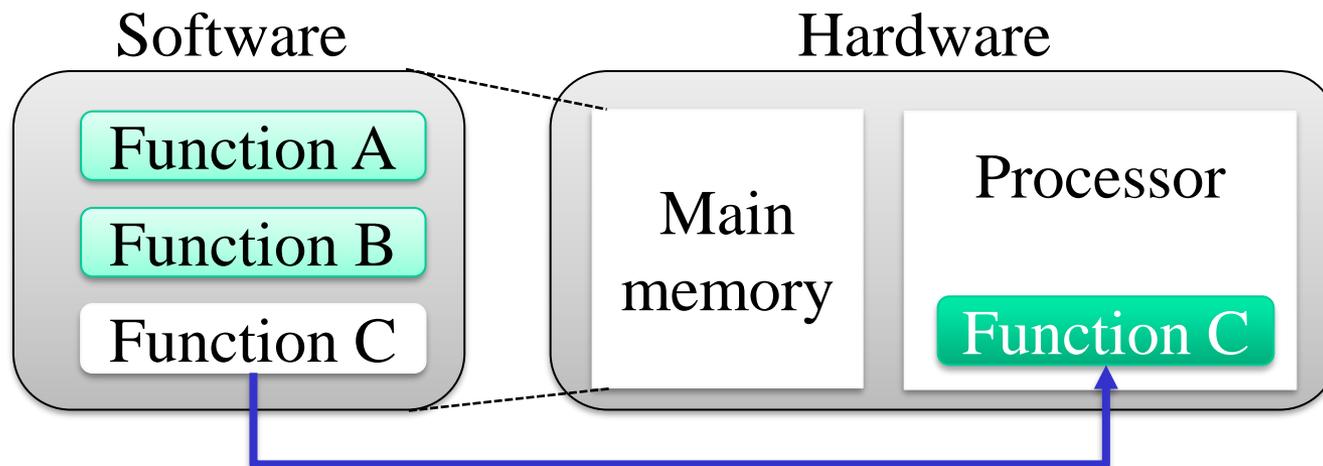


State of the art

- Design methodology
 - Old: home-made converter (PLC inst. → HDL)
 - Merits: fine control, good for evaluation
 - Problems: much effort, tool support
 - Now: C-based high-level synthesis
 - Merits: commercial support, co-design, integration
 - Problems: black box, difficult control
- Security
 - Obfuscation: Software, Hardware
 - Use of secure processor

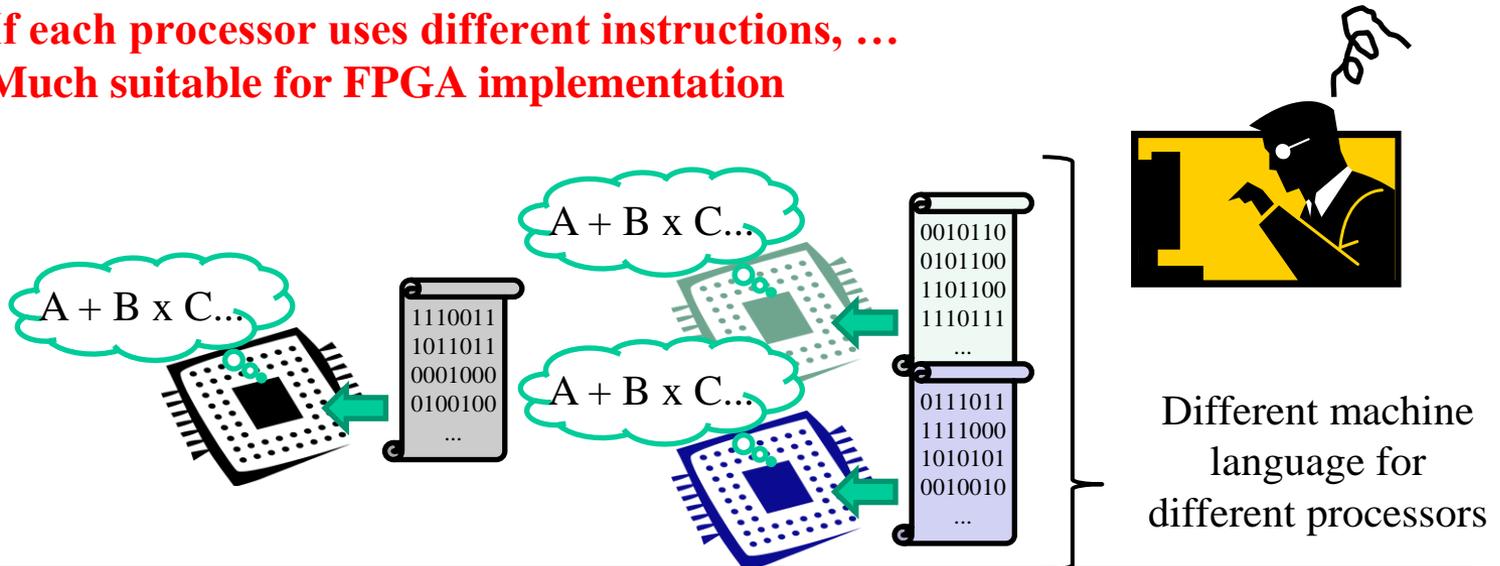
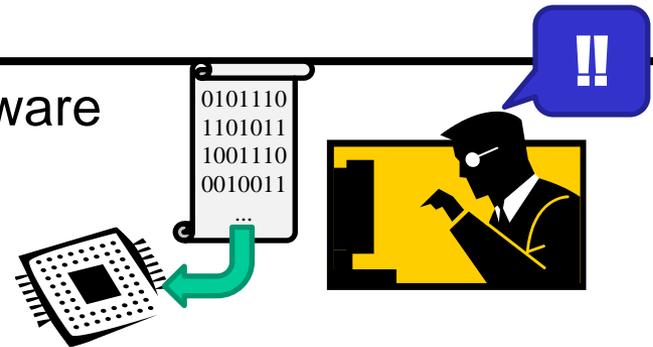
Processor specialization by high-level synthesis (HLS)

- HLS generates logic circuit from software (e.g., C language)
- Transform some part of software into hardware of processor
 - Soft-processor is written in hardware description language; easy to use
 - Tamper resistance: hardware is more difficult to analyze
 - Processor can be customized on case-by-case basis



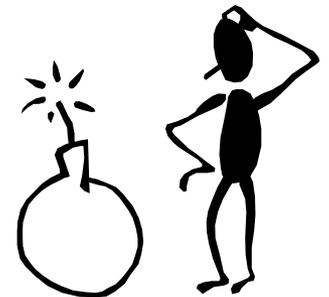
Secure Processor

- Analysis, plagiarism, tampering of software
 - Leakage of trade secret, piracy of software
- Secure processor
 - Supports software protection by hardware
 - Example: encryption of memory image
- **Diversification of processor**
 - **If each processor uses different instructions, ...**
 - **Much suitable for FPGA implementation**



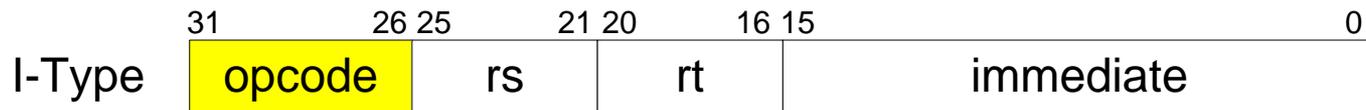
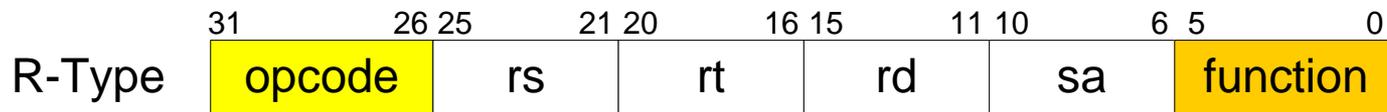
Instruction Set Randomization

- **If each processor has its unique ISA**
 - Binary software cannot be plagiarized.
 - Analysis of software would be difficult.
 - No unauthorized binary program
 - No injection attack (e.g., viruses)
- **If each processor has its unique ISA**
 - You have to design many processors
 - You have to prepare many tools
 - No portability of software



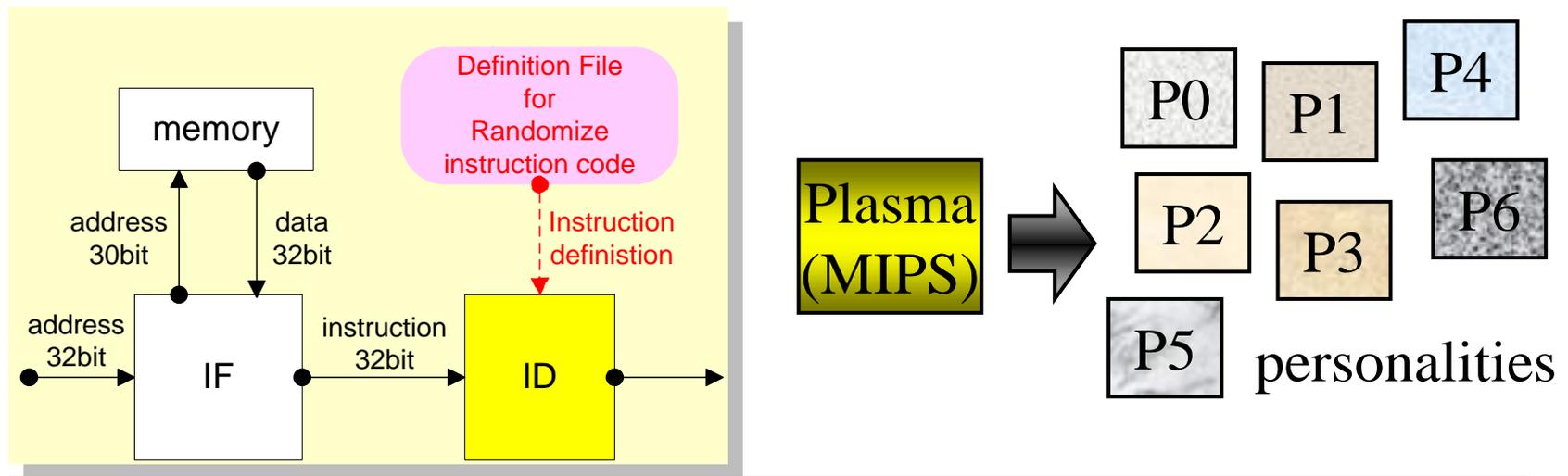
Example: MIPS ISA

- Use the same instruction format with different encoding
 - R-type: ADD (op=0, funct=32), SUB (op=0, funct=34)
 - I-type: BEQ (op=4), BNE (op=5), LW (op=35), SW (op =43)
 - J-type: J (op=2), JAL (op=3)



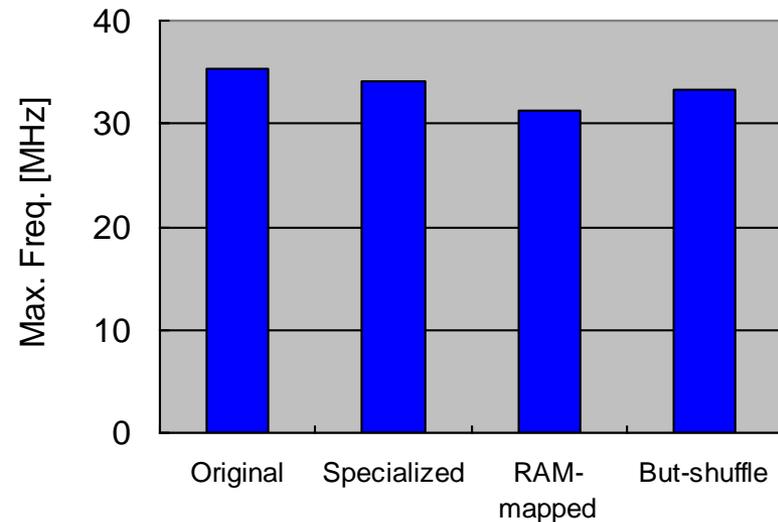
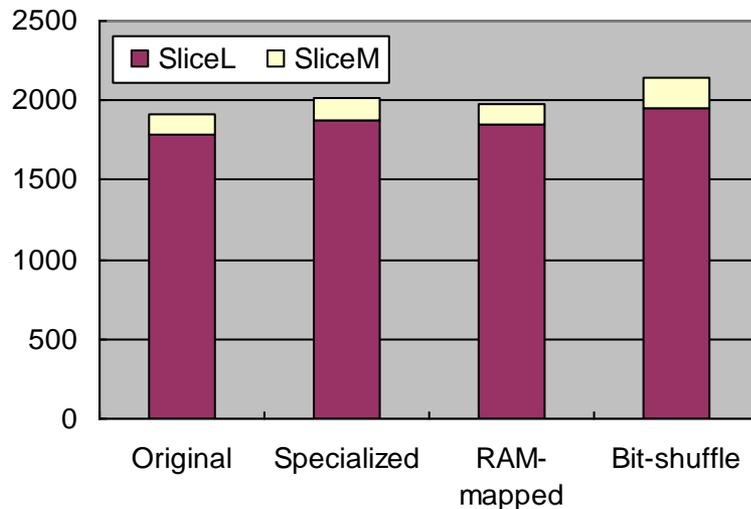
Personalization of ISA

- Use unique encoding with the same format
 - Use a separate definition file
 - Build personalized ID units
- Economical implementation of randomization



Evaluation

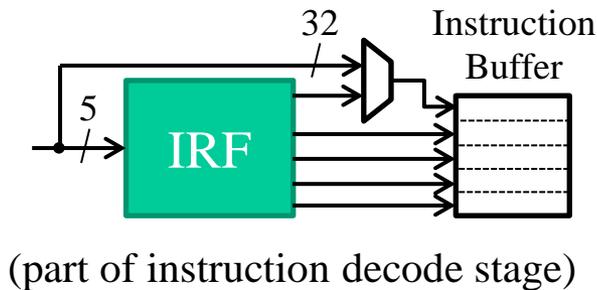
- Low overhead
 - Performance, Hardware resource
- High degree of freedom
 - Possible to generate enough number of products of different ISA.



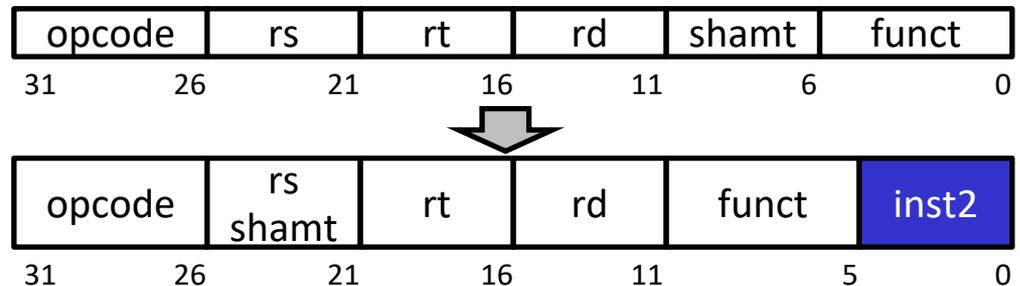
Instruction Register File (IRF)

- Small instruction storage accessed by indices of fetched instructions
 - originally used for *instruction compression*
 - can be applied for software obfuscation if the content of the IRF is hidden [2]

[2] D. Chang *et al.*: Program Differentiation, in *INTERACT-14 in conjunction with ASPLOS-XV*, No. 9 (2010).



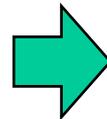
R-Type MIPS Instruction



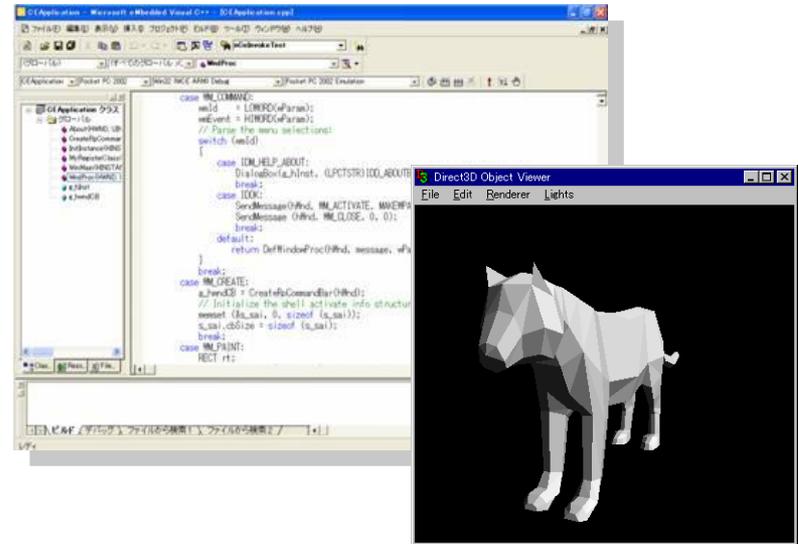
Information hiding, Steganography

- Embed information into copyrighted materials
 - E.g., software, 3D models

Embedding

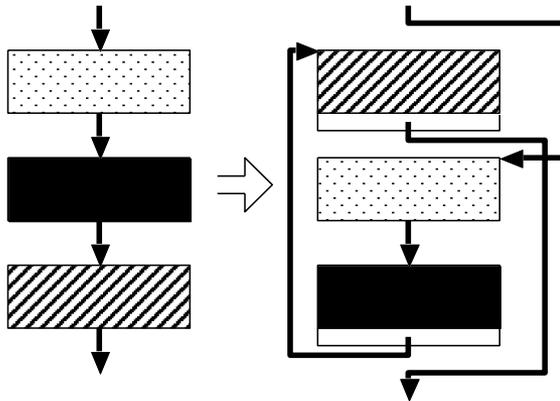


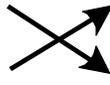
Avoid detection
Avoid deletion
Avoid modification



Freedom in Instruction Sequence

- How many expressions of a program exist?
- Freedom $f \rightarrow$ Information $\log_2 f$ (bit)
 - Equivalent instructions: $\text{sub } (r \leftarrow r - 1)$, $\text{add } (r \leftarrow r + (-1))$
 - The address of a basic block
 - The order of instructions in a basic block
 - The address of global variables
 - The allocation of registers to variables



(1) $ax := var1$		(2) $dx := var2$
(2) $dx := var2$		(1) $ax := var1$
(3) $ax := ax + dx$		(3) $ax := ax + dx$

(1) $ax := ax + bx$		(2) $dx := dx + cx$
(2) $dx := dx + cx$		(1) $ax := ax + bx$
(3) $ax := ax + dx$		(3) $ax := ax + dx$

Random Number Generator

(a key component for security)

- Many application utilizes “random numbers”
 - Simulations, games, ...
- True Random Number Generator (TRNG)
 - TRN is generated from various *physical phenomena*
→ unpredictable
 - Thermal noise, metastability, jitter, ...
 - Dedicated *hardware* is essential.
- Pseudo-Random Number Generator (PRNG)
 - Generated by a pre-determined algorithm and initial values
→ predictable
- Unpredictable RNG (URNG)
 - Between TRNG and PRNG, practically unpredictable.
 - Utilizes randomness (entropy) of the system



Ayumu Chiba, Shuichi Ichikawa: "Evaluation of Random Number Generator Utilizing Weather Data and LFSR," IEEJ Transactions on Industry Applications, vol. 143, no. 2, pp. 80--86 (2023).

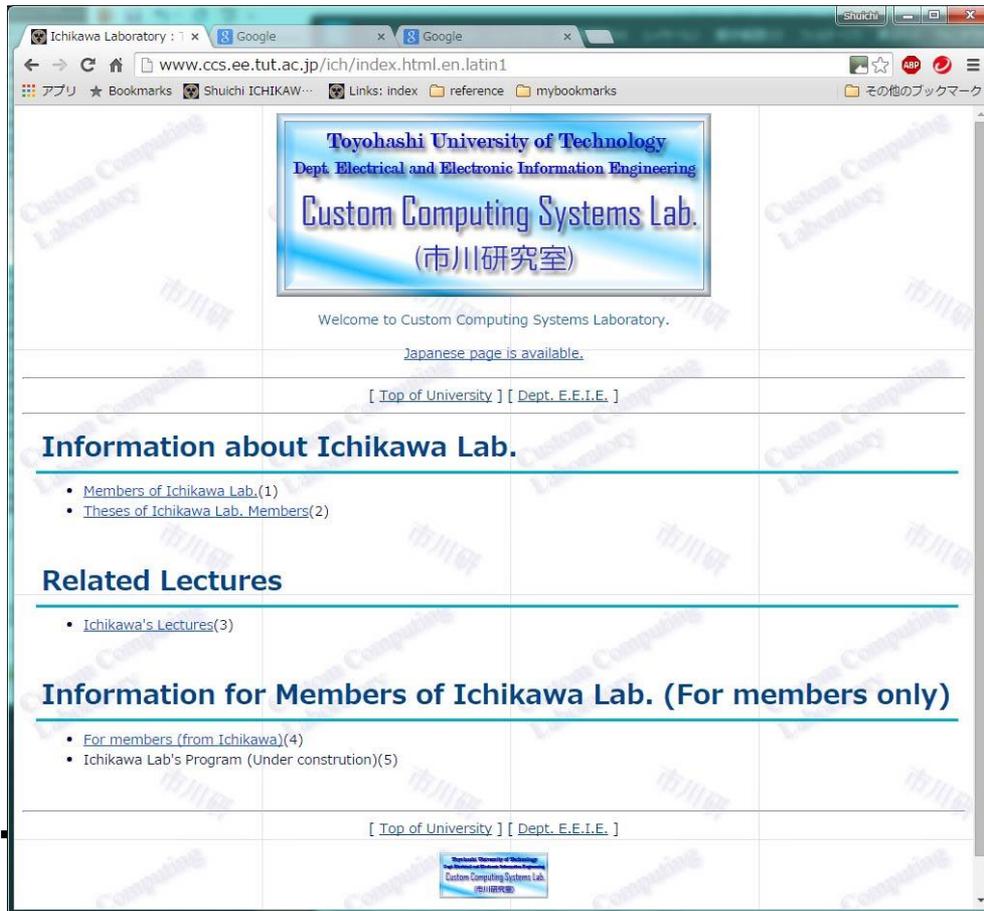
Shuichi Ichikawa: "Pseudo-Random Number Generation by Staggered Sampling of LFSR," Proc. Eleventh International Symposium on Computing and Networking (CANDAR 2023), pp. 134--140 (2023).

Hidetaka Masaoka, Shuichi Ichikawa, Naoki Fujieda: "Random Number Generation from Internal LFSR and Fluctuation of Sampling Interval," IEEJ Transactions on Industry Applications, vol. 141, no. 2, pp. 86--92 (2021). (in Japanese)

Hisashi Hata, Shuichi Ichikawa: "FPGA Implementation of Metastability-based True Random Number Generator," IEICE Transactions on Information and Systems, Vol. E95-D, No. 2, pp. 426--436 (2012).

Our web site

<http://www.ccs.ee.tut.ac.jp/ich/>



Information on

- Academic staffs
- Research themes
- Theses of past under-graduates and graduates