Modern DSP Architectures

Implementation of Digital Signal Processing

Sabih H. Gerez
University of Twente

OUTLINE

- Software-Defined Radio
- Parallel DSP solutions
- DSP platforms for software-defined radio

Acknowledgement

Some pictures in the next slides are from:
Kessler, C.W., "Compiling for VLIW DSPs", In: S.S. Bhattacharyya et al. (Eds.), *Handbook of Signal Processing Systems*, Springer Nature, pp. 979-1020, (2019).

SOFTWARE-DEFINED RADIO (1)

- Consider the techniques discussed in this course:
  - Fixed-point optimization
  - Multiplierless filters
  - CORDIC
- These techniques are especially useful when designing dedicated hardware with no programmability or a low-degree of programmability.
- Hardware can operate at relatively low frequencies such as 10 to 20 MHz.

SOFTWARE-DEFINED RADIO (2)

- One could also imagine one or more processors doing these calculations.
- Advantage: one can improve the design, adapt to standard changes just by software updates → *software-defined radio* (SDR).
- Goal: put analog-to-digital converter as close as possible to the antenna and perform digital processing in hardware.
- Example: GNU Radio.
- Disadvantage: area and power overhead.
- Problem: one processor may not be enough, parallelism is needed.

PARALLEL PROCESSING

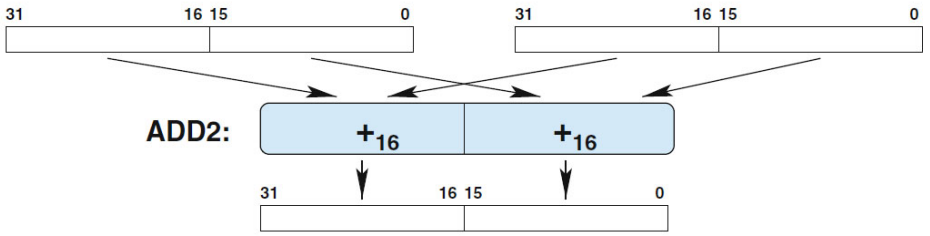
- Central question:
  - How to increase the performance?
- Increasing the clock frequency:
  - Leads to the generation of too much power, overheating, etc.
- Parallel processing is the solution:
  - Not only for computations
  - Also for data transport, memories, etc.

VECTOR PROCESSING, SIMD (1)

- One way to introduce parallelism without modifying too much a processor's architecture is to apply the same instruction to the multiple data:
- *Single Instruction Multiple Data* (SIMD)
- Also called: *vector processing*
- Think of computations that are repeated on multiple data and are mutually independent:
  - Taps in an FIR filter
  - Butterflies in the same stage of an FFT
  - Etc.

VECTOR PROCESSING, SIMD (2)

- Example: a 32-bit adder re-used as 2 16-bit adders.



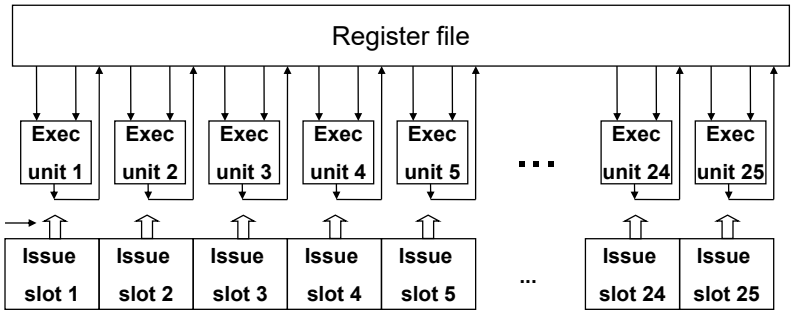
© Kessler 2019, Figure 11.

VERY-LARGE-INSTRUCTION WORD: VLIW (1)

- Multiple FUs, also called EXUs (execution units)
- Load-store architecture:
  - Communication with memory is always via register files.
  - Register files are multi-ported.
- Each FU can receive an instruction every clock cycle
- Each RISC instruction = one issue slot
- No dependencies between different RISC instructions
  - Orthogonal microcode
  - Compiler friendly
- One instruction = many RISC instructions © Jef van Meerbergen (TUE/Philips)

VLIW (2)

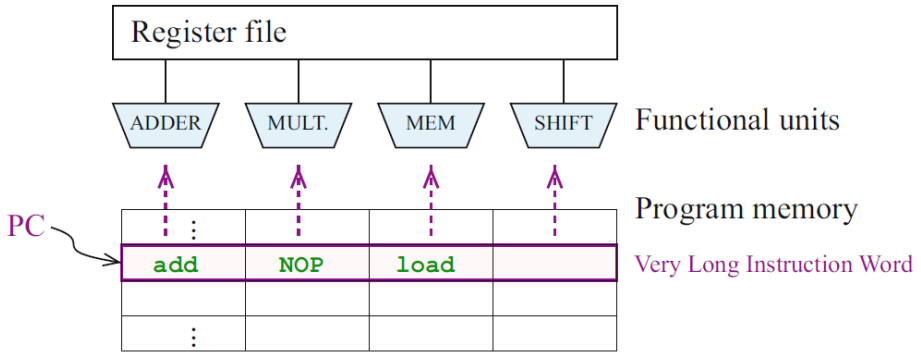
- Example: PHILIPS/NXP TRIMEDIA



- Assume 128 registers → 7 bits address
- Long instruction words e.g. (3*7+4)*25=625 bits
- Many ports on the register file e.g. 75

© Jef van Meerbergen (TUE/Philips)

VLIW (3)

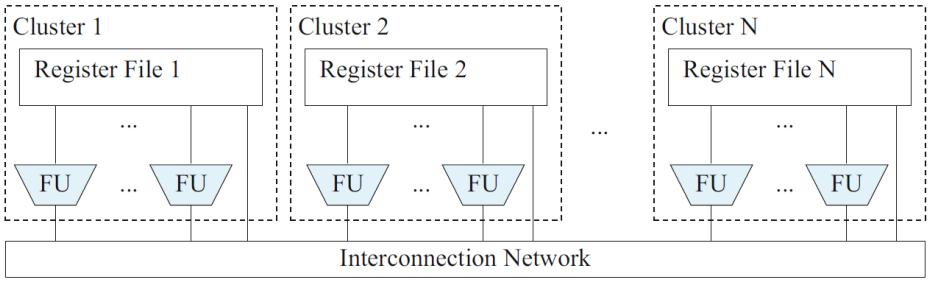


© Kessler 2019, Figure 1.

CLUSTERED VLIW (1)

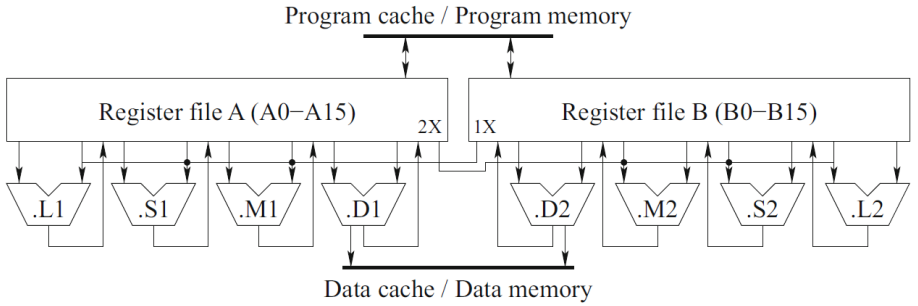
- VLIW has high demands on the design of the register file when each FU should be able to read any register.
- *Clustered VLIW* alleviates this issue.
- Registers within a cluster remain fully accessible by all FUs
- There is limited exchange of data between clusters:
  - Typically, one item per clock cycle per cluster

CLUSTERED VLIW (2)



© Kessler 2019, Figure 5.

CLUSTERED VLIW EXAMPLE: TI 'C6201



.L = logic unit; .M = multiplier
.S = shift unit;
.D = data addressing unit

© Kessler 2019, Figure 9.

HARDWARE LOOPS

- Consider a for loop, with an index *i* incremented from 0 to *N*.
- Normally, next to the execution of the actual loop body:
  - *i* will need to be incremented
  - and compared to *N*.
- This can be a considerable overhead (when the actual loop body is relatively simple).
- So, hardware support for loops can make a major difference:
  - Incrementation (or decrementation) and comparison can be done in parallel to the actual loop body execution.

HARDWARE LOOP EXAMPLE

LR = loop register

```
        ...                          ...
        add  8192, R17 ; trip count in R17   repeat  2, 8192    ; loop count in LR
LOOPLBL:sub  R17, 1, R17             load  R15, LR, R18
        load  R15, R17, R18          store  R18, R16, LR
        store  R18, R16, R17         ...
        bnez  R17, LOOPLBL
NEXTLBL:...
```

2 instructions

© Kessler 2019, Figure 8.

MULTICORE PROCESSORS

- Chips consists of multiple full-fledged processors.
- Each of these can e.g. be SIMD.
- *Threads* are often the model of computation.
- A run-time scheduler dispatches threads across the cores
  - Cores may be able to execute multiple threads simultaneously.

COARSE-GRAIN RECONFIGURABLE

- FPGAs are *fine-grain* reconfigurable:
  - One roughly builds digital systems by connecting bit-level building blocks such as AND and OR gates (actually, by configuring look-up tables and interconnections)
- *Coarse-grain reconfigurable* architectures have building blocks at the level of ALUs, multipliers, etc.
  - Proper configuration e.g. creates a data-path able to compute an entire FFT butterfly.

DSP FOR SOFTWARE-DEFINED RADIO

- Check the following paper:
  - Anjum, O, T. Ahonen, F. Garzia, J. Nurmi, C. Brunelli and H. Berg, *State-of-the-Art Baseband DSP Platforms for Software-Defined Radio: A Survey*, EURASIP Journal on Wireless Communication and Networking, Vol. 2011(5).
- The paper presents several ICs proposed for *software-defined radio* (SDR):
  - SDR: approach to realize radio functions (mixing, filtering, etc.) on processors.
- Check references in paper to really understand specific solutions.

SDR-PLATFORM CHARACTERISTICS

- Platforms are mixture of generic processors and dedicated co-processors (e.g. for LDPC decoding; LDPC = low-density parity check).
- Often also a mix of SIMD and VLIW.
- Next to DSPs a RISC-style processor is available for overall control and control-dominated parts of the processing.
- Programming such platforms is very complex and quite some effort is spent in compilers and other programming aids.