VLIW processors

E. Sanchez, M. Sonza Reorda

Politecnico di Torino

Dipartimento di Automatica e Informatica (DAUIN)

Torino - Italy

Introduction

- Superscalar processor evolution led to complex processors composed of several functional units, and all the logic for
  - detecting and managing dependencies
  - dynamic scheduling
  - branch prediction and speculation.
- This significantly increased the complexity of processors.
- Alternative approaches have also been investigated, including VLIWs.

VLIW processors

- *Long Instruction Word* (LIW) and *Very Long Instruction Word* (VLIW) processors have long instructions encoding several operations, which are issued in parallel.
- The hardware includes as many functional units, as the operations in a single instruction.
- VLIW processors are relatively common for embedded applications (in particular for signal processing and graphics).

Basic characteristics

- More complex software
  - It is up to the compiler to decide which instructions to pack together, exploiting parallelism, unrolling loops, scheduling code across basic blocks, etc.
- Simpler hardware
  - In a VLIW processor, the hardware does not perform any check on possible dependencies among instructions (this task is completely left to the compiler). This significantly simplifies the processor.

Stalls

When an operation requires stalling (e.g., due to a cache miss) the whole instruction packet is stalled, in order to preserve the flow decided by the compiler.

Example
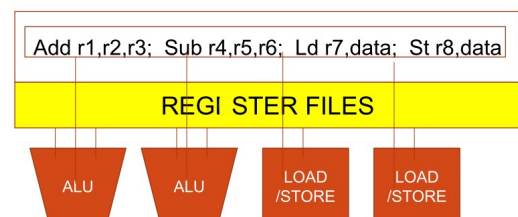
Example

- Consider a VLIW processor that in every clock cycle can issue:
  - two memory references
  - two FP operations
  - one integer operation or branch.
- Which is the processor behavior on the following loop-based example?
- The branch delay slot is not considered.

Example

```
Loop:    L.D        F0, O(R1)
         ADD.D   F4, F0, F2
         S.D        F4, O(R1)
         DADDI   R1, R1, #-8
         BNE       R1, R2, Loop
```

Solution

Memory reference 1	Memory reference 2	FP operation 1	FP operation 2	Integer operation/branch
L.D F0,0(R1)	L.D F6,-8(R1)			
L.D F10,-16(R1)	L.D F14,-24(R1)			
L.D F18,-32(R1)	L.D F22,-40(R1)	ADD.D F4,F0,F2	ADD.D F8,F6,F2	
L.D F26,-48(R1)		ADD.D F12,F10,F2	ADD.D F16,F14,F2	
		ADD.D F20,F18,F2	ADD.D F24,F22,F2	
S.D F4,0(R1)	S.D F8,-8(R1)	ADD.D F28,F26,F2		
S.D F12,-16(R1)	S.D F16,-24(R1)			DADDUI R1,R1,#-56
S.D F20,24(R1)	S.D F24,16(R1)			
S.D F28,8(R1)				BNE R1,R2,Loop

Solution

Memory reference 1	Memory reference 2
L.D F0,0(R1)	L.D F6
L.D F10,-16(R1)	
L.D F18,-32(R1)	L.D F22,-40
L.D F26,-48(R1)	
S.D F4,0(R1)	S.D F8,-8(R1)
S.D F12,-16(R1)	S.D F16,-24(
S.D F20,24(R1)	S.D F24,16(R
S.D F28,8(R1)	

The code containing 7 iterations is executed in 9 clock cycles, corresponding to 1.29 cycles per vector element.

In the average, 2.5 operations per clock cycle are executed.

The efficiency (percentage of available slots containing an operation) is about 50%.

The number of required FP registers is 15, compared with 3 for the normal pipelined processor, 5 for the unrolled version, and 6 for the superscalar one.

Instruction size

- In our example we have 5 functional units.
- To control each functional unit we need 16 to 24 bits.
- Therefore, the length of an instruction is from 80 to 120 bits.

Advantages

A VLIW processor does not require the hardware for choosing the instructions to be executed in parallel (i.e., for detecting dependencies).

Limitations

The performance that can be attained by a multiple-issue processor is limited by

- inherent limitations of ILP in programs
- difficulties in building the hardware
- limitations specific to a superscalar or VLIW processor.

Limitations in the inherent ILP

- It is difficult to find a sufficient number of independent instructions that can be executed in parallel.
- This is even more difficult due to pipelined functional units, having a latency greater than 1.
- In general, to avoid any stall we need to find a number of independent operations roughly equal to
  average pipeline depth   number of functional units.
- With 5 functional units, and an average pipeline depth of 4 clock cycles, we need as many as 20 independent operations to avoid stalls.

Hardware cost

- By increasing the number of functional units, there is also an increase in the bandwidth towards the register file and the memory.
- This means increasing the hardware complexity and possibly decreasing the performance.
- Possible solutions are
  - memory interleaving
  - multiport memories
  - multiple access per clock cycle memories.

Code size

- It is much larger for VLIW processors, mainly due to two factors:
  - loops are intensively unrolled to extract more parallelism
  - the empty slots in instruction encoding.
- Sometimes instructions are compressed in memory and expanded when loaded to the processor.

Memory access

In VLIW processors, access to memory is often a bottleneck, since the required bandwidth is higher, and stalls due to cache misses cause a stall on the whole processor (all functional units are synchronized by the execution order decided by the compiler).

Binary Code Compatibility

- Any change in the implementation of a VLIW processor (e.g., changing the latency of a functional unit) requires recompiling the code.
- This is a major disadvantage with respect to superscalar processors, which can easily be made binary compatible with their previous versions.
- Object code translation or emulation will possibly solve this problem in the future.

Examples

- Trimedia TM32
- Transmeta Crusoe.

Trimedia TM32

- It is intended for media applications (e.g., MPEG compression and decompression).
- It was developed by NXP (formerly Philips Semiconductors).
- Every instruction contains five operations.
- The processor
  - is completely statically scheduled
  - does not detect hazards.

Transmeta Crusoe

- It was intended for the low-power market (mobile PCs and mobile Internet appliances).
- It guarantees instruction set compatibility with the x86 instruction set (a software system translates from the x86 instruction set to the Crusoe one).
- Instructions come in two sizes:
  - 64 bits (2 operations)
  - 128 bits (4 operations).
- The processor includes:
  - a 6-stage pipeline for integer instructions
  - a 10-stages pipeline for FP instructions.

Other VLIW architectures

- SHARC (by Analog Devices)
- C6000 (by TI)
- ST200 (by STMicroelectronics).

EPIC

- EPIC (*Explicitly Parallel Instruction Computing*) is the definition for the architecture of some HP and Intel processors introduced in late 90s, such as Itanium.
- It combines ideas from VLIWs with a higher degree of flexibility, like in superscalar processors.
- The EPIC architecture got some success in the area of high-end processors (e.g., for servers).

Classification (I)

- Processing instructions in parallel requires three major tasks:
  - checking dependencies between instructions to determine which instructions can be grouped together for parallel execution
  - assigning instructions to the functional units in the hardware
  - determining when instructions are initiated together.
- These three tasks can each be assigned to the compiler or to the processor.

Classification (II)

	Grouping	Fn unit asgn	Initiation
Superscalar	Hardware	Hardware	Hardware
EPIC	Compiler	Hardware	Hardware
Dynamic VLIW	Compiler	Compiler	Hardware
VLIW	Compiler	Compiler	Compiler

Advantages of VLIW

- Dependencies are determined by compiler and used to schedule according to function unit latencies.
- Function units are assigned by compiler and correspond to the position within the instruction packet.
- Reduces hardware complexity.
- Tasks such as decoding, data dependency detection, instruction issues etc. become simple.
- Ensures potentially higher Clock Rate.
- Ensures lower power consumption

Disadvantages of VLIW

- Higher complexity of the compiler
- Compatibility across implementations: Compiler optimization needs to consider technology dependent parameters such as latencies and load-use time of cache.
- Unscheduled events (e.g., cache miss) stall entire processor.
- Code density: In case of un-filled opcodes in a VLIW, memory space and instruction bandwidth are wasted (low slot utilization).
- Code expansion: Causes high power consumption.

Applications

- VLIW architecture is suitable for Digital Signal Processing applications.
- Processing of media data like compression/decompression of Image and speech data.

Examples of VLIW processors

- VLIW Mini supercomputers:
  - Multiflow TRACE 7/300, 14/300, 28/300
  - Multiflow TRACE /500
  - Cydrome Cydra 5
  - IBM Yorktown VLIW Computer
- Single-Chip VLIW processors:
  - Intel iWarp, Philip's LIFE Chips
- Single-Chip VLIW Media processors:
  - Trimedia, Chromatic, Micro-Unity
- DSP Processors (TI TMS320C6x, STMicroelectronics ST200)