The LIFE Family of High Performance Single Chip VLIWs

Gerrit A. Slavenburg, Philips Research Palo Alto
Andrew S. Huang, Carnegie Mellon University[1]
Yen C. Lee, Philips Research Palo Alto

1. Department of Electrical and Computer Engineering

contents

Introduction

A 'Philips' micro-processor ?

- LIFE chips are *not currently* sold on the open market
- For Philips, this is but one of the many approaches to implement the complex processing needs of today's consumer and professional equipment

What is LIFE then ?

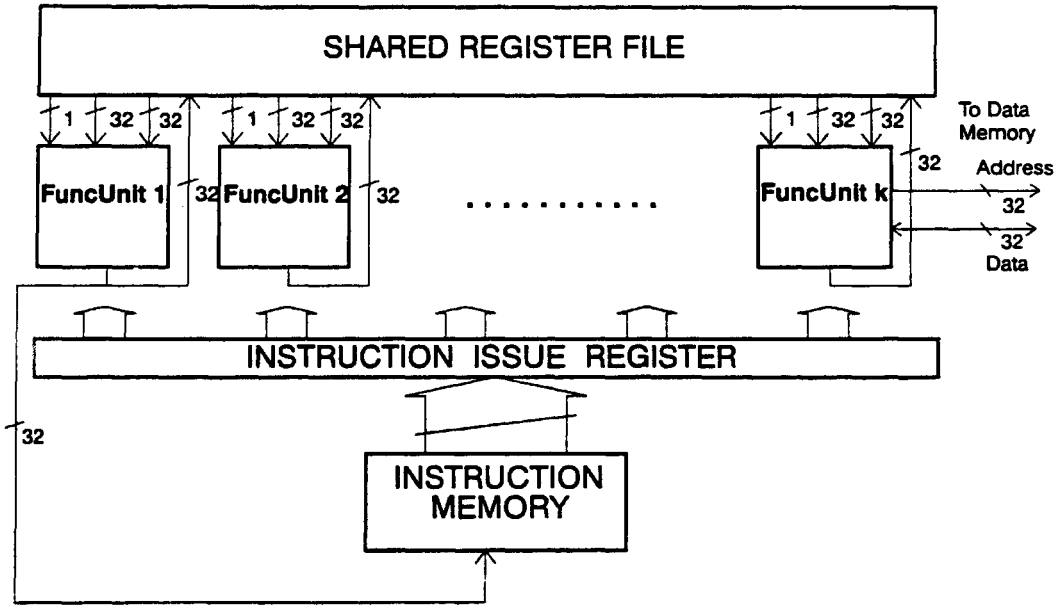
- A method for rapid design of custom chips
- Based on adding problem specific hardware to the heart of a high performance VLIW[2] CPU

This talk will:

- Compare the general purpose capabilities of LIFE with application specific performance of LIFE
- Show SPEC numbers and the elements of performance
- Convince you that VLIWs are *great* single chip CPU's

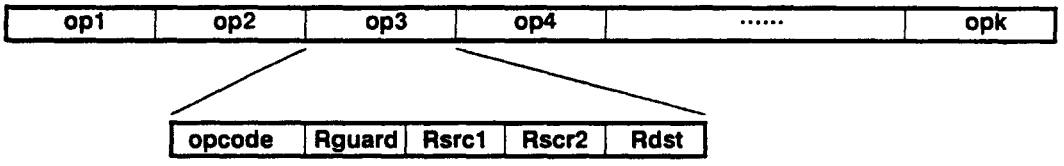
2. Very Long Instruction Word

General LIFE block diagram



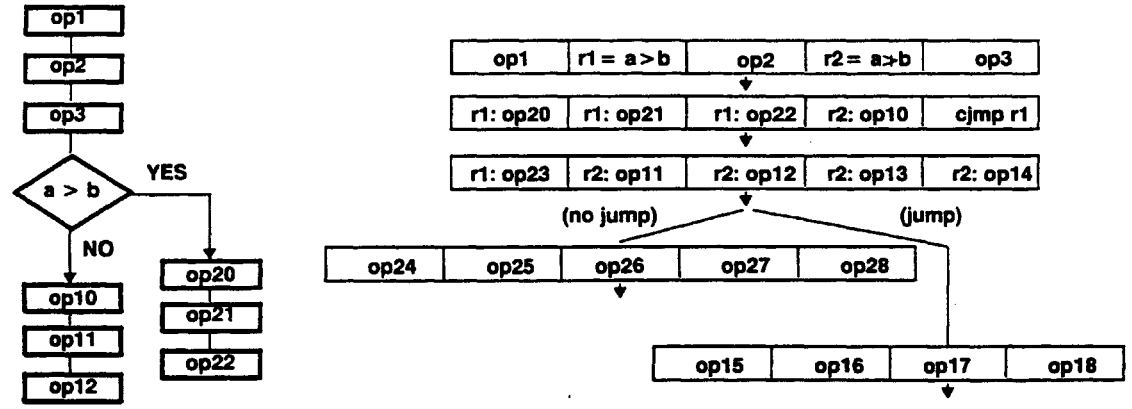
LIFE Execution model (general)

- ☆ multiple, pipelined *functional units*
- ☆ all functional units receive their arguments from and store their result to the central shared register file
- ☆ an *operation* is started on each unit each clock cycle
- ☆ external data memory access is implemented by one or more functional units that implement *load* and *store* operations
- ☆ flow of control is implemented by one or more functional units that implement (un)conditional jumps
- ☆ all this is controlled by a single 'Very Large Instruction Word' composed by a parallelizing compiler



LIFE Execution model (guarding)

- ☆ Guarded operations :
  The 'Rguard' field selects a boolean
  - if FALSE : suppress effect of operation
  - if TRUE : allow effect of operation
- ☆ Guarded operations are used to alleviate the effect of many branchdelay slots inherent in a VLIW with delayed branches



LIFE Execution model (speculation)

Speculative execution = the execution of operations from a basic block ahead of the evaluation of the condition that enables transfer of control to the basic block

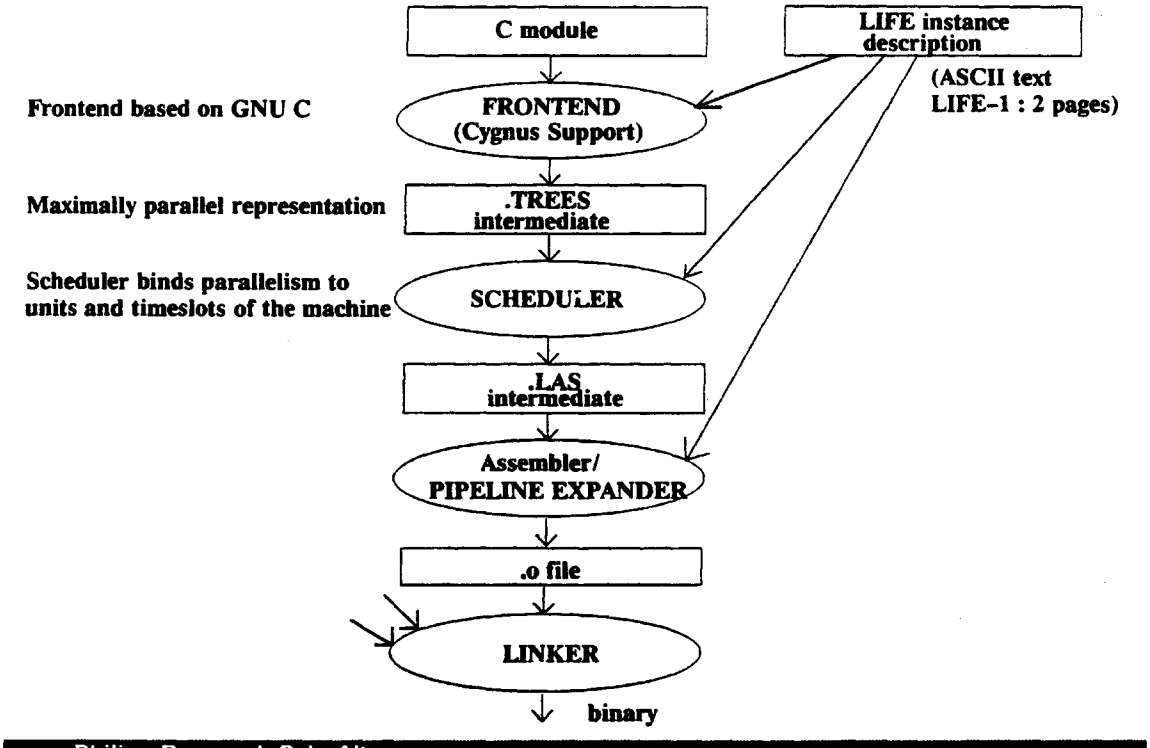
Speculative execution is used to enhance performance, and can be applied to Superscalar and VLIW architectures alike.

The current LIFE compiler performs:

- ☆ speculative loads
- ☆ speculative expression evaluation
- ☆ non speculative stores to registers and memory
- ☆ non speculative execution of operations that can raise exceptions (divide, IEEE floating point, etc. etc.)

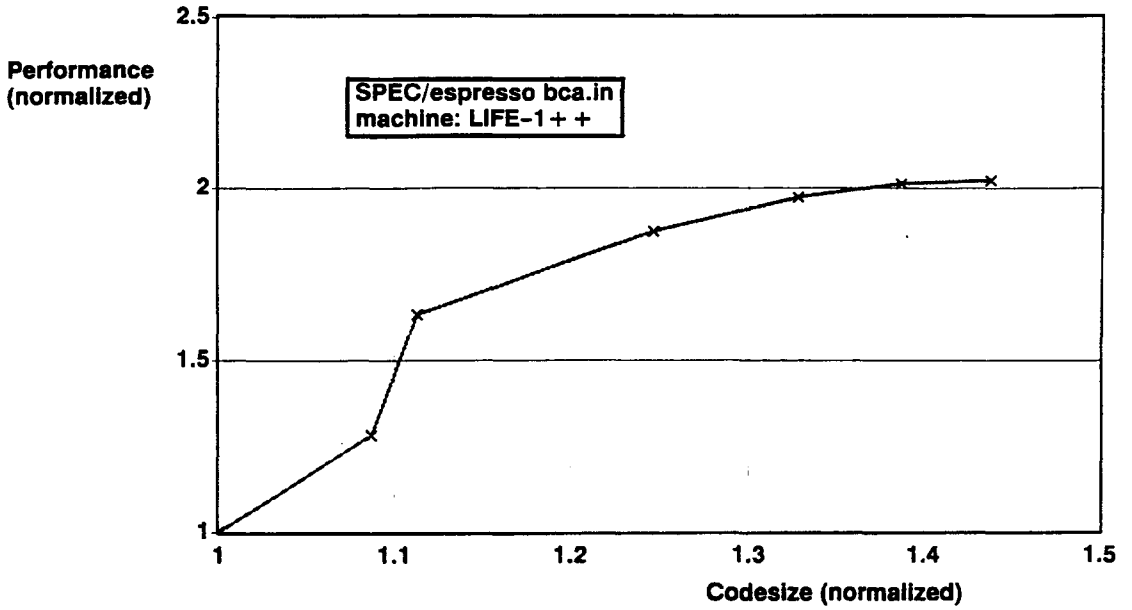
This is a simple model as compared to e.g. trace scheduling compilers for VLIWs, that may issue stores or exception raising operations early and may have to undo certain operations upon an erroneous prediction.

LIFE family compiler architecture

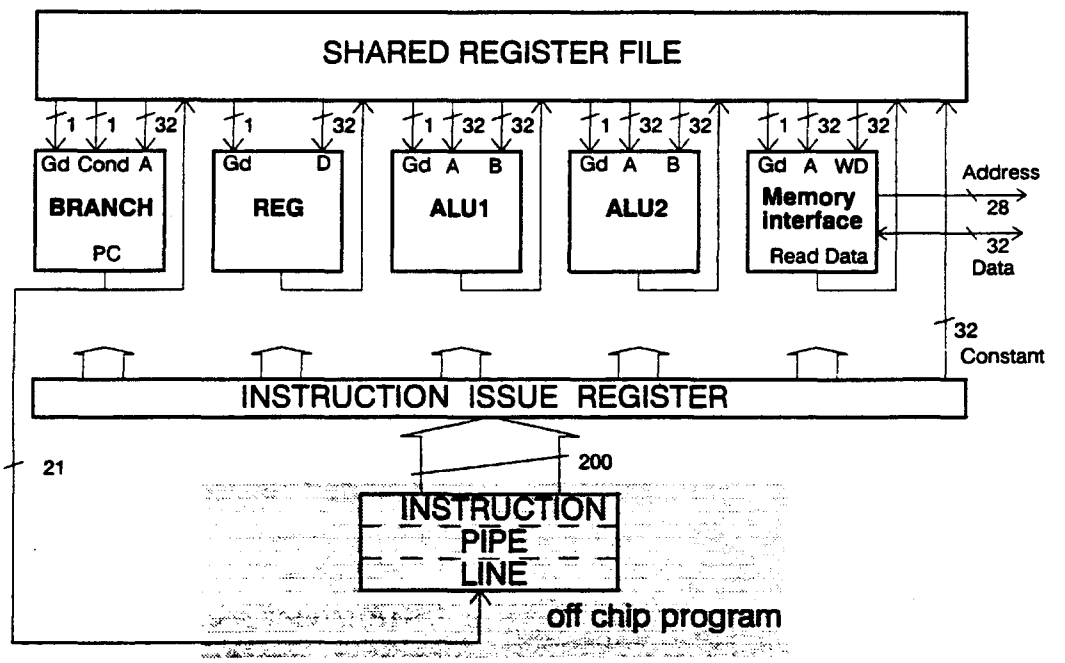


LIFE codesize vs. performance

The LIFE compiler frontend uses code-replication ('decision tree grafting') to enhance parallelism at the expense of codesize. This process currently requires hand control to pick the desired point in a graph like the one below.



LIFE-1 prototype chip (1989)



LIFE-1 operation set & latencies

LIFE-1 units provide the following operations:

CONST		**-2^{31} .. $2^{31}-1$**
ALU		
	Arithmetic [3]:	**ADD, SUB, NEG**
	Bit operations:	**AND, OR, XOR, INV**
	Shifts:	**ASR, ASL, LSR, LSL, ROL**
	Integer Comparison :	**EQL, NEQ, LES, LEQ, GTR, GEQ**
	Boolean:	**BEQL, BNEQ, BAND, BOR, BNOT, A and not B**
	Conversion:	**LBOOL, CBOOL**
REGISTER		**READREG(I), WRITEREG(I), I = 0..26**
MEMORY		**8/16/32 bit signed/unsigned LOAD, 8/16/32 bit STORE**
BRANCH		**JUMP, CJMPT, CJMPF**

Note the absence of unsigned compare, sign-extend, multibit-shift, integer multiply, divide, modulo and floating point operations.

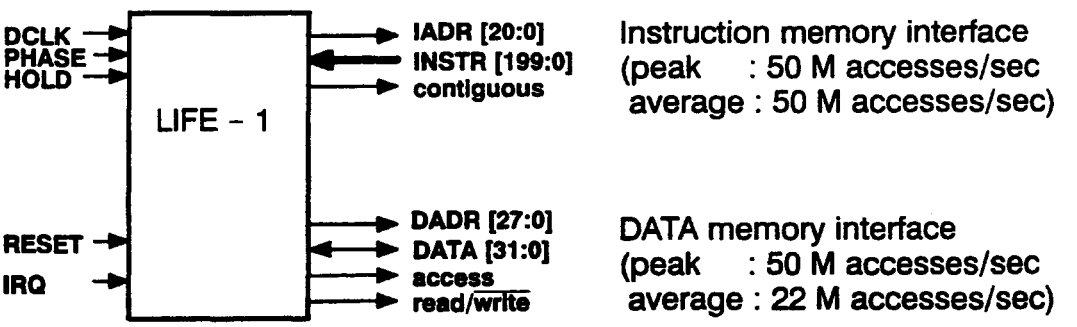
The LIFE-1 unit latencies (pipeline depths) are :

CONST 0 cycles, ALU 2 cycles, REGISTER 1 cycles
MEMORY 3 cycles (from address), BRANCH delay 5 cycles

3. available in both modulo arithmetic and overflow exception generating versions

LIFE-1 chip facts

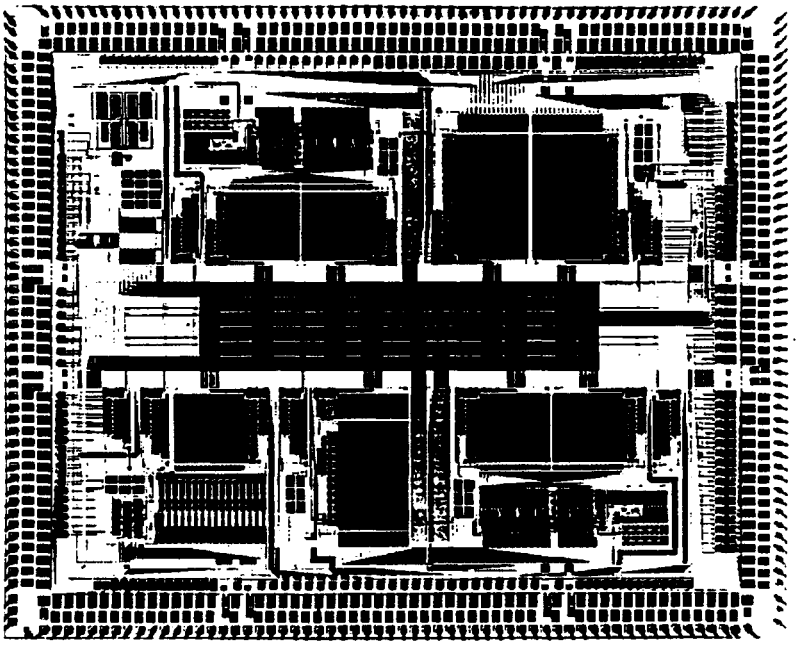
- ☆ 1.5 micron CMOS double metal (1989)
- ☆ 50 MHz clock and instruction issue rate
- ☆ 8 x 10 mm die size
- ☆ 77,000 transistors
  (no on chip caches, just a 6 unit VLIW CPU)
- ☆ 224 pads, of which 32 power/ground
- ☆ 1'st time right, working on prototype board



Instruction memory interface
(peak : 50 M accesses/sec
average : 50 M accesses/sec)

DATA memory interface
(peak : 50 M accesses/sec
average : 22 M accesses/sec)

LIFE-1 chip photograph



LIFE-1 SPEC performance

LIFE-1, when equipped with a 16 bit FPU, performs 8.8 x SPARCstation 2 on the 1D-DCT operation.

The SPEC ratio's for LIFE-1[4], SPARC and MIPS on 3 'C' benchmarks are:

SPEC ratio			
	LIFE-1 50 MHz	SPARC 40 MHz	MIPS 33 MHz
LI	37.2	23.2	32.1
EQNTOTT	25.9	21.5	24.7
ESPRESSO	28.8	19.0	26.3

Why are the SPEC ratio's less good for LIFE-1:

- SPEC Integer benchmarks have less fine-grain parallelism then typical LIFE application codes, and we are not allowed to change the sources
- With little fine-grain parallelism, LIFE-1's deep average pipeline results in a bad CPI (0.58 instead of 0.17 optimum)

4. Assuming a 3 cycle pipelined data memory and a 5 cycle pipelined instruction memory

LIFE-1 Elements of Performance

LIFE operations are '**finer-grained**' than RISC or VAX operations:

RISC: **LOAD R3, 14(R12)**
LIFE: **r1 = #14, r2 = R12, r3 = r1+r2, r4 =LOAD r3, R3=r4**

The table below compares the total number of instructions executed by RISC machines (averaged over SPARC, MIPS, IBM RS/6000, Motorola M88K) with the number of non-speculative operations executed by LIFE-1.

Total operations executed (in 10^9)				LIFE-1 stats	
	RISC	LIFE-1	Ratio (RISC/ LIFE)	issues/ cycle	non speculative issues/ cycle
LI	5.45	12.2	0.45	2.87	1.75
EQNTOTT	1.18	2.70	0.44	4.14	1.77
ESPRESSO	2.44	5.40	0.45	3.60	1.65
Geom. Average	2.50	5.62	0.44	3.50	1.72

For these benchmarks, 1 LIFE operation is equivalent to 0.44 RISC instructions. LIFE-1 is about as good as a RISC with 0.76 issues per cycle (CPI of 1.32).

LIFE-1++

LIFE-1++, a hypothetical 1991 technology redesign, illustrates these points

☆ **Shallow pipelines:**

	new latency	old latency	comment
CONST unit	**0**	**0**	
ALU units	**1**	**2**	
DATA MEM	**2**	**3**	
BRANCH delay	**1**	**5**	**(on chip I-cache)**
REGISTER unit	**1**	**1**	

☆ **Add more complex operations to enhance grainsize**
sign-extend, unsigned-compare, multibit-shifts

☆ **Add 1 CONST unit and 1 REG unit to utilize parallelism**

SPEC ratio (@ 50 MHz)			LIFE-1++ stats	
	LIFE-1	LIFE-1++	issues/ cycle	useful issues/ cycle
LI	37.2	63.5	4.26	2.89
EQNTOTT	25.9	58.2	6.52	3.21
ESPRESSO	28.8	52.4	5.30	2.88
Geom. average	30.3	57.9	5.28	2.99

Pushing it to the limit

The tables below shows the additional performance that can be attained by assuming a VLIW with many more units (but the same timing) and the additional benefit of a compiler with perfect alias analysis capability.

SPEC ratio's (@ 50 MHz)				
	LIFE-1	LIFE-1 + +	unlimited resources	perfect alias info
LI	37.2	63.5	82.7	94.9
EQNTOTT	25.9	58.2	87.1	87.4
ESPRESSO	28.8	52.4	73.3	77.7
Geom. Average	30.3	57.9	80.8	86.4

DCT performance (relative to 40 MHz SPARC)				
	LIFE-1	LIFE-1 + +	unlimited resources	perfect alias info
1D-DCT	8.8	9.1	11.1	20.7

The limit is ca. 1.6 VAX MIPS/MHz for the SPEC Integer benchmarks, but *much higher* for a realistic application.

Beyond that limit . ..

The current integer SPEC performance limit of 1.6 VAX MIPS/MHz can be raised by one or more of the following actions:

- ☆ improve the currently buggy code-replication heuristics....
- ☆ improve the function inlining heuristics
- ☆ enhance the operation grainsize (2x potentially possible)
- ☆ use a more complex compile/execution model with more agressive reordering (not quantified, highly depends on branch probability distribution)

Conclusions

- Single chip VLIWs provide competitive VAX MIPS/MHz and competitive clockrates, even when using only a simple runtime model
- The general purpose compute performance of the early LIFE family members is comparable to state-of-the-art general purpose RISC and SuperScalar CPU's
- The special purpose compute performance of the LIFE family members is excellent
- The ability to integrate special purpose functional units into a high performance CPU core makes the LIFE approach very attractive for the design of algorithmically non-trivial systems