

# Unleashing the power of SIMDization

Ayal Zaks


IBM Haifa Research Lab

[zaks@il.ibm.com](mailto:zaks@il.ibm.com)

[http://www.haifa.il.ibm.com/dept/svt/code\\_compiler.html](http://www.haifa.il.ibm.com/dept/svt/code_compiler.html)



## SIMD/short vector registers – one form of ubiquitous parallelism

- ◆ *“Extracting concurrency is not a big problem*
  - ◆ *a good compiler can extract concurrency from sequential code to the instruction level if required”, Chris Jesshope, yesterday*
  - ◆  *GCC is a good compiler*
  
- ◆ *“The problem is in expressing, distributing and scheduling that concurrency*
  - ◆ *it is because compilers must express concurrency with ad-hoc constructs and perform distribution and scheduling statically that this becomes difficult “*



## Extracting SIMD Concurrency – Where?

1. Traditionally – across iterations of an innermost do-loop
2. Within an iteration of an innermost loop  
(e.g. programmer unrolled loop)
3. Across iterations of an outer loop  
(i.e. unroll-and-jam followed by 2 above. VIRAM, Tenllado et al.)
4. In straight-line code  
(aka “SLP”, Larsen and Amarasinghe, PLDI '00)
5. in the presence of control flow  
(using if-conversion. Chen, Shin, Hall, CGO '05)



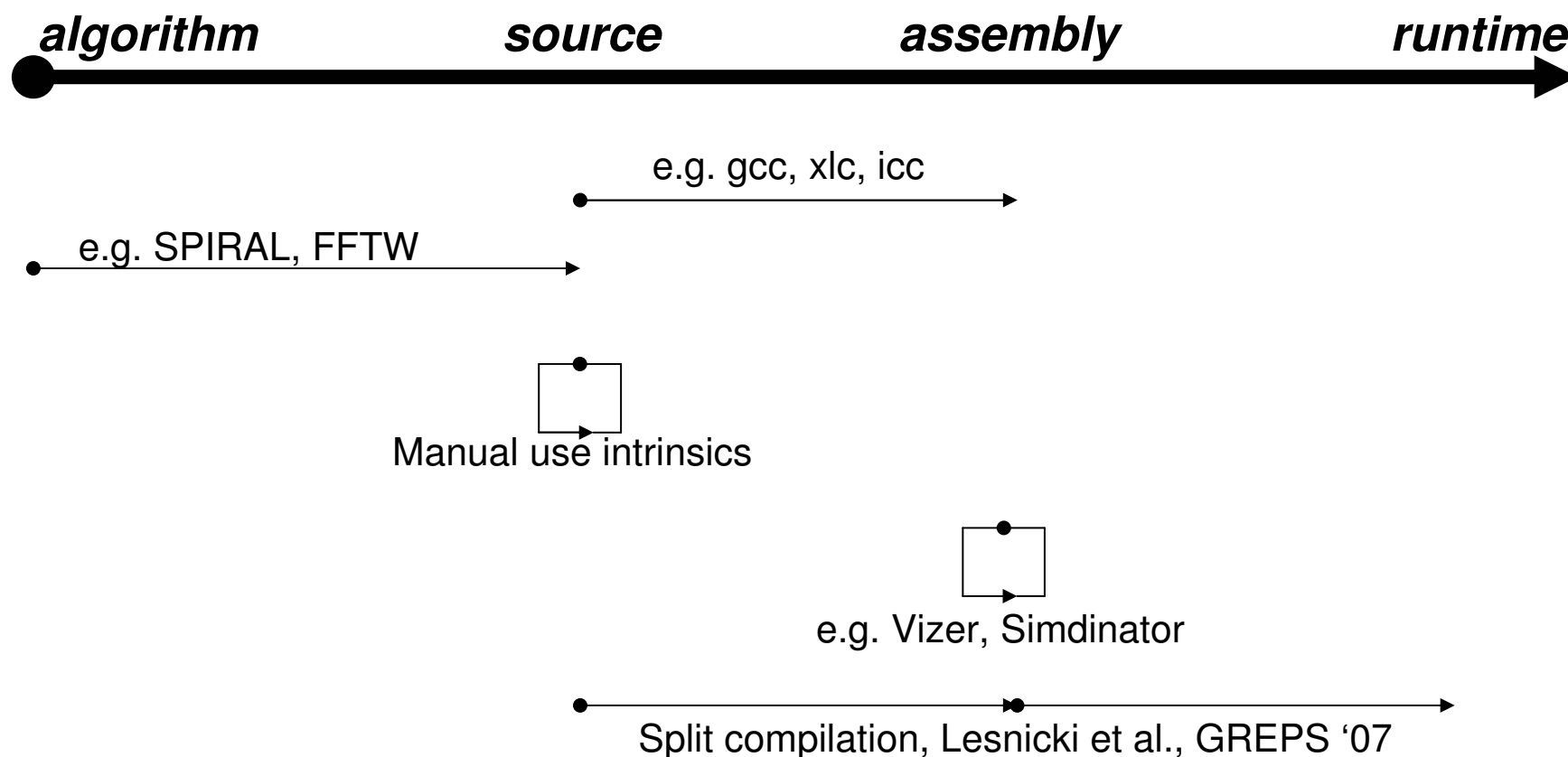
## Programming for SIMD

### ◇ Vector types and intrinsics

(link to [http://developer.apple.com/documentation/DeveloperTools/gcc-4.0.1/gcc/PowerPC-AltiVec-Built\\_002din-Functions.html](http://developer.apple.com/documentation/DeveloperTools/gcc-4.0.1/gcc/PowerPC-AltiVec-Built_002din-Functions.html))



## Extracting SIMD Concurrency – When?





## The problem is in ... feeding the beast

- ◇ With aligned data\*
  - ◇ Optimize permutations  
(Eichenberger, Wu, O'Brien, Wang, Zhao, PLDI '04, ICS'05, CGO '05)  
(Ren, Wu, Padua, PLDI '06)  
(Fireman, Petrank, Z., CC '07)
- ◇ With packed data
  - ◇ Handle non unit stride accesses (Nuzman, Rosen, Z., PLDI '06)
- ◇ With reuse of data
  - ◇ (Shin, Chame, Hall, PACT '02)
- ◇ Or else let it sleep
  - ◇ Cost model

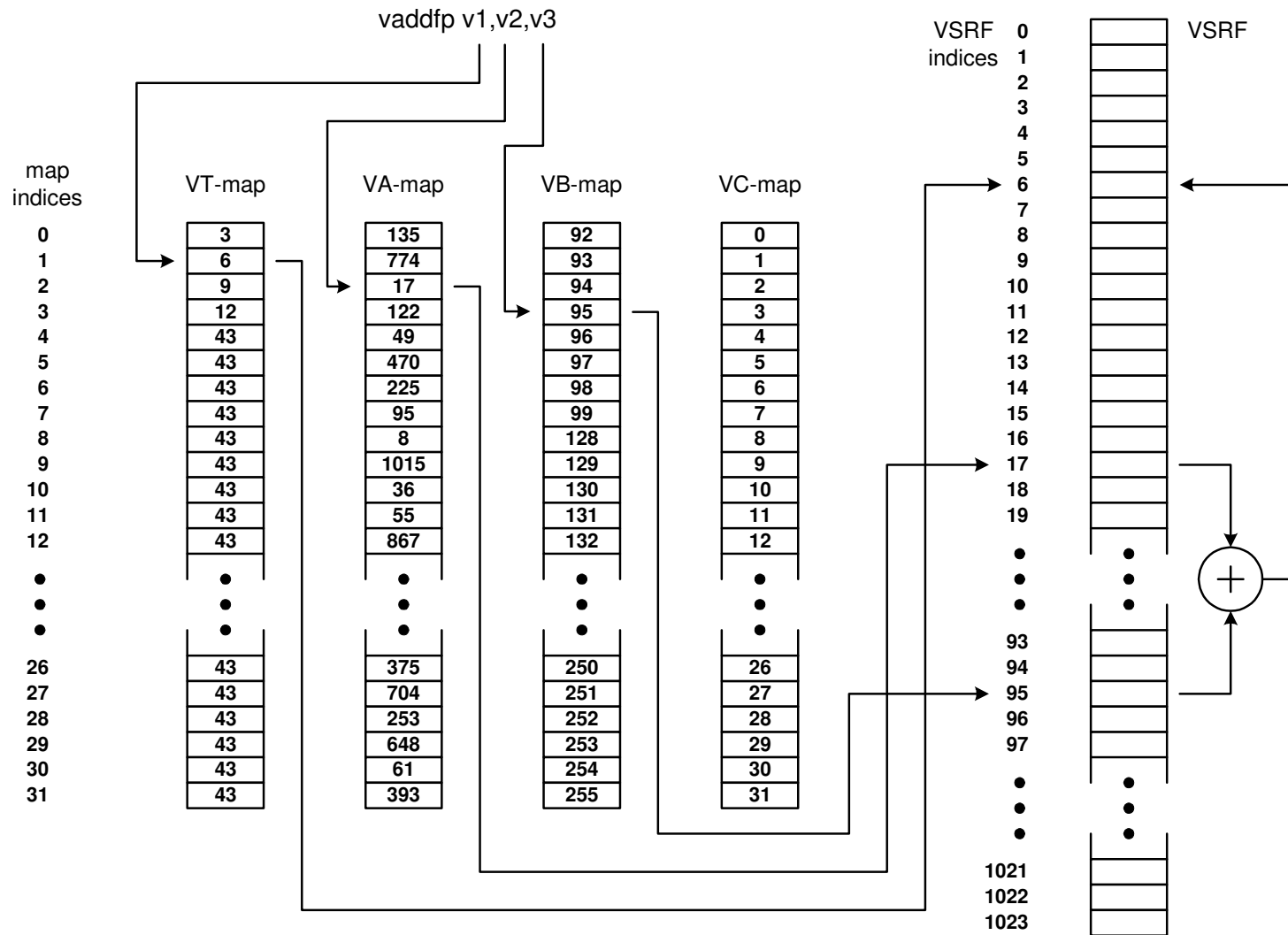


## Reuse of data ... need more vector registers

- ◇ than the standard 32 architected registers
- ◇ 128 as in Cell BE and VMX128
- ◇ 256 (16x16) as in MOM
  - ◇ (Matrix Oriented Multimedia. Corbal, Espasa, Valero, Supercomp. '99)
- ◇ 64 - 4096 as in eLite
  - ◇ (SIMdD – disjoint data. Moreno et al. IBM JR&D '03, CASES '03)
- ◇ 1024 - 4096 as in iVMX
  - ◇ (indirect VMX. Derby, Montoye, Moreira, CF'06, Namolaru, GREPS'07)



# iVMX Indirection Mechanism – An Example





## Summary

- ◆ SIMD parallelism is ubiquitous (Nuzman and Henderson, CGO '06)
- ◆ SIMDization as compiler optimization is becoming ubiquitous
- ◆ Can program directly reasonably easy if you insist (not portably)
- ◆ Efficiency problems mainly in providing data; further opportunities explored for data reuse
- ◆ For more information:
  - ◆ What's done: <http://gcc.gnu.org/projects/tree-ssa/vectorization.html>
  - ◆ What's to do: <http://gcc.gnu.org/wiki/VectorizationTasks>

Thanks to Dorit Nuzman, Ira Rosen,  
and many others from the GCC community