Experiences with High-Level Programming of FPGAs on Cray XD1

Thomas Steinke

Alexander Reinefeld, Thorsten Schütt

CUG 2006, May, Lugano

Thomas Steinke
Zuse Institute Berlin (ZIB) <www.zib.de>
Berlin Center for Genom Based Bioinformatics (BCB) <www.bcbio.de>
steinke@zib.de

Outline

- Why we are interested in using FPGAs?
  - HPC application areas at ZIB

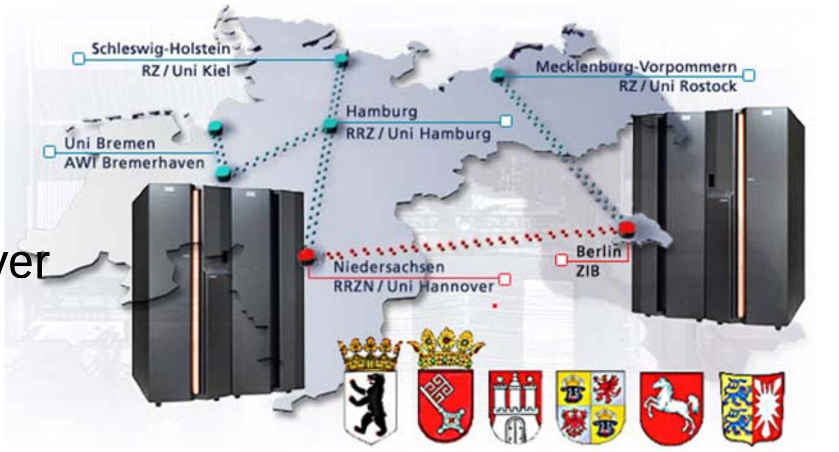
- FPGA programming with Mitrion-C
  1. learning phase: bandwidth measurements
  2. Lennard-Jones (LJ) potential energy & forces
     - add Fortran support
  - preliminary results

- lessons learned and outlook

Scaling to New Heights
CUG 2006

About the Zuse-Institut Berlin (ZIB)

- North-German High-Performance Computer Consortium
  - HLRN/2 at ZIB ...
    - 16 x IBM p690
    - 64 ... 256 GB RAM per node
    - dedicated fibre optics to 250 km apart peer-system Hanover
  - single virtual supercomputer

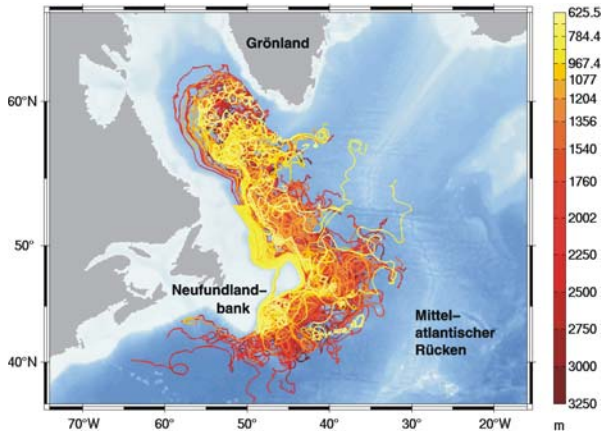


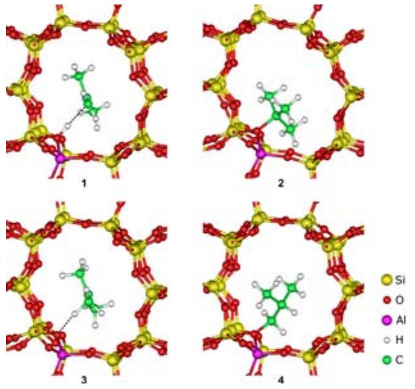
- **Cray XD1 with attached FPGAs**
  - 6x: dual 2.2 GHz Opteron blades, 2 GB RAM, Xilinx XC2VP50-7
  - Jan 2006: working with Mitrion-C

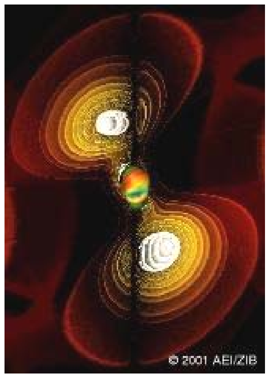
Scaling to New Heights
CUG 2006

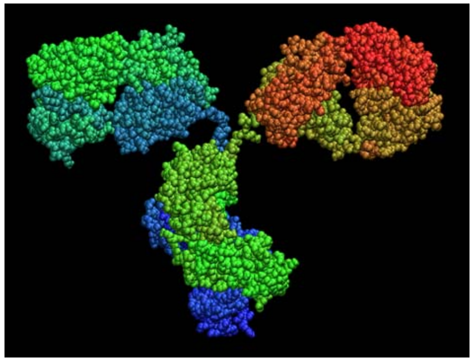
HPC Applications at ZIB

- physics
- mechanics
- environmental
- chemistry
  - molecular simulations
  - CP-MD, CI, DFT, MD/HMC
    - kernels: FFT, BLAS3, LJ









Why FPGAs?

- power consumption of standard µP systems
  - HLRN/2 at ZIB: EUR 250,000 per year for power consumption

- the ~15% performance “barrier” in cache-based µP architectures

→ hardware can be configured for a specific application

Some Issues to be Considered

- programming of FPGAs
  - We have no HW engineering background!
    - no VHDL

  → HLL approaches

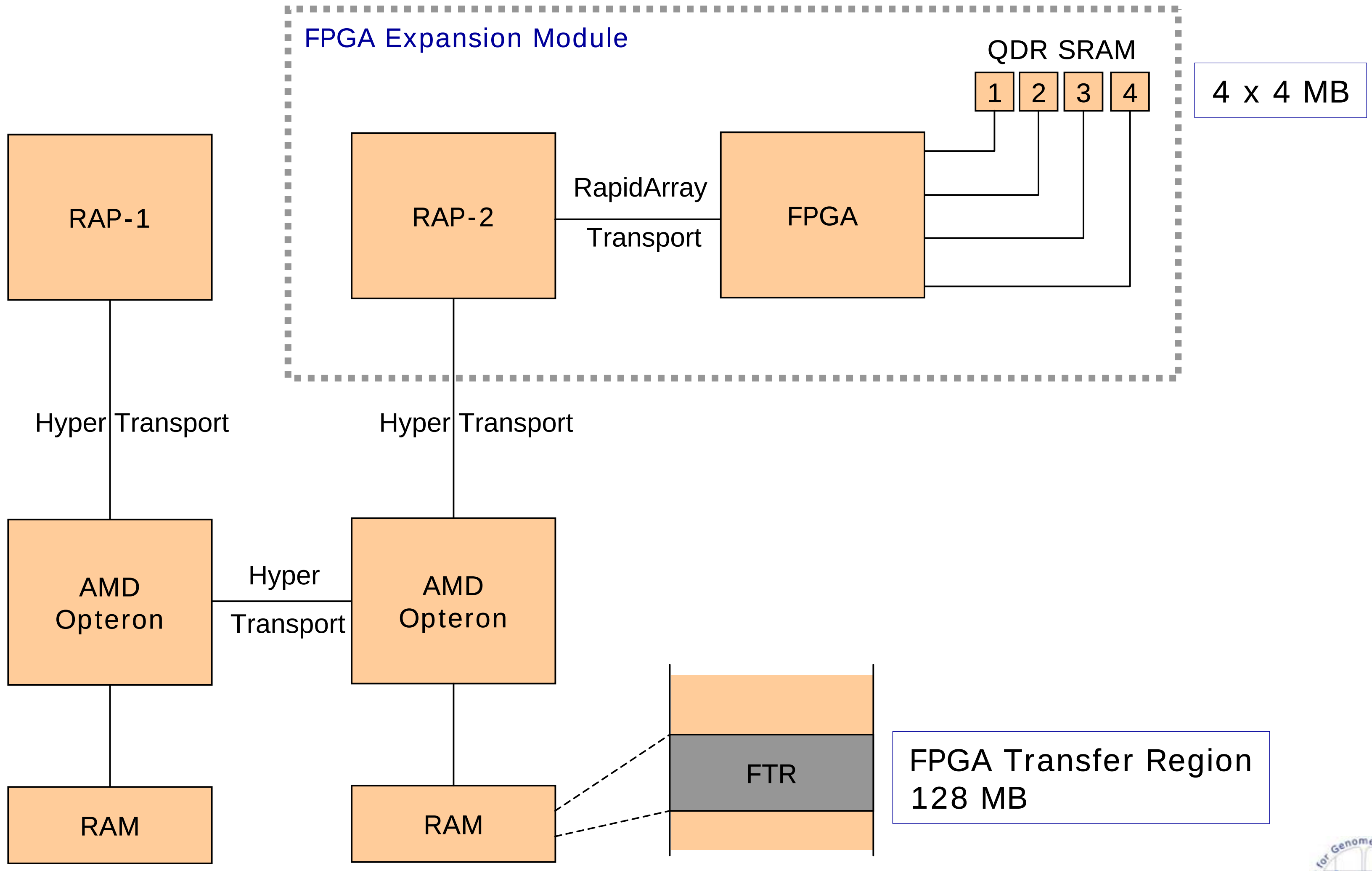
- portability & transparency
  - hardware abstraction layer
  - support for Fortran

- FPGA system integration
  - communication performance FPGA - host

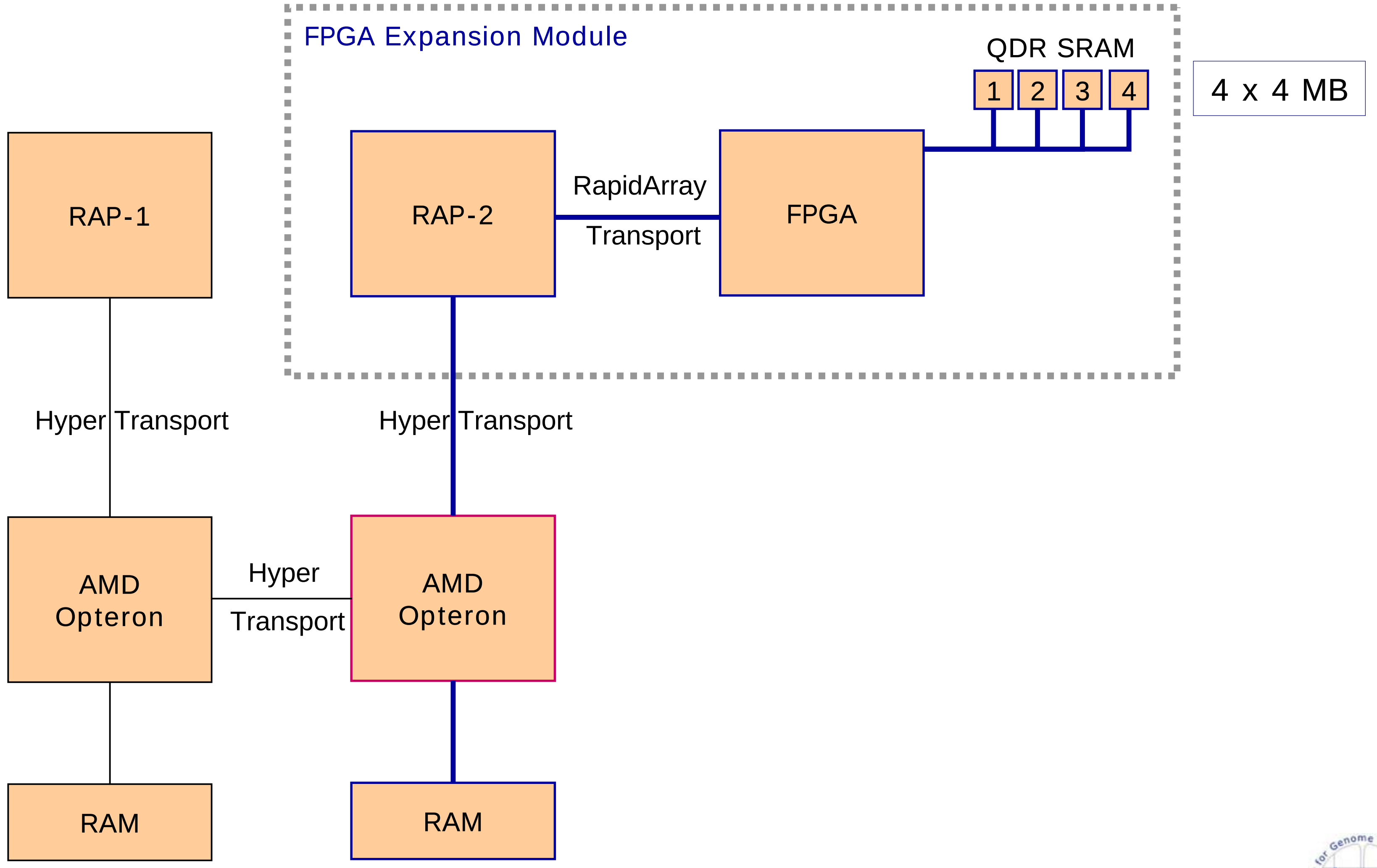
Outline

- Why we are interested in using FPGAs?
- FPGA programming with Mitrion-C
- Lessons learned and Outlook

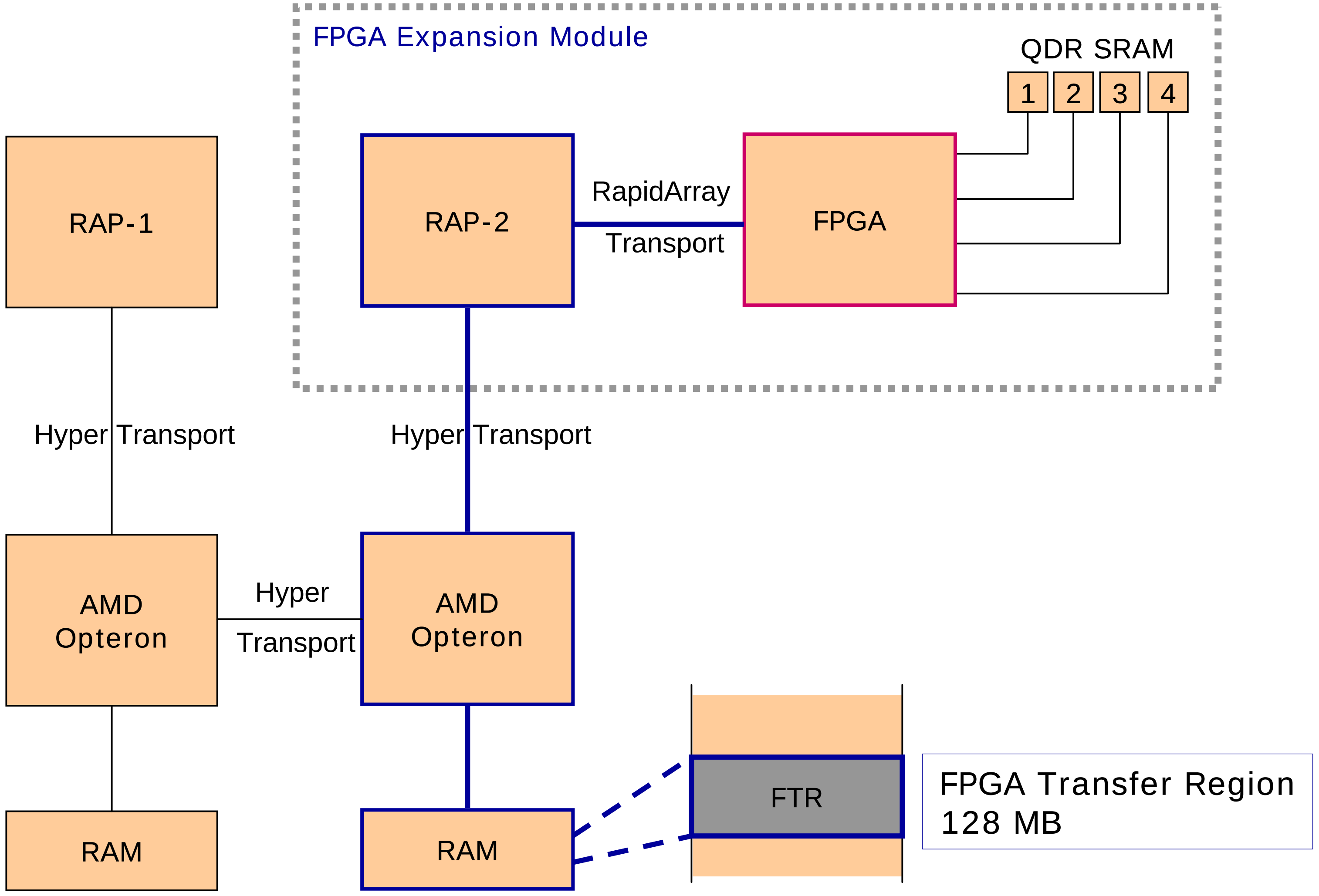
Host – FPGA Communication with Mitrion-C



Host – FPGA Communication: “stream”

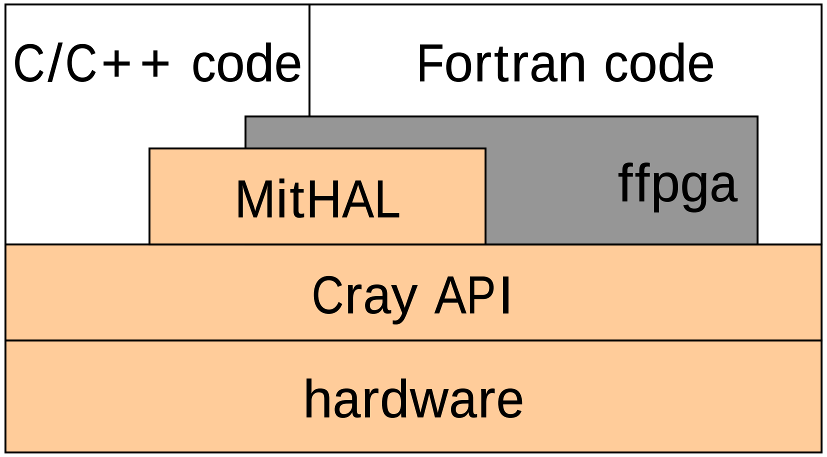


Host – FPGA Communication: “dram”



FPGA API: Fortran Layer **ffpga**

Cray API and Mitrion Hardware Abstraction Layer (Mithal)



```
ffpga_init  (bitfile, stat)
ffpga_reg_buffer(id, nbytes, mode, stat)
ffpga_start ( stat )
ffpga_wait  ( stat )
ffpga_close ( stat )
ffpga_wrreg (id, ptr, stat)
ffpga_wdta  (id, ptr, offset, nbytes, stat)
ffpga_rdta  (id, ptr, offset, nbytes, stat)
```

In/Out Bandwidth Measured with Mitrion-C

Type (Mithal version)	Data Transfer Chain	Bandwidth [Mbytes/s]
stream (1.1.1)	host memory → SRAM	764.9
	SRAM → host memory	4.9
	(total throughput)	(9.2)
dram (1.1.5)	FPGA → host DRAM	675.1

compiler: PGI 5.2
Mitrion-C 1.1.1/1.1.5
timing: PAPI3

Outline

- Why we are interested in using FPGAs?

- FPGA programming with Mitrion-C
  1. learning phase: bandwidth measurements
  2. Lennard-Jones (LJ) potential energy & forces

- Lessons learned and Outlook

Simulation of Large Molecular Systems

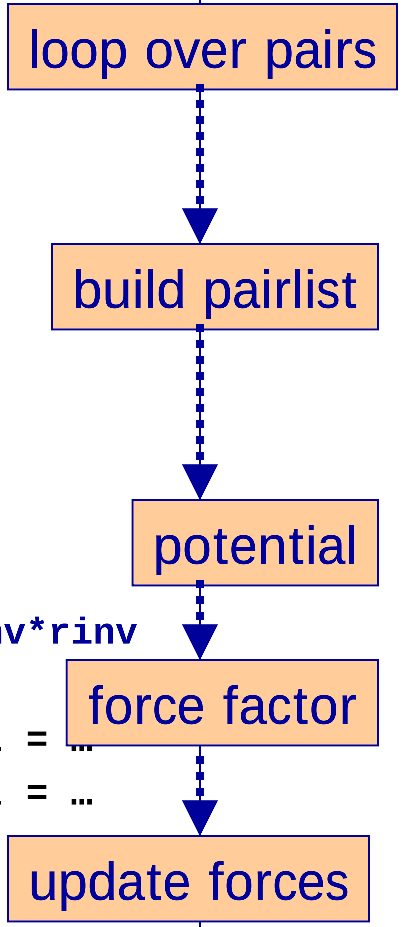
- life science, materials science, nano-tech...
- non-bonding interactions
- Lennard-Jones energy + forces

$$U(r) = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^6\left(\left(\frac{\sigma}{r}\right)^6 - 1\right)\right]$$

$$F(r) = \frac{24\varepsilon}{r^2}\left[\left(\frac{\sigma}{r}\right)^6\left(2\left(\frac{\sigma}{r}\right)^6 - 1\right)\right]$$

p1MD: nforces (ASIS)

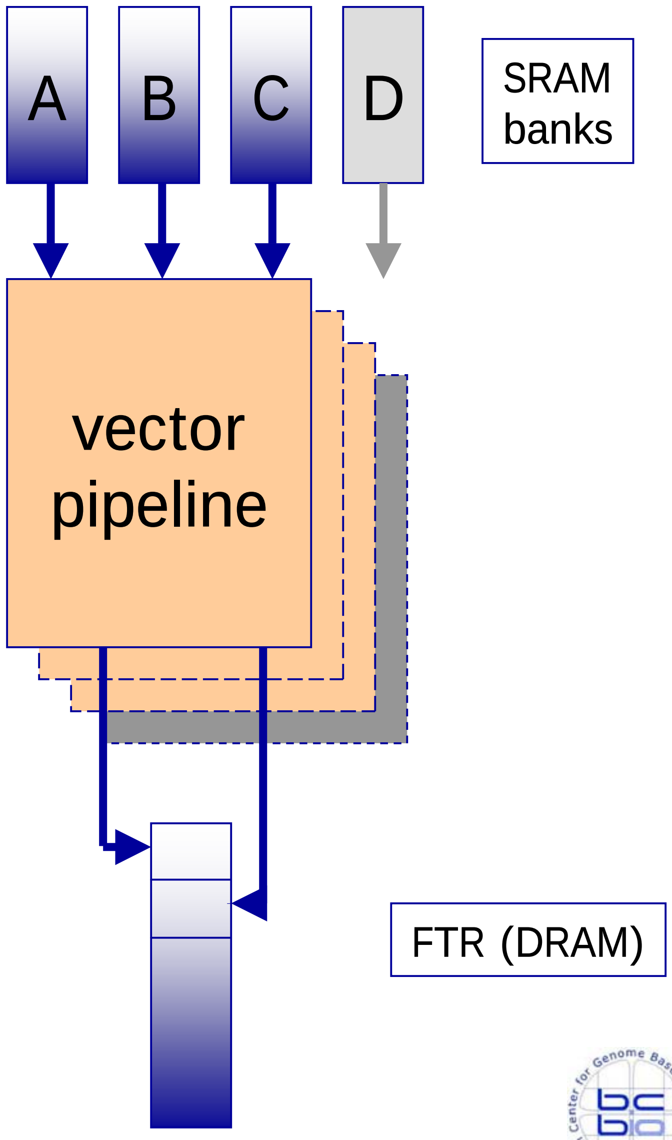
```
  subroutine nforce
    do imol=1, nmol-1
       …
       do jmol=imol+1, nmol
           …
           if ( r .lt. rtab ) then
               ntab = ntab+1
               …
               rinv = 1. / r
               r6   = (sigij(ki,kj)*rinv)**6
               ep   = epsij(ki,kj)*r6*(r6-1.0)
               epot = epot + 4.*ep
               A0   = epsij(ki,kj)*r6*(2.0*r6-1.0)*rinv*rinv
               A    = 24.*A0
               fx(imol) = fx(imol) + A*xx ; fy = …; fz = …
               fx(jmol) = fx(jmol) - A*xx ; fy = …; fz = …
           end if
       end do
    end do
end subroutine nforce
```



1st Approach(es): on FPGA

- streaming of pairlist data
  - 3(4) input streams: SRAM
    - $1/r$, ε, σ (, q)
    - 2x entities/cycle/SRAM
  - 1 output stream, interleaved: FTR (DRAM)
    - e_{pot}, $A0$ (, e_{Coul})

- naïve approach /w all 32bit IEEE won't work...
  - resource limitations on chip
  - decreased to 10 – 15 fp ops
  - deep pipeline(s)



p1MD: nforces (FPGA version)

```
subroutine fpga_nforce
   integer(kind=8)     :: ramA(1), …, ramH(1)
   pointer                (p_ramA, ramA)
   real                :: buffA(1), …, buffH(1)
   pointer                (p_buffA, buffA)
   if ( first ) then
      p_ramA = ffpga_reg_buffer('Am0', nbytes, WRITE_ONLY_DATA, istat)
      p_ramH = ffpga_reg_buffer('Hm0', nbytes, READ_ONLY_DATA, istat)
      p_buffA = p_ramA ; p_buffH = p_ramH
   end if
   do imol=1, nmol-1
      do jmol=imol+1, nmol
         …
         if ( r .lt. rtab ) then
            rinv = 1. / r
            buffA(ntab) = rinv
            buffB(ntab) = epsij(ki,kj)
            buffC(ntab) = sigij(ki,kj)
         end if
      end do
   end do …
```

init: allocate comm buffer

fill buffer for SRAM

p1MD: nforces (FPGA version)

```
    call ffpga_wrreg ('ftr_byteaddr', p_ramH, istat)
    call ffpga_start ( istat )
    …
    call ffpga_wait  ( istat )
    do i=1,ntab
       epot      = epot + 4.* buffH(k)
       A         = 24.   *     buffH(k+2)
       …
       fx(imol) = fx(imol) + A*xx(i)
       fx(jmol) = fx(jmol) - A*xx(i)
    end do
    return
end subroutine fpga_nforce
```

start FPGA calc

wait for termination

update data

LJ in Mitrion-C: main (1 pipe)

```
(ExtRAM, ExtRAM, ExtRAM, ExtRAM, ExtDRAM)
main (ExtRAM Am0, ExtRAM Bm0, ExtRAM Cm0, ExtRAM Dm0, ExtDRAM Hm0,
       RegSize ftr_byteaddr, RegSize start_offset)
{
  DType<STREAMLEN><2> buf_a; …
  DType<STREAMLEN2> av; …
  (buf_a, buf_b, buf_c, Am1,Bm1,Cm1) = read_data(Am0,Bm0,Cm0);
  av = reshape(buf_a, <STREAMLEN2>); …
  (wv,xv) = foreach(ri,eps,sig  in av,bv,cv)
  {
     (elj,fA) = efALJ_ij(ri,eps,sig);
  } (elj,fA);
  buf_x = reshape(xv, <STREAMLEN><2>); …
  Hm1   = write_data(Hm0,  buf_w,buf_x, offset);
  Hm2   = _wait(Hm1);
  AmFinished = Am1; …
} (AmFinished, BmFinished, CmFinished, DmFinished, HmFinished);
```

LJ in Mitrion-C: potential + force factor

```
efALJ_ij(rinv, epsilon, sigma)
{
   DType a  = rinv * sigma;
   DType b  = a*a*a*a*a*a;
   DType ce = b - 1;
   DType cf = 2 * b - 1;
   DType de = b * ce;
   DType df = b * cf;
   DType ep = epsilon * de;
   DType fA = epsilon * df * rinv * rinv;
} (ep,fA);
```

Performance of the Month: March 06 ...

version	code block	time [ms]	(for 28599 pairs)
ASIS	- loops	21.51	loop time in **`nforce`**
	call	21.61	function call time
1.1.1	- init	0.61	add. initialization overhead
03/06	- - wdta	0.61	data transfer to SRAM
	- - run	2.40	FPGA runtime
	- - rdta	43.50	data transfer from SRAM
	- - update	0.63	update force arrays
	- loops	69.45	loop time
	call	70.20	function call time

→ total speedup: 0.3

compiler:	PGI 5.2
	ASIS: –O3 –fastsse
	Mitrion 1.1.1
timing:	PAPI3

Performance of the Month: May 06 ...

LJ: 32bit IEEE floating point, 1 pipe:

version	code block	time [ms]
ASIS	- loops	21.51
	call	21.61
1.1.5	- init+fill buff	0.86
05/06	- - run +xfer	0.77
	- - update	0.73
	- loops	19.82
	call	20.81
total speedup: 1.1		

(for 28599 pairs)

loop time **`nforce`**

function call time

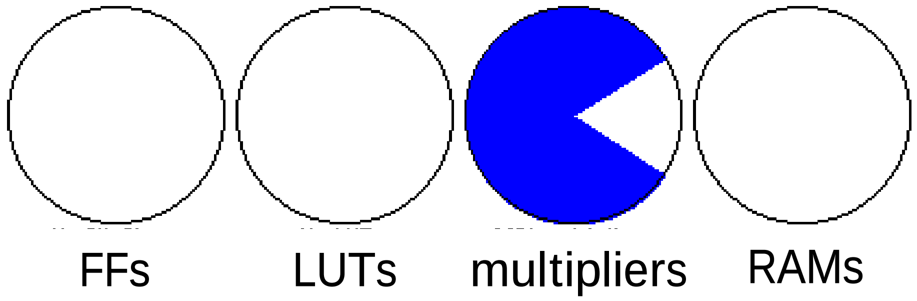
runtime includes data transfer

what happens here in between???

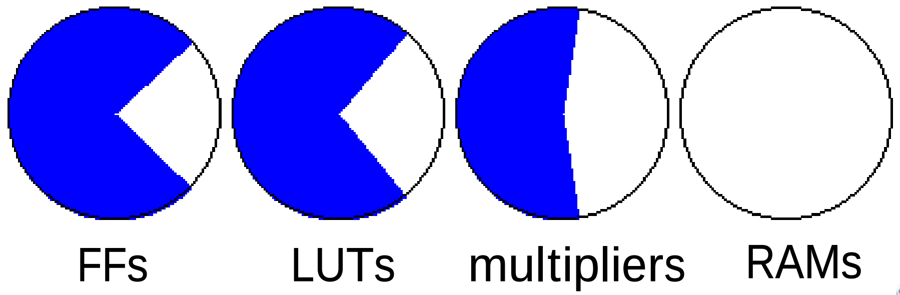
compiler:	PGI 5.2
	ASIS: –O3 –fastsse
	Mitrion 1.1.5
timing:	PAPI3

Multiple Pipes: Failed so far...

- **3 pipes**, 32bit IEEE fp, 1 particle position; stream interaction partners through...
  - Number of Slices: 159%
  - Number of Slice Flip Flops: 111%
  - Number of 4 input LUTs: 108%



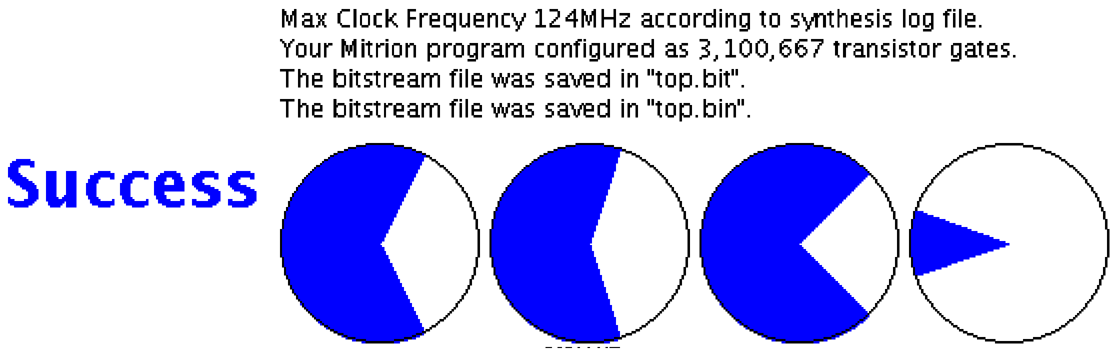
- **2 pipes**, 32bit IEEE fp, 1 particle position; stream interaction partners through...
  - Number of Slices: 104%



Work in Progress...

4 pipes, IEEE 32bit, force factor only

Number of Slice Flip Flops:	65%
Number of 4 input LUTs:	50%
Number of occupied Slices:	88%
Total Number 4 input LUTs:	60%
Number of Block RAMs:	11%
Number of MULT18X18s:	75%



```
Max Clock Frequency 124MHz according to synthesis log file.
Your Mitrion program configured as 3,100,667 transistor gates.
The bitstream file was saved in "top.bit".
The bitstream file was saved in "top.bin".
```

FFs LUTs multipliers RAMs

Lessons Learned

- strengths of Mitrion-C language approach
  - Application programmers can implement numerical algorithms on FPGA within a couple of hours.
  - The data dependency graph viewer is an effective tool to trace and analyse data streams.

- strength of the Mitrion simulator:
  - Substantially reduces the time used for the implementation and validation of algorithms.

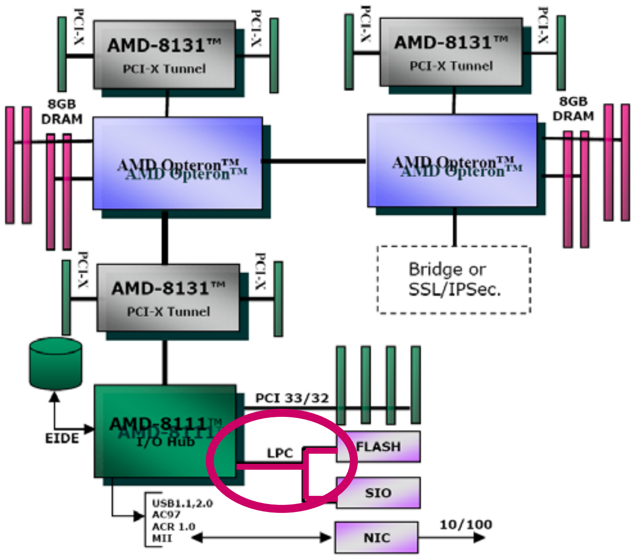
Lessons Learned

- host – FPGA interface
  - data transfer bandwidth is a critical issue and can limit the overall performance

- limited resources on the FPGA:
  - implementation of floating-point intensive computational kernels may fail due to limited functional components (BRAM, multiplier) on FPGA or constraints.
    - painful on the elder and smaller devices such as our XCVIIPro50.

Outlook

- fixed-point ops with different precision
  - precision simulation environment on Opteron
- RC at runtime → multi-step applications
  - load time 2.3 MB bitfile: 1.7 (disk) – 1.8 s (RAMdisk)
    - bottleneck: LPC bus performance



source: HPC group Karlsruhe

Scaling to New Heights
CUG 2006

Acknowledgments

Stefan Möhl
Pontus Bergendahl
Johan Lövdahl

Steve Margerm
Dave Strenski

Mitrionics A.B.

Cray Inc.

Open fpga
www.openfpga.org

Berlin Center for Genome Based Bioinformatics

funded by

AN INITIATIVE FROM
Federal Ministry of Education and Research