Programming for performance in FPGAs using multiple processors and accelerators with C/C++ programming

Kees Vissers kees.vissers@xilinx.com
Xilinx
MPSoC 2011

Contents

- Today's Semiconductor Landscape
- Processors and Pipelines
- C to RTL works
- Next step: ARM processors + FPGA
- Medical Application
- Conclusion

Key Industry Drivers for Change

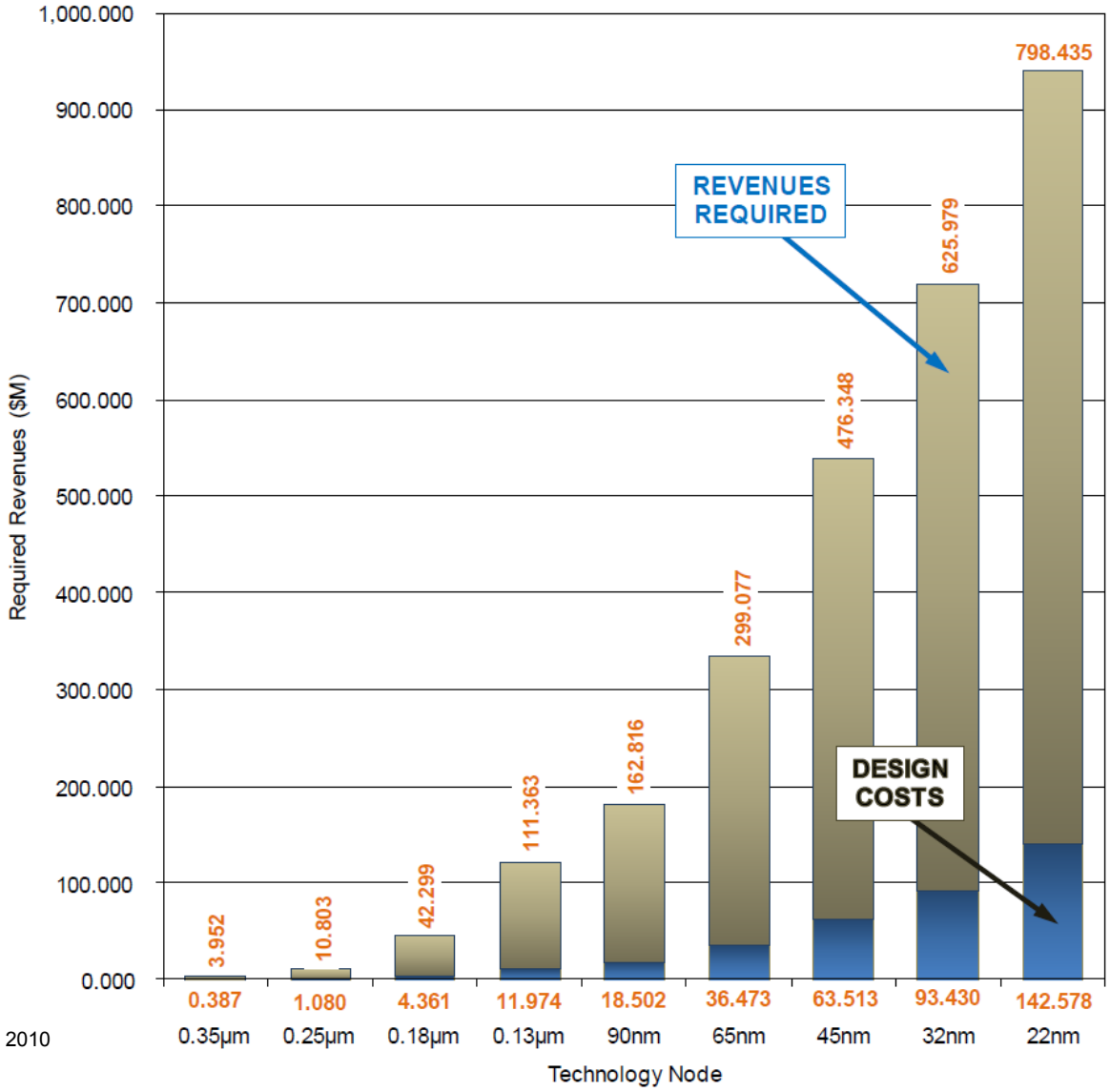
Insatiable Bandwidth

Ubiquitous Connected Computing

The Programmable Imperative

Broadening Markets

Fewer Companies Can Do it All



Investments from a few must be leveraged by many

Source: IBS March 2010

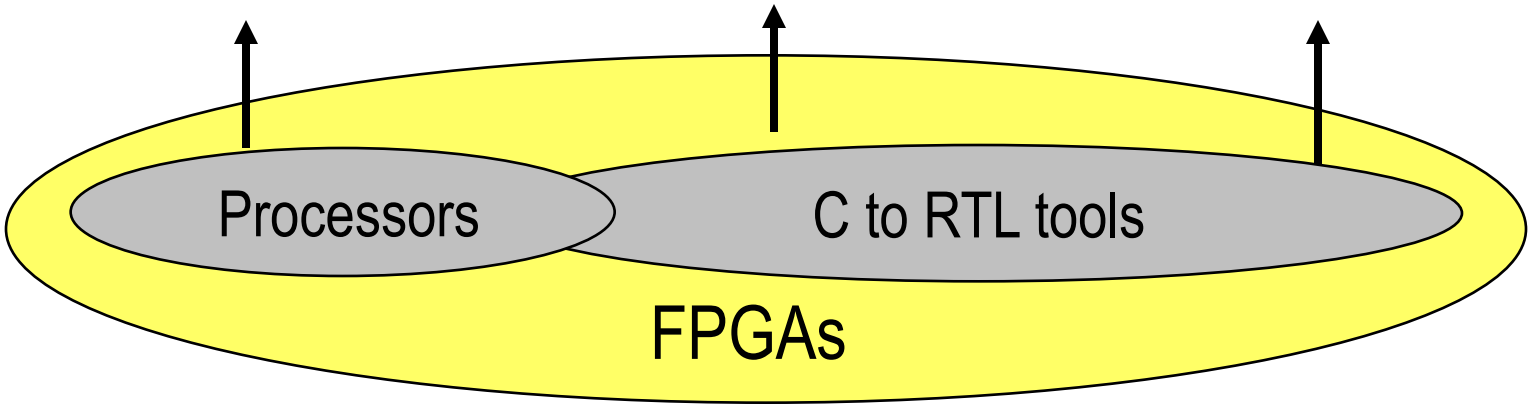
XILINX

The Era of ‘Crossovers’

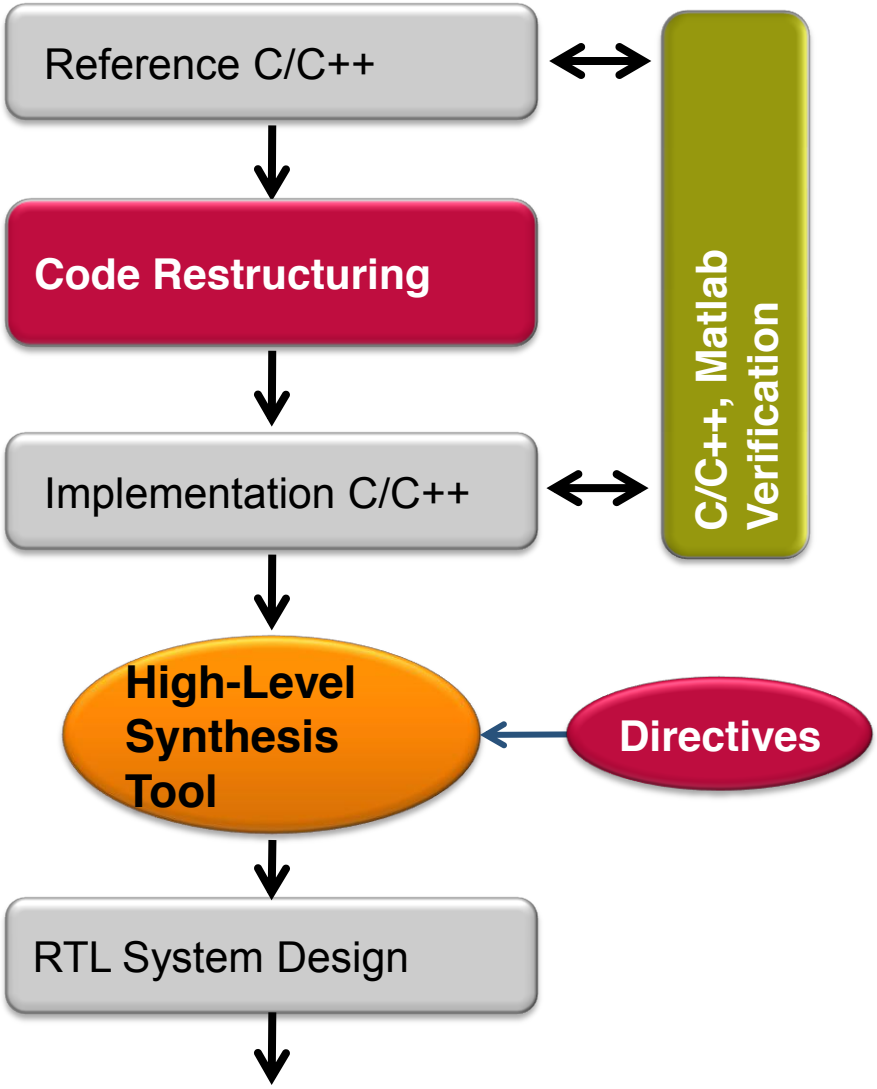
Processors and Pipelines

Design approach	RISC Proc.	Proc. w/ accels.	Folded datapath	Pipelined datapath	Replicated datapath
clock:sample	1000:1	100:1	10:1	1:1	1:10
Data Rate (200MHz clock)	200Ks/s	2Ms/s	20Ms/s	200Ms/s	2 Gs/s

Applications control → audio → mobile video → HDTV → comms → networking



High-Level Synthesis Tools for FPGA



- **Code Restructuring**
  - Macro-architecture description
  - Parameterization
  - FPGA Optimizations
- **Directives (*pragmas*)**
  - Specify performance
  - Mapping resources
- **C/C++ level verification**
  - Use traditional tools (C/C++ compiler, Matlab)
  - Re-use C/C++ testbench

C to FPGA tools summary

Programming with C to FPA tools for performance is comparable to programming for performance on a DSP

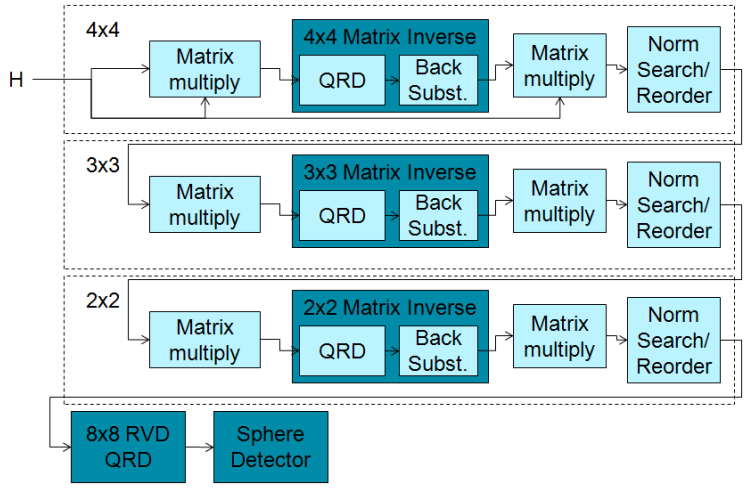
FPGAs deliver 30x the cost performance benefit compared to this DSP, programmed in C/C++

The results of good RTL design are comparable to what these C to FPGA tools can achieve

Xilinx acquired AutoESL in January 2011

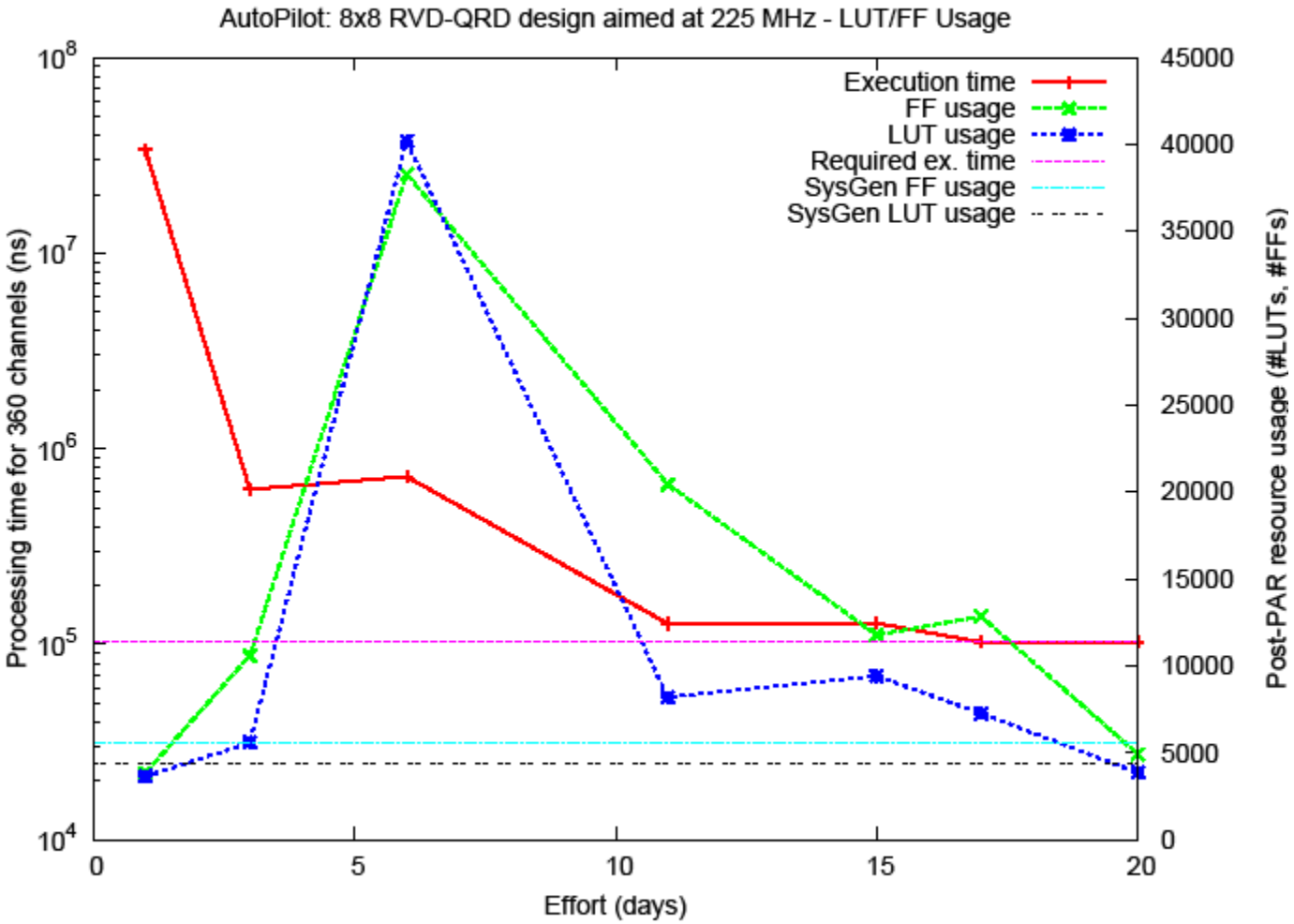
Mimo Sphere decoder application

- Results compared with good standard design: tool is mostly better
- Performance of this problem in the range of 150 Gops (mostly 16 bit)
- All C++ code with AutoESL directives
- Synthesized at 225MHz



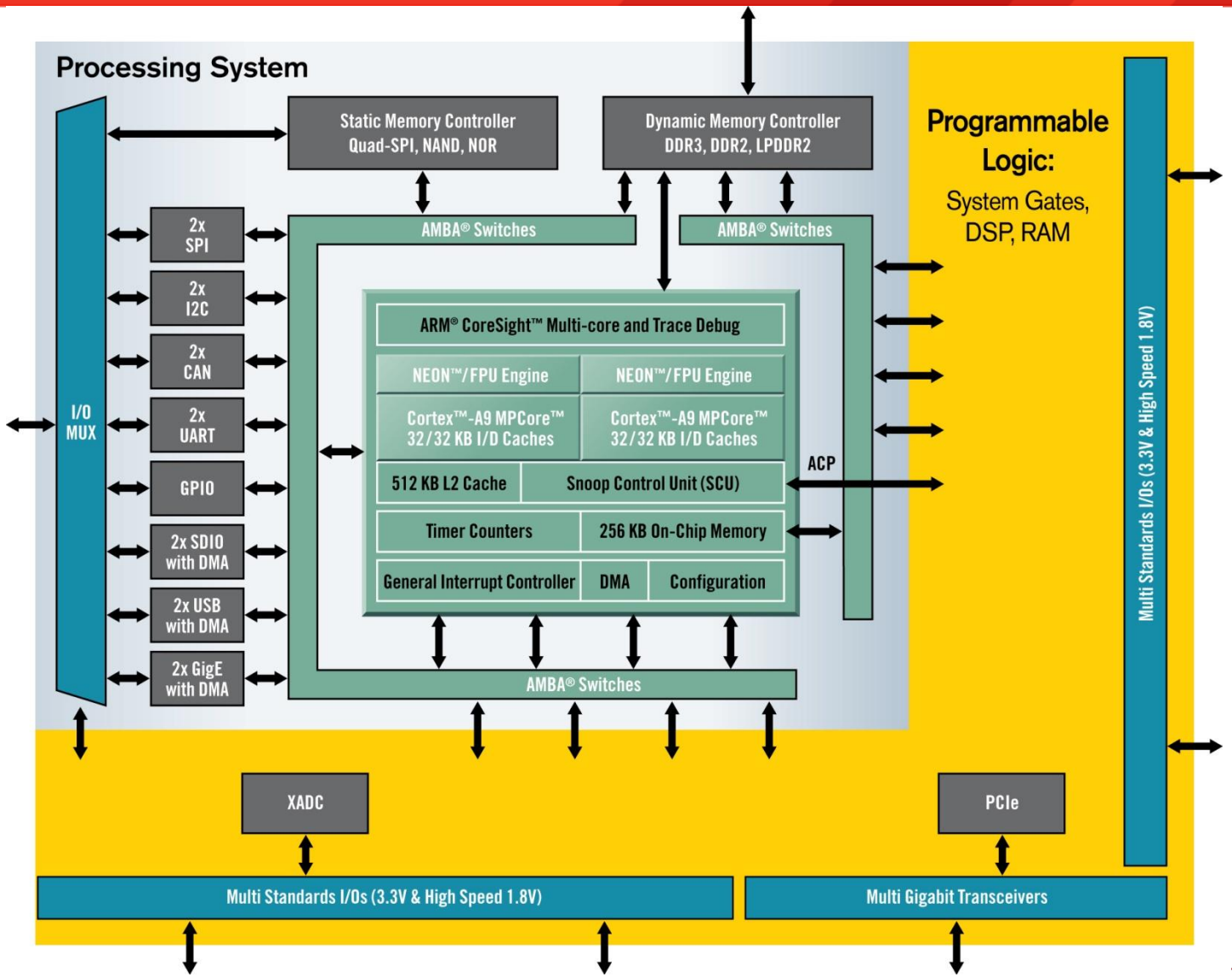
Metric	SysGen	AutoESL – Expert Result	% Diff
Development Time	16.5	5	-9%
LUTs	27,870	29,060	+4%
Registers	42,035	31, 000	-26%
DSP48 slices	237	201	-15%
18K Brams	138	99	-28%

Time to result



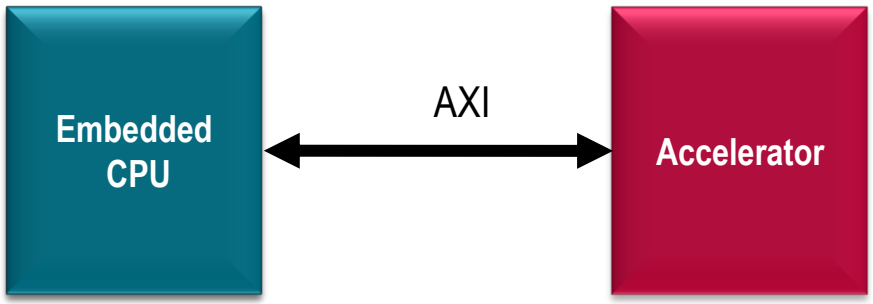
XILINX

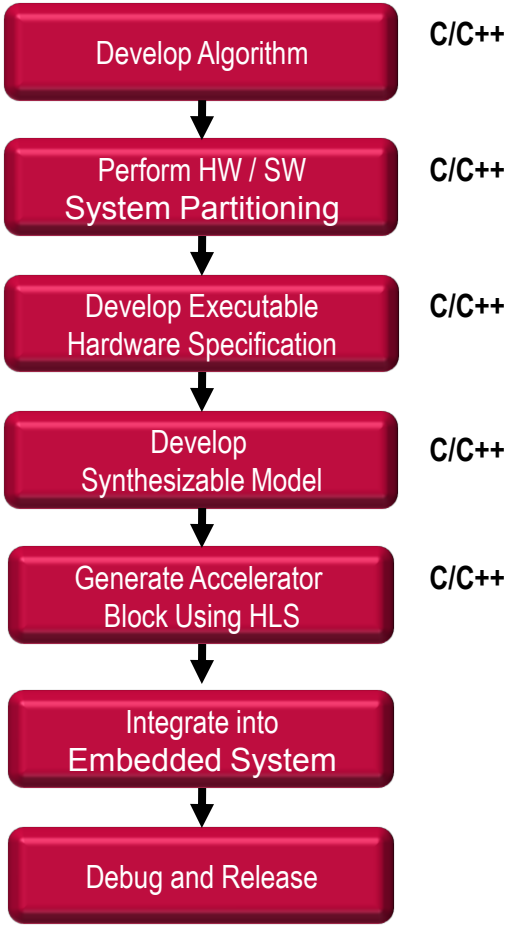
Zynq Architecture



The programmer's view: Partition and Integrate

```
Matrix Multiply Program;

void main() {


  for(i=0; i<DIM; i++) {
    for(j=0; j<DIM; j++) {
      a_re[i][j] = (i+1);
      b_re[i][j] = (j+1);
      out_re1[i][j] = 0.0f;
    }
  }

  matrix_multiply_accelerator(a_re, b_re, out_re);

}
```



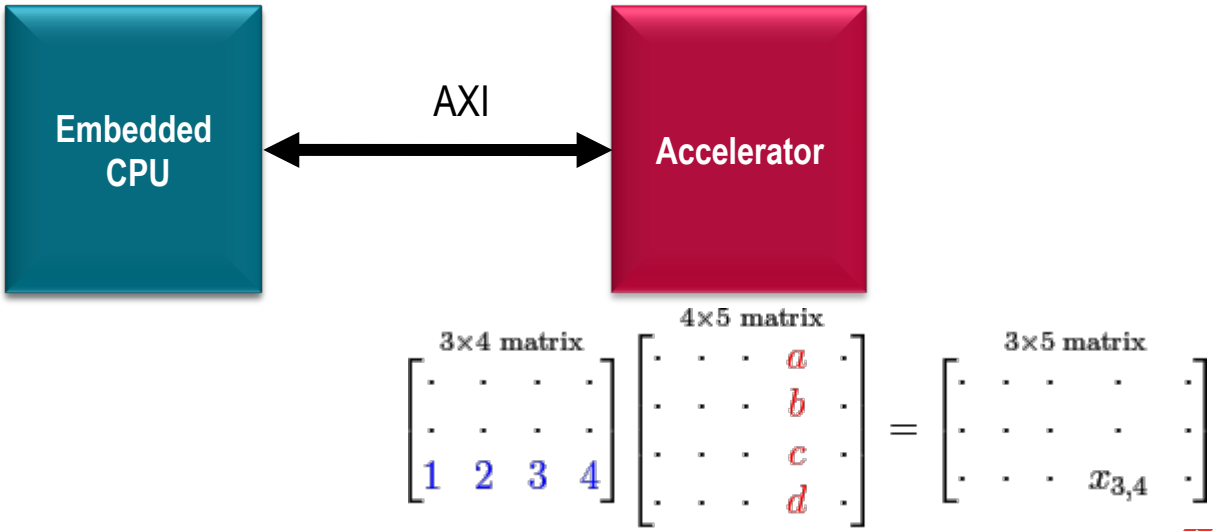


The programmers view: partition the code

```
Matrix Multiply Program;

void main() {



  for(i=0; i<DIM; i++) {
    for(j=0; j<DIM; j++) {
      a_re[i][j] = (i+1);
      b_re[i][j] = (j+1);
      out_re1[i][j] = 0.0f;
    }
  }
```

```
matrix_multiply_accelerator(a_re, b_re, out_re);
```

```
}
```



XILINX

Data movement matters

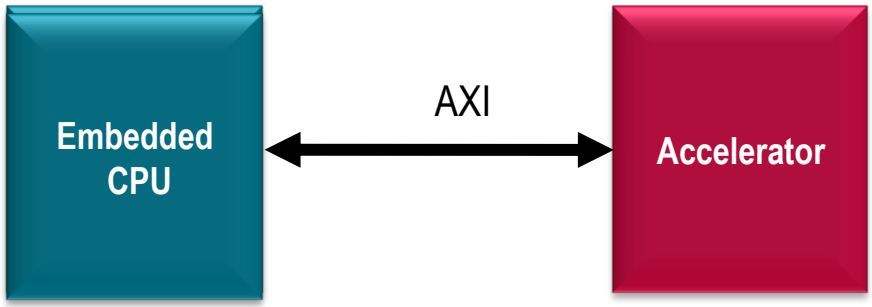
```
Matrix Multiply Program;

void main() {

  init_hw_accelerator();

  for(i=0; i<DIM; i++) {
    for(j=0; j<DIM; j++) {
      a_re[i][j] = (i+1);
      b_re[i][j] = (j+1);
      out_re1[i][j] = 0.0f;
    }
  }
```

Move 2 matrices

```
matrix_multiply_accelerator(a_re, b_re, out_re);
```

Move 1 matrix

```
}
```



Software programmer
Abstracts from data movement

Conceptually,
all variables are free
And can be used everywhere

Performance programmer
Needs Abstraction
from the details:

- Feedback on total data
- Impact on total time

Many ways to move data:
- memcopy
- DMA, central, distributed
- with ACP, with on chip memory
- flush cache and external mem.

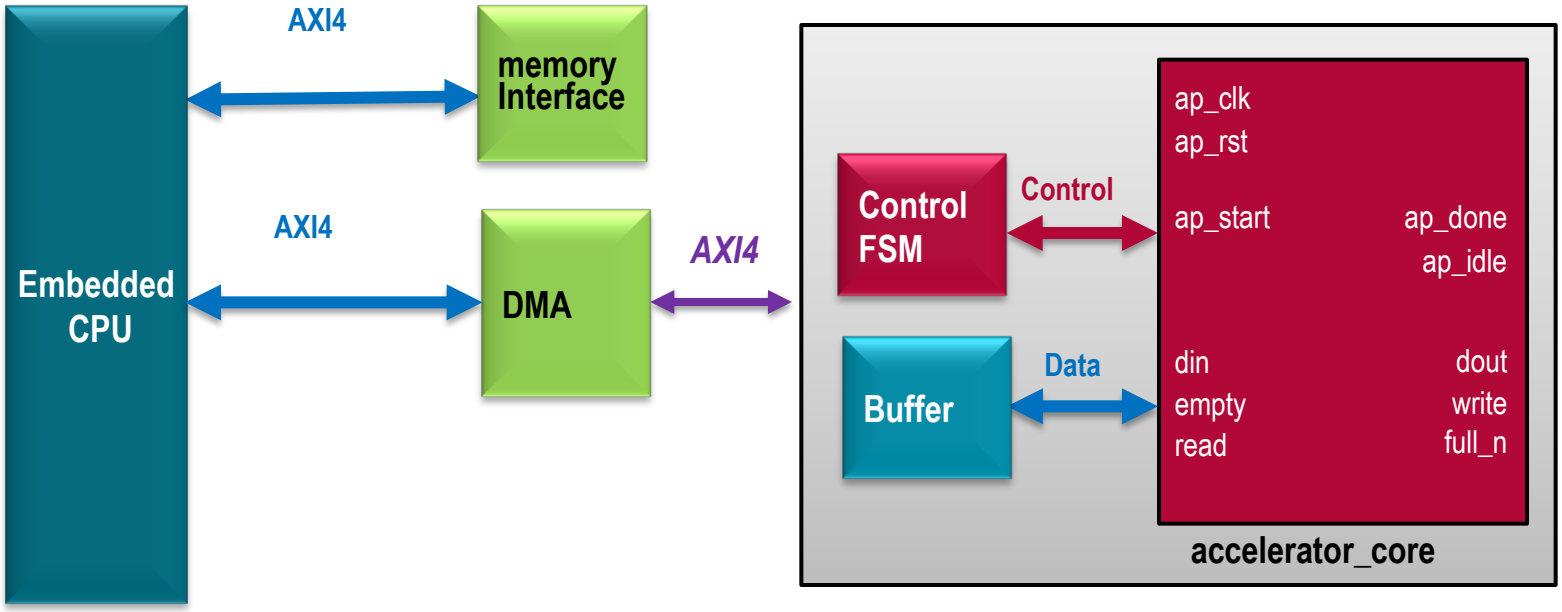
Some interesting challenges

- Simple example: 32 x 32 matrix multiply floating point calculation.
- A 32 x 32 x 4 byte matrix = 4Kbyte, is this page aligned?
- Do we run an OS on the processor(s)? Linux? SMP?
- Do we have the ACP port in cache coherency mode?
- Do we use DMA, what burst length? Buffer sizing?
- Is it faster using the On – chip memory?
- Are the accelerators using the same floating point math, same order of compute?
- Translation of User address space to physical address space?
- Can I stream multiple matrix multiplies back to back?

I'm not an SOC builder, I'm a software programmer: your platform should take care of this……

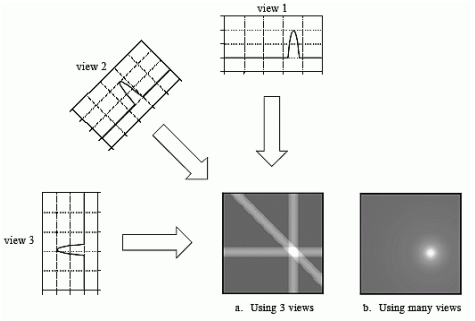
Abstractions for the software programmer

- Abstract the core generated by C to FPGA tools
- Provide main memory interface abstraction
- Provide DMA abstraction, buffer size abstraction, driver abstraction
- Software Programmers view for the total system

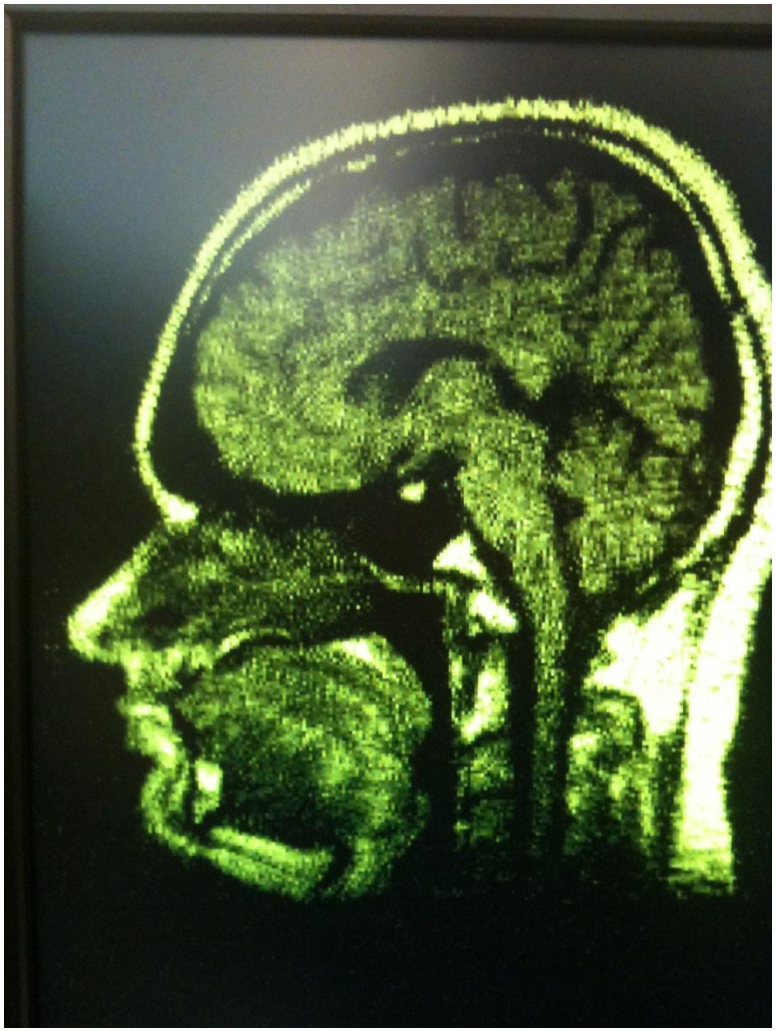


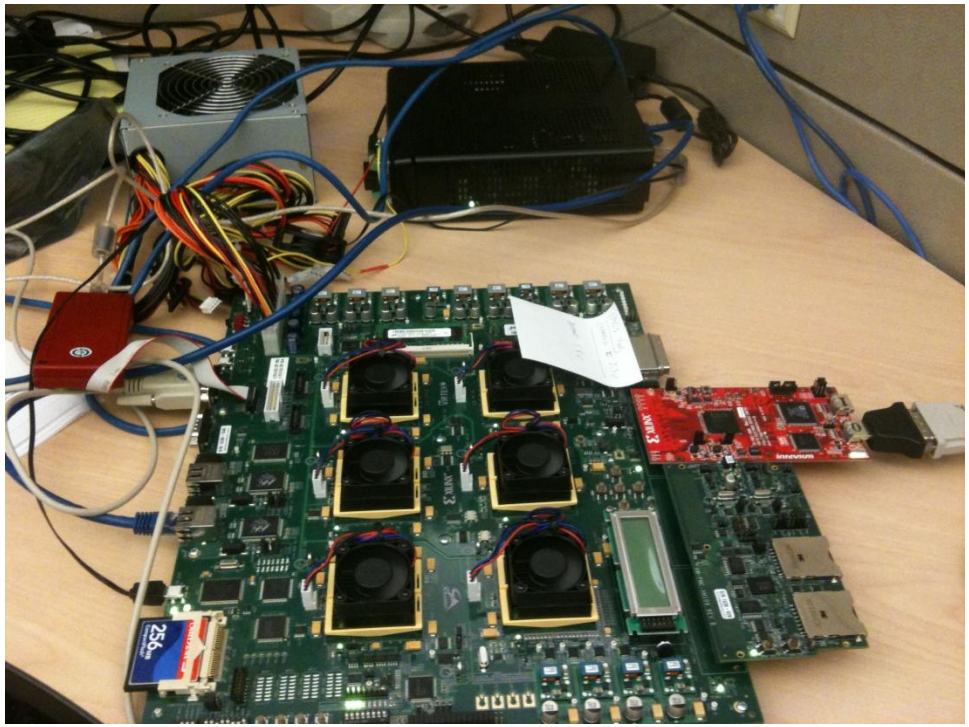
Back Projection problem

- The backprojection algorithm is used in a variety of tomography applications, including CAT scanners.
- Takes raw data from a scan at different angles and reconstructs an image based on that data.
- This design recreates a 256x256 pixel image from a 256x367 dataset (256 angles, each of which is 256*sqrt(2) = 367 elements)
- Floating point application.
- Runs on Arm processors with Neon
- Significant acceleration with 2 dedicated accelerators



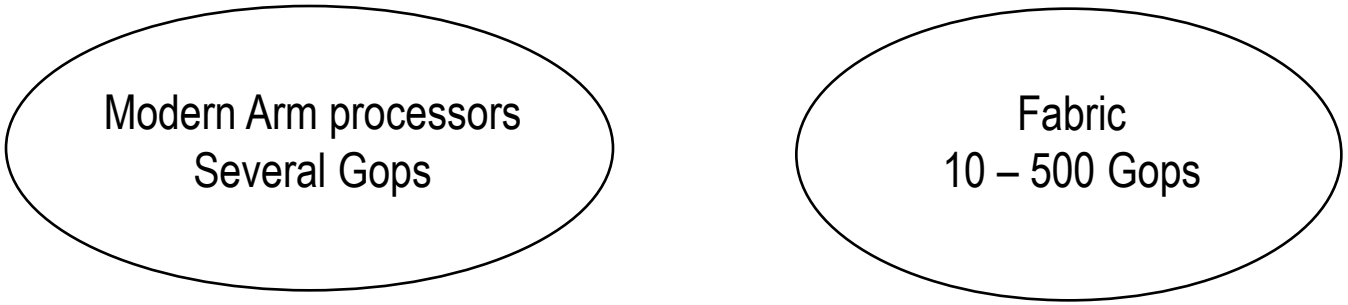
The output and our FPGA based prototype





Zynq Products in context

Design approach	RISC Proc.	Proc. w/ accels.	Folded datapath	Pipelined datapath	Replicated datapath
clock:sample	1000:1	100:1	10:1	1:1	1:10
Data Rate (200MHz clock)	200Ks/s	2Ms/s	20Ms/s	200Ms/s	2 Gs/s
Applications	control → audio → mobile video → HDTV → comms → networking				



Concluding Remarks

- C to FPGA tools work well in the signal processing domain and for floating point calculations.
- FPGA fabric can issue, the ‘equivalent’ of 100 -1000 risc operations every clock cycle.
- Communication latency and bandwidth between the processors and accelerators is important.
- Overlap transfer and compute: DMA, multi-thread.
- Next generation Processors + FPGA fabric become affordable.
- Rapid cost reduction: 28nm, 22nm and further.
- Pre- fabricated Heterogeneous programmable platforms offer power efficient flexibility.
- We do the hard platform construction, so that you can use it.

References

- Jason Cong, Bin Liu, Stephen Neuendorffer, Juanjo Noguera, Kees Vissers, Zhiru Zhang: **High-Level Synthesis for FPGAs: From Prototyping to Deployment**, IEEE Transactions On Computer-aided Design Of Integrated Circuits And Systems, Vol. 30, No. 4, April 2011, pages 473 – 491.
- Juanjo Noguera, Stephen Neuendorffer , Sven Van Haastregt, Jesus Barba, Kees Vissers, Chris Dick: **Sphere Detector For 802.16e Broadband Wireless Systems Implementation On FPGAs Using High-level Synthesis Tools**, Software Defined Radio Forum, December 2010
- Kees Vissers, Stephen Neuendorffer, and Juanjo Noguera: **Building real-time HDTV applications in FPGAs using processors, AXI interfaces and high level synthesis tools**. Design Automation and Test Europe Conference (DATE). 2011.
- www.xilinx.com/products/silicon-devices/epp/zynq
- www.bdti.com/resources/benchmarkresults/hlstcp/autopilot