Don't worry about the cores

April 2011
Overview

- Essentials
- Target Markets
- The picoArray concept
- Tool flow
- Hardening
- Conclusions
Essentials and Target Markets
Essentials for Multicore Adoption

• Reference designs - believable and realisable
• Scalable systems
• Short development time
• Mature Toolset
  – Single programming environment
  – Intuitive – not disruptive
  – Needs to do ‘what it says on the tin’
• System centric debugging
  – Not core outwards
  – Allow whole systems to be debugged and verified – in the field if necessary
### Market: Wireless Base Stations/Access Points

<table>
<thead>
<tr>
<th>“Large” – Macrocell</th>
<th>“Medium” - Picocell</th>
<th>“Small” - Femtocell</th>
</tr>
</thead>
<tbody>
<tr>
<td>400 users</td>
<td>40+ users</td>
<td>4/8/16/32 users</td>
</tr>
<tr>
<td>30+ Mbps total</td>
<td>14/21/42 Mbps total</td>
<td>14/21/42 Mbps total</td>
</tr>
<tr>
<td>$20,000 → $10,000</td>
<td>$2,000 → $1000</td>
<td>$200 → $100 → &lt; $50</td>
</tr>
</tbody>
</table>
# Reference Designs

<table>
<thead>
<tr>
<th></th>
<th>Multi-Core DSP</th>
<th>Multi-Core DSP + ARM</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image" /></td>
<td>PC102</td>
<td></td>
</tr>
<tr>
<td><img src="image2.png" alt="Image" /></td>
<td>PC203</td>
<td></td>
</tr>
<tr>
<td><img src="image3.png" alt="Image" /></td>
<td>PC205</td>
<td></td>
</tr>
<tr>
<td><img src="image4.png" alt="Image" /></td>
<td>PC202</td>
<td></td>
</tr>
</tbody>
</table>

**PC102**

**PC203**

**PC205**

**PC202**
Typical applications

• Aimed at communications systems e.g.
  – UMTS FDD/TD-SCDMA
  – IEEE 802.16
  – CDMA2000
  – UMTS-LTE
• Standards tend not to be 'fixed'
• Consist of:
  – Stream based DSP
  – Block based DSP
  – Control
• Natural parallelism from:
  – Channel per user
  – Multi antenna systems
Why have a family of devices?

- Programmability provides:
  - good “time to market”
  - adaptability when “standards” are changing
  - flexibility to allow a range of applications to be targeted
  - software updates installed “in the field”
  - however, higher cost than application specific design
- picoArray device family allow these attributes to traded off
- Customers initially require flexibility, then cost reductions
- Cost reductions come from “hardening” of software blocks into hardware blocks (AEs)
picoArray family of devices

- **PC101**
  - Purely programmable AEs with application specific instructions
- **PC102**
  - Hardened correlation and Viterbi operations into an AE type
- **PC20x (3 device types)**
  - ARM 9 processor added + Hardened FFT, Cryptographic operations, Viterbi, Turbo decoder, Reed-Solomon in AE types
- **PC3x2/PC3x3**
  - ARM 1176 processor
  - Uplink/downlink processing as AE types
Mobile networks are changing...

Voice network

Data network
Device Instantiation
picoArray PC102

- One instantiation of picoArray
- Peak performance
  - 197 GIPS
  - 38.6 GMACs
  - 3.3Tbps communications bandwidth
- 308 programmable processors
- 14 co-processor accelerators for FEC, correlators
- Support for device fabrication errors increases yield
WCDMA Femtocell
WCDMA Femtocell
Femtostick
The picoArray Concept
The picoArray concept

• Targeted at wireless communications applications
• Why highly parallel hardware?
  – wireless systems have great deal of parallelism
  – software design simpler than hardware
  – software defined gives flexibility
  – scalable solution
• Replacement for DSP, ASIC, FPGA combinations
  – single architecture
  – single development environment
• Inter-process communications is fixed at compile time
  – no run-time arbitration
  – deterministic
The picoArray concept: Array Elements (AE)

- 16-bit processor
- 64-bit LIW targeting 3 execution units
- 160MHz clock
- Harvard architecture
- Processor and ports work independently
Array Element execution units

- ALU.0
- Comms Unit
- Memory Access Unit/ALU.1
- Branch Unit
- Spread/Despread Unit
- MAC Unit
- STAN

- ALU.0
- Comms Unit
- Memory Access Unit/ALU.1
- Branch Unit
- Multiply Unit
- MEM CTRL

- LIW.0
- LIW.1
- LIW.2
The picoArray concept: Architecture overview

- Processor
- Switch Matrix
- Inter-picoArray Interface or Parallel Asynchronous Interface
- Example signal flows
The picoArray concept: Interconnected picoArrays

• Several tens of picoArrays may be connected together and programmed as one group.
The picoArray concept: picoBus
picoBus: cont.
picoBus: cont.
picoBus: cont.
picoBus: cont.
picoBus: cont.
picoBus: cont.
picoBus: cont.
picoBus: cont.
The toolchain
Tool chain

- VHDL parser
- ANSI C Compiler (gcc)
- Cycle accurate simulator
- Design partitioning
- Place and Switch (Plastic)
- Network checking
- Debugging
  - Simulation
  - Hardware acceleration
Tool chain cont.

```
vhdl File
  \|-- picoAnalyse
     \|-- picoElaborate
        \|-- picoGcc
        \|-- Assembler
            \|-- picoPartition
                \|-- .pdes File
                    \|-- .tdl File
                    \|-- .seg File
                        \|-- picoDebugger
                            \|-- Simulator Executable
                                \|-- picoDebugger
                                    \|-- .pa File
                                        \|-- picoPlastic
                                            \|-- .pa File
                                                \|-- Library generation
                                                    \|-- .c File
                                                        \|-- .o File
                                                            \|-- .s File
                                                                \|-- picoGcc
                                                                    \|-- .a File
                                                                        \|-- ar
" lors
Input language is a mixture of:

- C/ASM – Used to program individual processes
- VHDL – Used to connect processes together using signals

All processes and signals are allocated at compile time.

Processes communicate over signals using PUT/GET in ASM, and builtin (intrinsic) functions in C.
entity Producer is
  port (decodedData:out integer32 @8);
end entity Producer;

architecture ASM of Producer is
begin
  MEM
    CODE
      COPY.0 0,R0 \ COPY.1 1,R1
    loopStart:
      PUT R[0,1],decodedData \ ADD.0 R0,1,R0
      BRA loopStart
    ENDCODE;
end;

Entity Consumer is
  port (decodedData:in integer32 @8);
end;

architecture C of Consumer is
begin
  STAN
    CODE
      long array[10];
      int main() {
        int i = 0;
        while (1) {
          array[i] = getdecodedData();
          i = (i + 1) % 10;
        }
        return 0;
      }
    ENDCODE;
end Consumer;
use work.all;

entity Example is
end;

architecture STRUCTURAL of Example is
  signal dataChannel: integer32 @0;
begin
  producerObject: entity Producer
    port map
    (decodedData=>dataChannel);
  consumerObject: entity Consumer
    port map
    (decodedData=>dataChannel);
end;

- Work.all is rest of current file
- STRUCTURAL defines connectivity
- Port map actually assigns signals to ports
C Compiler

- Based on GNU Compiler Collection (GCC)
- ANSI/ISO standard C
- Scheduling of LIW uses DFA scheduling algorithm
- Can be used to generate stand-alone libraries
- Communications supported using special functions
- Supports intrinsics e.g. BREV
Simulation

- Models entire systems including peripherals
- Cycle accuracy possible with back annotation from Plastic
  - Signal timings
  - Inter picoArray timings
- Used to verify rtl design to ensure working silicon
Simulation: cont.

```c
26 {
27     int i = 0;
28     integer16pair value;
29     while (1)
30     { 
31         for (i=0; i<10; ++i)
32         {
33             value.el1 = i;
34             value.el2 = i + 1;
35             putoutPort(value);
36         }
37         value.el1 = 0;
38         value.el2 = 0;
39     }
40 }
```

```
#0 0x0032 in main () at cchain.vhd:92
```

```
{ int i = 0;
    integer16pair value;
    while (1)
    {
        value = getinPort();
        value.el1 = (unsigned)value.el1 / (unsigned)putoutPort(value);
        i = (i + 1) % 1024;
    }
    for (;;);
}
```

Design Partitioning

- Partitions design between multiple chips (manual)
- Automatically splits signals which cross chip boundaries
- Peripherals must be placed on specific processors
- Output provided for Plastic in the form of
  - Tcl command file (one per chip)
  - Segmented design file (one per chip)
Plastic (Place and Switch to IC)

- Works on a single chip at a time
- Automatically places processes on processors
- Automatically switches (routes) signals between processors
- Attempts to minimise overall bus usage
- Manual operation is possible but difficult
- Output is a load file
Debugging and verifying picoArray systems

Differs to debugging and verifying sequential, or small scale parallel systems in the following ways.

• Scale
  - Thousands of processes and signals
  - System-wide debug and verification rather than process-centric

• H/W support
  - Silicon area best used for computation. Keep support to a minimum to allow more processors to be fitted onto a device.
  - System-wide debug and verification, rather than processor-centric

• Embedded environment

• Communications and synchronisation
  - Deterministic interconnect fabric – the picoBus
  - No runtime arbitration – removes source of possible bugs
Probes

- AEs are used to “spy” on communications in order to gather useful data
- This approach is non-intrusive and has no impact on the performance of signal processing blocks
- Relies on signals ability to be one-to-many
- The term “Probe AE” is used to describe an AE whose sole function is to gather data
- These are not special purpose pieces of hardware
Probes: cont.
Hardening Approach
Behavioural Simulation Instance

- Allows arbitrary C++ model connected to the picoBus in simulation
- Can be used for
  - design decomposition
  - basis for “hardening flow”
Hardening flow

- Wireless applications produced using Software Reference Designs (SRD) – typically 400-500 AEs
- One or more SRDs taken as basis
  - Partitioned into a number of blocks (minimizing the picoBus communications between blocks)
  - Blocks smaller than minimum size are combined
  - Partitioning revised depending on reuse criteria
- Blocks are modelled as BSIs
- Blocks then coded in RTL using BSI as “golden reference”
- Test benches from SRDs can be reused in verification
Hardening Flow
Dual FFT software block diagram (PC102)
Dual FFT AE block diagram (PC20x)
• The PC102 is based upon the picoArray™ architecture.

• The picoArray™ core contains the array of processing elements.
picoXcell: femtocell complete Solution

- PC2257 OAM TR069/196
- PC2258 SON
- PC2259 Radio Resource Management
- PC2252 Synch
- Radio
- Network Sniffer PC8210/11
- PHY/Node B PC302 / PC312 / PC3xx

Carrier Customization

- Not included from picoChip: part of OEM customer support for carrier

- PC2200 FAP SW
  - Object code from picoChip or PC2209 source from CCPU with options for customization Development

- picoChip standard product:
  - Radio devices from partners
  - RadioAPI, drivers, schematics design from picoChip
PC202 / PC205 Product Block Diagram

- UART 1
- UART 2
- JTAG
- SRAM
- Memory Interface
  - ARM926EJ
- DMA Controller 1
- DMA Controller 2
- Processor Interface & External Bus Interface (Master mode)
- Ethernet MAC
- Vectored Interrupt Controller
- GPIO
- Timers
- APB Bridge
- Watch-Dog Timer
- Remap/Reset Register
- Real-Time Clock
- Security
- Turbo
- Viterbi
- FFT
- Reed-Solomon
- SD GPIO
- IPI / ADI
- JTAG
Typical CPE Using PC302
Conclusions

- picoArray concept gives scalable, software defined systems
- Rapid development due to
  - deterministic communications
  - single programming environment
- Integrated tool set
- Probes provide non-invasive debugging and monitoring