More than just chips

September 2011
Overview

- The picoArray concept
- Tool flow
- Hardening
- Conclusions
The picoArray Concept
The picoArray concept: Architecture overview
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
picoBus: cont.
Several tens of picoArrays may be connected together and programmed as one group.
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.
picoDeveloper

![picoDeveloper Interface](image)

### Library
- `/home/chant5/Tools/Verification/flow/default.lib`
  - **Edit**
  - **New**
  - **Browse...**

- **Use last entity:**
- **Entity Name:**
- **Architecture Name:**

### Control
- **Execution mode:**
  - **Sw**
  - **Hw**

### Status
- **picoAnalyze**
- **picoElaborate**
- **picoPartition**
- **picoPartition Dev0**
- **Simulator**

- **Interactive**
  - **Options**

- **Clean**

### Commands
- **Source File...**
- **Save History...**
- **Exit Child**

### Logging
- **Command:**

- **Hide Log**
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
  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
  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 @8;
begnin
  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
int i = 0;
integer16pair value;
while (1)
{
    value = getinPort();
    value.el1 += (unsigned)value.el1 / (unsigned)value.el2;
    putoutPort(value);
    return 0;
}
```
Simulation: cont. - Design browser 1
Simulation: Cont. - Design browser 2
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
Plastic: cont.
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
Behvioural 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
Dual FFT software block diagram (PC102)
Dual FFT AE block diagram (PC20x)
Conclusions

- picoArray concept gives scalable, software defined systems
- However need more than just chips.
  - Deterministic communications
  - Single programming environment
  - Integrated tool set
  - Non-invasive debugging and monitoring using probes