MAMBO: A Simple Soft-core Processor used in the COPPER Mission

Steve Massey
Electrical Engineering 2013
Saint Louis University
Saint Louis University
Space Systems Research Lab

Parks College of Engineering, Aviation and Technology
36 full-time faculty, 600 students
AE, ME, EE, BME, Civil, Aviation, Physics
SSRL organized in 2009

Joined AFRL’s University Nanosatellite in 2009
COPPER, Nanosat-6, 2009-2010
Argus-High, Nanosat-7, 2011-2012

COPPER and Argus manifested through NASA CubeSat Launch Initiative
The COPPER Mission

Imaging Mission:
Utilize a commercially available compact uncooled microbolometer array to:
1. Capture infrared video of co-manifested satellites during separation phase
2. Capture infrared images of Earth’s oceans and atmosphere

Radiation Mission:
Improve the predictive performance modeling of radiation effects on small, modern space electronics devices by collecting radiation particle collision data from electronic monitoring experiments and relaying the data to the ground

Project Duration: 2009-2013
Initial concept: 2009-2010 Nanosat competition
Mission Modified to Fit the CubeSat Launch Initiative
Manifested for Launch: August 2013
Solution Requirements

- Video Payload must integrate into the COPPER spacecraft
  - Communicate with flight computer; 9600 baud downlink
  - Limited power budget (generation of 2W-hr/orbit)
  - Survive Low Earth Orbit environment (~500km altitude)

- Video Payload must store required information
  - 3-5 minutes of 320x240x14 video (10FPS min, 30FPS ideal)
  - Individual stills when requested
Solutions

• Looked closely at MicroBlaze and PicoBlaze offerings from Xilinx

  • Microblaze
    – Huge logic/BRAM usage requirements.
    – Offered libraries need off-chip storage (data and program RAM) for all but simplest programs
    – Bus transfers super slow, difficulty getting simulation license

  • Picoblaze
    – Limited codespace, address space

• Roll-your-own state machine
  – Quickly became too-complex, difficult to manage
Core Requirements

• Store data at sustained 25MB/s
  – FLIR’s CMOS bus is 75% downtime on a 12MHz clock
• Fit in a mid-grade Spartan3E-class part
• Easily extensible using a simple bus
Solution: MAMBO

• MAMBO: Moving And Manipulating Bytes
  – Midpoint between Microblaze and Picoblaze

• Programmable bus arbiter
  – 32-bit instructions focused on moving data across the bus, simple bitwise operations
  – 16-bit bus with 8-bit addressing (up to 32 with 24-bit extension)

• Harvard Architecture
Dataflow
Instruction Overview

• 32-bit instruction
  – 8-bit instruction identifier
  – 24-bit instruction data

• Typical Data Usage:
  – 8-bit standard address, 16-bit immediate data
  – 24-bit extended address
Instruction Overview

- Immediate data write

<table>
<thead>
<tr>
<th>xA0</th>
<th>xC5</th>
<th>x003E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier</td>
<td>Standard Addr</td>
<td>Immediate Data</td>
</tr>
<tr>
<td>31</td>
<td>24</td>
<td>23</td>
</tr>
</tbody>
</table>

- Relative branch if RA is less than immediate value

<table>
<thead>
<tr>
<th>xE3</th>
<th>xFE</th>
<th>xCD14</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier</td>
<td>Relative PC</td>
<td>Immediate Data</td>
</tr>
<tr>
<td>31</td>
<td>24</td>
<td>23</td>
</tr>
</tbody>
</table>
Bus Overview

• 8-bit address line
• 32-bit optional extended address line
• Two control signals:
  – Chip Enable
  – Write Enable
• 16-bit data “to-CPU” line
• 16-bit data “from-CPU” line
Current Peripherals

• SD Card interface
  – Standard capacity 4-bit 50MHz operation, 25MB/s
  – Single Read, Single Write, Burst Write
Current Peripherals

• UART
  – 9600bps, N81

• General Purpose I/O
  – 16-bit width, assignable
  – Separate Input and Output registers

• Cypress SRAM Controller
  – 4MB off-chip RAM
Current Peripherals

• Block RAM
  – 256 16-bit words available when using extended addressing

• Additional Registers
  – Supplement processor registers with 16x 16-bit words addressable

• FLIR Tau 320
  – 320x240x14 single-plane CMOS imager
  – Long-Wave Infrared
Example Code

```
-- program memory banks

type prog_mem is array ( 0 to num_instr ) of std_logic_vector(31 downto 0);

constant rom_instr : prog_mem := (  
  0 => x"FF000000", -- reset
  1 => x"FF000000", -- reset
  5 => x"1000000", -- rA = 0
  6 => x"1400000", -- rC = 0
  7 => x"1100000", -- rB = 0
  9 => x"A8d0000", -- write rB to (xE2, rC)
 10 => x"40000001", -- rC = rC + 1
 11 => x"26000001", -- rA = rA + 1
 12 => x"28000001", -- rB = rB + 1
 13 => x"e3fb0fff", -- rA brlt xFF
 17 => x"b0f40000", -- rA <= cache5
 19 => x"b0f50000", -- rA <= cache6
 21 => x"b0f60000", -- rA <= cache7
 23 => x"b0f70000", -- rA <= cache8
 25 => x"10000000", -- rA = 0
 26 => x"14000000", -- rC = 0
 27 => x"11000000", -- rB = 0
 29 => x"b7d00000", -- read (xE2, rC) to rB
 30 => x"40000001", -- rC = rC + 1
 31 => x"26000001", -- rA = rA + 1
 33 => x"e3fb01ff", -- rA brlt xFF
 50 => x"e0000002", -- restart system
 others => x"00000000" );
```
Example Code
Debugging

- Simulation: ModelSim SE
- On-Chip: ChipScope Pro
Future Roadmap

- FORTH Interpreter
- Interrupts
- Testing, testing, testing!
  - Wrap up for COPPER
  - Explore additional uses
MAMBO: A Simple Soft-core Processor used in the COPPER Mission

Steve Massey
Electrical Engineering 2013
Saint Louis University