Project Cheesy Fingers: OpenCompute Hardware Hacking

Andrew Cencini, Bennington College
Steven White, Nebula
Agenda

- About Us
- About OpenCompute
- The Problem
- Solution Design
- The Process
- The Future
- Q&A
Agenda

- **About Us**
- About OpenCompute
- The Problem
- Solution Design
- The Process
- The Future
- Q&A
About Us

- **Andrew Cencini**, Computer Science Faculty, Bennington College (VT)
  - Nearly 20 years industry and academic work; Standing Chair of OCP Academic Project
  - Previously: SQL Server, MSN Search/Bing, Exchange, Rackable/SGI, Verdiem, ZT Systems, Nebula (currently)
  - “Software guy” – now “Hardware and software guy”
  - Why? Lots of experience in this area, super interesting, makes for good teaching

- **Steven White**, Nebula
  - 20+ years experience in electronics and embedded programming
  - 13+ years experience in data center management and monitoring
  - Previously: Rackable Systems/SGI, ZT Systems, Microsoft Research
  - Extensive knowledge in hardware design from prototype to high volume production
Agenda

- About Us
- **About OpenCompute**
- The Problem
- Solution Design
- The Process
- The Future
- Q&A
About OpenCompute

- Facebook, Rackspace, others
- Open data center design
  - Motherboards
  - Servers
  - Racks
  - Networking
  - Facilities
About OpenCompute

Industry Standard

Open Compute Project

1.9
PUE

1.07
PUE
June 2011 – Summit I (Facebook HQ)

- 200 Participants
- 3 Contributions (all from Facebook)
- 38% More energy efficient
- 24% Lower cost
October 2011: Summit II (NYC)

- **350 Participants**
- **Launched foundation w/ Intel, Rackspace, Goldman Sachs, and Andy Bechtolsheim**
- **Defined principles, mission statement, project procedures**
- **200 Participants**
- **3 Contributions (all from Facebook)**
- **38% More energy efficient**
- **24% Lower cost**
May 2012: Summit III (San Antonio)

350 Participants
Launched foundation w/ Intel, Rackspace, Goldman Sachs, and Andy Bechtolsheim
Defined principles, mission statement, project procedures

200 Participants
3 Contributions (all from Facebook)
38% More energy efficient
24% Lower cost

500 Participants
Launched Open Rack and Open Vault
HP + Dell demo compatible designs
January 2013: Summit IV (Santa Clara)

- 500 Participants
- Launched Open Rack
- Open Vault
- HP + Dell demo compatible designs

- 350 Participants
- Defined principles, mission statement, project procedures

- 200 Participants
- 3 Contributions (all from Facebook)
- 38% More energy efficient
- 24% Lower cost

- 1500+ Participants
- 50+ Official members
- Dozens of contributions, including Group Hug, silicon photonics, and SOCs

- 500 Participants
- Launched Open Rack and Open Vault
- HP + Dell demo compatible designs

- 200 Participants
- Defined principles, mission statement, project procedures

- 1500+ Participants
- 50+ Official members
- Dozens of contributions, including Group Hug, silicon photonics, and SOCs
OpenCompute

- Strong vendor support
- Other ways of engaging community?

- **OCP Hackathon #1** (Summit IV: Santa Clara)
  - 12 hours
  - Support from GrabCAD, Upverter
  - Winning entry: Provisional Patent Application funded by OCP

- **OCP Hackathon #2** (Facebook HQ – June 2013)
  - 12 hours
  - Support from TechShop, GrabCAD, Upverter, SKTA Innopartners
  - Winning entry: $10,000
OCP Hackathon
OCP Hackathon
Agenda

- About Us
- About OpenCompute
- **The Problem**
- Solution Design
- The Process
- The Future
- Q&A
The Problem

- Data Center Management
  - “Crash cart” – used to debug failed machines
  - In OCP data centers (e.g. Facebook), tech used “donglezilla” to get POST codes, other diagnostics from motherboard
  - 14-pin debug header as part of OCP Motherboard spec

- How can we simplify this and scale it out?

- Can we make it so that we can debug without having to touch the machine?

The OCP Debug Header

<table>
<thead>
<tr>
<th>CPU (Channels 0 &amp; 1)</th>
<th>Code</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU0</td>
<td>A0</td>
<td>CPU 0 channel o DIMM o (furthest) failure</td>
</tr>
<tr>
<td></td>
<td>A1</td>
<td>CPU 0 channel 0 DIMM 1 failure</td>
</tr>
<tr>
<td></td>
<td>A2</td>
<td>CPU 0 channel 1 DIMM 0 failure</td>
</tr>
<tr>
<td></td>
<td>A3</td>
<td>CPU 0 channel 1 DIMM 1 (closest) failure</td>
</tr>
<tr>
<td>CPU0 (Channels 2 &amp; 3)</td>
<td>A4</td>
<td>CPU 0 channel 2 DIMM 0 (furthest) failure</td>
</tr>
<tr>
<td></td>
<td>A5</td>
<td>CPU 0 channel 2 DIMM 1 failure</td>
</tr>
<tr>
<td></td>
<td>A6</td>
<td>CPU 0 channel 3 DIMM 0 failure</td>
</tr>
<tr>
<td></td>
<td>A7</td>
<td>CPU 0 channel 3 DIMM 1 (closest) failure</td>
</tr>
<tr>
<td>CPU1 (Channels 0 &amp; 1)</td>
<td>B0</td>
<td>CPU 1 channel o DIMM o (furthest) failure</td>
</tr>
<tr>
<td></td>
<td>B1</td>
<td>CPU 1 channel 0 DIMM 1 failure</td>
</tr>
<tr>
<td></td>
<td>B2</td>
<td>CPU 1 channel 1 DIMM 0 failure</td>
</tr>
<tr>
<td></td>
<td>B3</td>
<td>CPU 1 channel 1 DIMM 1 (closest) failure</td>
</tr>
<tr>
<td>CPU1 (Channel 2 &amp; 3)</td>
<td>B4</td>
<td>CPU 1 channel 2 DIMM 0 (furthest) failure</td>
</tr>
<tr>
<td></td>
<td>B5</td>
<td>CPU 1 channel 2 DIMM 1 failure</td>
</tr>
<tr>
<td></td>
<td>B6</td>
<td>CPU 1 channel 3 DIMM 0 failure</td>
</tr>
<tr>
<td></td>
<td>B7</td>
<td>CPU 1 channel 3 DIMM 1 (closest) failure</td>
</tr>
</tbody>
</table>

Figure 14 Efficient Performance Motherboard DIMM Error Code Table

<table>
<thead>
<tr>
<th>Pin (CKT)</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Low HEX character [0] least significant bit</td>
</tr>
<tr>
<td>2</td>
<td>Low HEX character [1]</td>
</tr>
<tr>
<td>3</td>
<td>Low HEX character [2]</td>
</tr>
<tr>
<td>4</td>
<td>Low HEX character [3] most significant bit</td>
</tr>
<tr>
<td>5</td>
<td>High HEX character [0] least significant bit</td>
</tr>
<tr>
<td>6</td>
<td>High HEX character [1]</td>
</tr>
<tr>
<td>7</td>
<td>High HEX character [2]</td>
</tr>
<tr>
<td>8</td>
<td>High HEX character [3] most significant bit</td>
</tr>
<tr>
<td>9</td>
<td>Serial transmit (motherboard transmit)</td>
</tr>
<tr>
<td>10</td>
<td>Serial receive (motherboard receive)</td>
</tr>
<tr>
<td>11</td>
<td>System reset</td>
</tr>
<tr>
<td>12</td>
<td>Serial console select (1=SOL, 0=local)</td>
</tr>
<tr>
<td>13</td>
<td>GND</td>
</tr>
<tr>
<td>14</td>
<td>VCC (+5VDC)</td>
</tr>
</tbody>
</table>

Figure 23 Debug Header Pin Definitions

Figure 22 Debug Header
Agenda

- About Us
- About OpenCompute
- The Problem
- **Solution Design**
- The Process
- The Future
- Q&A
Solution Design

- **Solution #1: “Cheesy Fingers”**
  - “Be able to run a data center from your recliner, with a beer and a bag of cheetos – only needing to wipe the cheetos cheese off of your fingers to diagnose problems.”
  - “If necessity is the mother of invention, then laziness must be its father.” (Carl Stoddard)
  - Small module that plugs into the debug header port on OCP motherboards, and uses a low-power 802.15.4 wireless mesh network to send and receive data:
    - POST codes
    - Reset server
    - Console access
    - Sensor data
    - MRF24J40, PIC18, I2C x 2, 14-pin debug header connector

- Winning entry at OCP Hackathon #1 (January 2013)
Solution Design

- Solution #2: “Big Cheese”, “CheeseMonger”, “EZ Cheese”
  - “Big Cheese” – Debug info aggregator
  - “CheeseMonger” – iPad application that talks to Big Cheese
  - “EZ Cheese” – Open protocol for talking between components

- Big Cheese – built as shroud on top of Raspberry Pi GPIO header
  - MRF24J40 and Bluetooth module

- CheeseMonger talks EZ Cheese over Bluetooth to Big Cheese
- Big Cheese talks EZ Cheese over 802.15.4 to Cheesy Fingers

- Winning entry at OCP Hackathon #2 (June 2013)
Agenda

- About Us
- About OpenCompute
- The Problem
- Solution Design
- **The Process**
- The Future
- Q&A
The Process

- Identify the Project Budget and Volume Production Target Cost
- Physical Shape and Size Restrictions
- Component Research and Identification
- Schematic Capture
- PCB Layout
- PCB Fabrication
- PCB Assembly
- Firmware Development
- Testing and Revisions
Identify the Project Budget and Volume Production Target Cost

- Price tools and parts for development (compiler, programmers, components, PCBs, etc.)
- Ensure the end cost of the finished product is attractive for the application
- Don’t forget to include contract manufacturer assembly fees and quality control costs in the total cost
Physical Shape and Size Restrictions

- Work closely with mechanical engineers to identify physical restraints and optimal wire routing paths
- Download 3D models of component packages and connectors
Component Research and Identification

- Read Datasheets
- Design around readily available components that are stocked at multiple vendors by searching with octopart.com and eciaauthorized.com
- Keep the bill of materials cost in mind when making decisions
Schematic Capture

- Free offerings from some PCB houses
- Create symbols for parts that aren’t in existing libraries
PCB Layout

- Create component footprints
- Position components for easiest routing
- Minimize layer count for cost savings
- Try to avoid less than 6 mil trace width and spacing
- Keep assembly in mind as components are placed
PCB Fabrication

- Advanced Circuits (4pcb.com) – Bare Bones (no soldermask or silkscreen), $33 each 2 layer (min. 4, 5 day), $66 each 4 layer (min. 4, 5 day)
- PCBPool.com – Low prices and a free desktop stencil for prototype orders (price varies, 1-8 day turn, 2 day overseas shipping)
- Sunstone Circuits (sunstone.com) – ValueProto (up to 2 weeks, free ground shipping)
- Zephyrtronics – Syringes of solder paste
PCB Assembly

- Surface mount technology is still within the grasp of the electronics hobbyist and hacker
- Inexpensive tools, a steady hand, and a good set of eyes are required
Stencil Printing
Pick and Place
Reflow Oven
Firmware/Software Design
Firmware/Software Design

- **Cheesy Fingers Firmware**
  - Written in C using low-level 802.15.4 via SPI – standard packet; custom payload
  - Talks to debug port (e.g. reset, read POST code, etc.)
  - On power-up, broadcasts on PAN to find Big Cheese for rack, caches BC address.
  - Receives commands, sends responses

- **Big Cheese Software**
  - Written in C using WiringPi for GPIO ops
  - One startup, broadcasts on PAN to get all Cheesy Fingers IDs
  - Uses asynchronous serial comm. to receive commands from Bluetooth / send response
  - Uses SPI to send and receive data via 802.15.4
  - Would have liked to use MiWi; no impl for ARM – this could change
Firmware / Software Design

- EZ Cheese 802.15.4 Protocol: **Packet**

```
<table>
<thead>
<tr>
<th>Byte</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5..6</th>
<th>7..8</th>
<th>9..10</th>
<th>11..n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Desc</td>
<td>Header length</td>
<td>Frame length</td>
<td>Frame control</td>
<td>Frame control</td>
<td>Sequence number</td>
<td>PAN_ID (LSB first)</td>
<td>Dest address</td>
<td>Source address</td>
<td>Payload</td>
</tr>
<tr>
<td></td>
<td>(ignored)</td>
<td>(header len +</td>
<td>LSB</td>
<td>MSB</td>
<td>number</td>
<td>(LSB first)</td>
<td>(LSB first)</td>
<td>(LSB first)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>payload len)</td>
<td>Always 0x41</td>
<td>Always 0x88</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(data packet,</td>
<td>(short src and</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>compress</td>
<td>dest addr present)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

- EZ Cheese 802.15.4 Protocol: **Payload**

```
<table>
<thead>
<tr>
<th>Byte</th>
<th>0</th>
<th>1..n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Desc</td>
<td>Command Id</td>
<td>Optional – command data</td>
</tr>
</tbody>
</table>
```
Firmware / Software Design

- **EZ Cheese 802.15.4 Protocol:** **Commands**

<table>
<thead>
<tr>
<th>Command Id / Name</th>
<th>Request</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00 : CMD_GET_BCID</td>
<td>No Command Data. Broadcast by CF nodes to locate Big Cheese for the current PAN.</td>
<td>No Command Data. Empty response by Big Cheese directly to the requester – requester caches Big Cheese source address for future use.</td>
</tr>
<tr>
<td>0x01 : CMD_GET_CFID</td>
<td>No Command Data. Broadcast by BC node to locate all CF nodes in the current PAN.</td>
<td>No Command Data. Empty response by each Cheesy Finger to the Big Cheese – Big Cheese caches source address in list of active CF nodes in PAN.</td>
</tr>
<tr>
<td>0x02 : CMD_RETRIEVE_DATA</td>
<td>Command Data: ID of Sensor/Action to trigger. Additional bytes: TBD.</td>
<td>None – data is consumed by receiver and a CMD_SEND_DATA response is sent later, once data is sent.</td>
</tr>
<tr>
<td>0x03 : CMD_SEND_DATA</td>
<td>Command Data: ID of Sensor/Action triggered, length of response, followed by response data if response length &gt; 0.</td>
<td>None – data is consumed by receiver.</td>
</tr>
</tbody>
</table>
Firmware / Software Design

- CheeseMonger
  - iPad app, uses Bluetooth and text-based EZ Cheese protocol

- Request/Response BNF
  - `;;;<CMD>$$`
  - `<CMD> ::= <CMDID>[<DATA>]`
  - `<CMDID> ::= {A..F}`
  - `<DATA> ::= @@<value>[<DATA>]`

- Example:
  - **CMD8: Get CF OCP POST Code**
  - **SEND:** `;;;8@@0000$$`
  - **RETURNS:** `;;;8@@0000@@00$$`

  - 0000 is the address of the CF to get OCP POST Code from.
  - 00 is the OCP POST code in hex.
Results!
Agenda

- About Us
- About OpenCompute
- The Problem
- Solution Design
- The Process
- **The Future**
- Q&A
The Future

- Formalize and contribute to OCP, GitHub
- Testing, refinement & adoption

- Educational adaptation
  - Davis Educational Foundation Grant / 300ac Wireless Mesh Network
  - Modify Cheesy Fingers & Big Cheese (866MHz vs 2.4 GHz)
  - Build enclosure (3D print + PVC pipe)
  - Design antenna
  - Monitoring & measurement

- Many more ideas
Agenda

- About Us
- About OpenCompute
- The Problem
- Solution Design
- The Process
- The Future
- Q&A
THANK YOU

- **Thank you** for being a part of this presentation.

- Questions?

- Andrew: [acencini@bennington.edu](mailto:acencini@bennington.edu)

- Steve: [steven.white@nebula.com](mailto:steven.white@nebula.com)
Appendix: Hardware Hacking URLs

- Vacuum Pickup Tool
- Tweezers
- Dental Pick Set
- Toaster Oven Reflow
- Syringes of Lead-Free Solderpaste
- Plunger for syringe (Required; Power Palm Plunger Recommended)
- Whitepaper on using stencils
- Prototype PCBs
  - Advanced Circuits - http://www.4pcb.com
  - Sunstone Circuits - http://www.sunstone.com
Appendix: Hardware Hacking URLs

- Solder Paste Stencils
  - Sunstone Circuit - http://www.sunstone.com - Order with prototype PCBs
  - ExpressPCB - http://expresspcb.com/index.htm

- Free PCB Layout Software
  - Upverter - https://upverter.com/
  - ExpressPCB - http://expresspcb.com/ExpressPCBhtm/Free_cad_software.htm