Lucian Codrescu
Sr. Director, Technology
Qualcomm Technologies, Inc.

Qualcomm Hexagon DSP: An architecture optimized for mobile multimedia and communications
Hexagon™ DSP processors in Snapdragon products

- **aDSP**: Real-time media & sensor processing
  - Hexagon™ aDSP

- **mDSP**: Dedicated modem processing
  - Hexagon™ mDSP

- **Snapdragon 800**
  - Multimedia Fabric
  - System Fabric
  - Fabric & Memory Controller
  - LPDDR3
  - LPDDR3

- Components:
  - Camera
  - Display
  - JPEG
  - Video
  - Other
  - Adreno GPU
  - Krait CPU
  - Krait CPU
  - Krait CPU
  - 2MB L2
  - Audio
  - Sensors
  - Misc. Connectivity
  - Modem
Expansion of Hexagon DSP use cases beyond audio

Hexagon DSP is evolving for use beyond voice and audio to computer vision, video and imaging features.
The Hexagon DSP evolution

Generational improvements in performance and power efficiency driven by both architecture and implementation

<table>
<thead>
<tr>
<th>Version</th>
<th>Technology</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1</td>
<td>65nm</td>
<td>Oct 2006</td>
</tr>
<tr>
<td>V2</td>
<td>65nm</td>
<td>Dec 2007</td>
</tr>
<tr>
<td>V3M</td>
<td>45nm</td>
<td>June 2009</td>
</tr>
<tr>
<td>V3L</td>
<td>45nm Nov</td>
<td>2009</td>
</tr>
<tr>
<td>V3C</td>
<td>45nm Aug</td>
<td>2009</td>
</tr>
<tr>
<td>V4M</td>
<td>28nm Dec</td>
<td>2010</td>
</tr>
<tr>
<td>V4L</td>
<td>28nm Apr</td>
<td>2011</td>
</tr>
<tr>
<td>V4C</td>
<td>28nm Dec</td>
<td>2010</td>
</tr>
<tr>
<td>V5A</td>
<td>28nm Dec</td>
<td>2012</td>
</tr>
<tr>
<td>V5H</td>
<td>28nm Dec</td>
<td>2012</td>
</tr>
</tbody>
</table>
Key characteristics of modem & multimedia applications

Requirements
- Require fixed real-time performance level (fps, Mbit/sec, etc.)
- Extremely aggressive power & area targets

Characteristics
- Mix of signal processing & control code
  - For modem, Qualcomm does not use a split CPU/DSP architecture. All processing is done on Hexagon DSP
  - Multimedia apps have significant control in the RTOS & frameworks
- Heavy L2$ misses
  - Multimedia is data intensive
  - Modem is code intensive
Hexagon DSP blends features targeted to modem & multimedia

**VLIW**
- Need multi-issue to meet performance
- Low complexity for Area & Power

**Multi-Threading**
- To reduce L2$ miss penalty without the need for a large L2
- Increases instructions/VLIW packet because compiler doesn’t need to schedule latency

**Innovate in ISA to maximize IPC**
- More work/VLIW packet reduces energy/instruction
- Keep the pipelines full for MIPS/mm2
- Target both Signal Processing & Control code
VLIW: Area & power efficient multi-issue

- Dual 64-bit execution units
- Standard 8/16/32/64bit data types
- SIMD vectorized MPY / ALU / SHIFT, Permute, BitOps
- Up to 8 16b MAC/cycle
- 2 SP FMA/cycle

- Unified 32x32bit General Register File is best for compiler.
- No separate Address or Accum Regs
- Per-Thread

Variable sized instruction packets (1 to 4 instructions per Packet)

Dual 64-bit load/store units
Also 32-bit ALU

Device DDR Memory

Dual 64-bit execution units
Standard 8/16/32/64bit data types
SIMD vectorized MPY / ALU / SHIFT, Permute, BitOps
Up to 8 16b MAC/cycle
2 SP FMA/cycle

Unified 32x32bit General Register File is best for compiler.
No separate Address or Accum Regs
Per-Thread
Maximizing the signal processing code work/packet

Example from inner loop of FFT: Executing 29 “simple RISC ops” in 1 cycle

Vector 4x16-bit Add

Complex multiply with round and saturation

64-bit Load and 64-bit Store with post-update addressing

Zero-overhead loops

\{ R17:16 = MEMD(R0++M1)  
  MEMD(R6++M1) = R25:24  
  R20 = CMPY(R20, R8)<<1:rd:st  
  R11:10 = VADDH(R11:10, R13:12)  
\}:endloop0
Maximizing the control code work/packet

Hexagon DSP ISA improves control code efficiency over traditional VLIW

<table>
<thead>
<tr>
<th>Traditional VLIW Assembly Code</th>
<th>Hexagon DSP: Dot-New Predication</th>
<th>Hexagon DSP: Compound ALU</th>
<th>Hexagon DSP: New-Value Store</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. p0 = cmp.eq(r0,#0)</td>
<td>1. p0 = cmp.eq(r0,#0)</td>
<td>1. p0 = cmp.eq(r0,#0)</td>
<td>1. p0 = cmp.eq(r0,#0)</td>
</tr>
<tr>
<td>2. if (!p0) r2=memw(r0)</td>
<td>2. if (!p0) r2=memw(r0)</td>
<td>2. if (!p0) r2=memw(r0)</td>
<td>2. if (!p0) r2=memw(r0)</td>
</tr>
<tr>
<td>3. if (p0) jmp r31</td>
<td>3. if (p0.new) jmp r31</td>
<td>3. if (p0.new) jmp r31</td>
<td>3. if (p0.new) jmp r31</td>
</tr>
<tr>
<td>4. r2 = add(r2,#2)</td>
<td>4. r2 = add(r2,#2)</td>
<td>4. r1 = add(r1,r2)</td>
<td>4. r1 = add(r1,add(r2,#2))</td>
</tr>
<tr>
<td>5. r1 = add(r1,r2)</td>
<td></td>
<td>5. memw(r0) = r1</td>
<td>5. memw(r0) = r1.new</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6. jmp r31</td>
<td>6. jmp r31</td>
</tr>
</tbody>
</table>

Instr/Packet =
7 instr/5 packets = 1.4

Example C code

```c
void example(int *ptr, int val) {
    if (ptr!=0) {
        *ptr = *ptr + val + 2;
    }
}
```

Instr/Packet =
7 instr/2packets = 3.5
High avg. instructions/packet for targeted use cases

Compound instructions count as 2

Source: Qualcomm internal measurements
Programmer’s view of Hexagon DSP HW multi-threading

- Hexagon V5 includes three hardware threads
- Architected to look like a multi-core with communication through shared memory
Hexagon DSP V1-V4: Interleaved multi-threading

Simple round-robin thread scheduling

- Number of threads match execution pipe depth (three threads ➔ three execute stages)
- All instructions complete before next packet dispatch
- Compiler schedules for zero-latency which helps to increase instructions/VLIW packet
Hexagon DSP V5: Dynamic HW multi-threading

Recover some performance when threads idle or stalled

• Remove a thread from IMT rotation
  - On L2 cache misses
  - When in wait-for-interrupt or off mode

• Additional forwarding to support 2-cycle packets

• VLIW packets with dependencies between long latency instructions will stall
  - But many VLIW packets with simple instructions can complete in 2 processor clocks

Source: Qualcomm internal measurements
Hexagon DSP instructions per cycle

Source: Qualcomm internal measurements
Qualcomm Hexagon DSP architecture

Highly efficient mobile application processor—designed for more performance per MHz

Clock Rate (MHz) | 430-520 | 100-233 | 300-700
DSP Performance (BDTImark2000) | 4730-5720 | 1810-4220 | 5440-12660*

* - Projected best case score for 3-threads

Source: BDTI - For more detailed information see www.BDTI.com. All scores ©2013 BDTI
Hexagon DSP Power Benefits
MP3 playback power for competitive smartphones

- Power measured at the battery for various phones
- Includes everything: DSP, CPU, memory, analog components, etc

Source: Qualcomm internal measurements
Augmented Reality Java App finding objects in image using FastCV Feature Detect
Comparison of Feature Detect run on:
• App CPU (ARM/Neon)
• App DSP (Hexagon)

Table: Comparison of Feature Detect run on:

<table>
<thead>
<tr>
<th>Feature Detect Function</th>
<th>Detection Time (%)</th>
<th>Total Device Power (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM Only FastCV Library</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hexagon (QDSP6) FastCV Library</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Graphs:
- CPU Utilization (%)
- Detection Time (%)
- Total Device Power (%)

52% Less CPU
7% Less Time
32% Less Power*

Source: Qualcomm internal measurements. * Power measured at the device battery
Hexagon DSP power for different thread utilizations

- Excellent near-linear power scalability
  (as threads go idle, power used by the thread is nearly eliminated)
- Achieved through optimized clock tree design & clock gating

**Dhrystone Power, IMT Mode**

**FIR Power, IMT Mode**

Source: Qualcomm internal measurements
Hexagon DSP Software Development
Independent Algorithm Developers on Hexagon DSP
Announcing the **Hexagon DSP SDK**

*See the Hexagon DSP SDK in action at Uplinq2013* (www.uplinq.com)

Thank you

Follow us on:  

For more information on Qualcomm, visit us at:
www.qualcomm.com & www.qualcomm.com/blog