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

- **aDSP**: Real-time media & sensor processing
- **mDSP**: Dedicated modem processing

![Diagram of Snapdragon 800 with Hexagon™ DSP processors](image)
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
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-Threaded**
- 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

- Variable sized instruction packets (1 to 4 instructions per Packet)
- Dual 64-bit load/store units
- Also 32-bit ALU
- 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

Device DDR Memory
L2 Cache / TCM
Instruction Unit
Instruction Cache
Data Unit (Load/Store/ALU)
Data Unit (Load/Store/ALU)
Execution Unit (64-bit Vector)
Execution Unit (64-bit Vector)
Data Cache
Register File/Thread
Maximizing the signal processing code work/packet

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

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

Vector 4x16-bit Add

Zero-overhead loops
- Dec count
- Compare
- Jump top

Complex multiply with round and saturation

\[
\begin{align*}
R17:16 &= \text{MEMD}(R0++M1) \\
\text{MEMD}(R6++M1) &= R25:24 \\
R20 &= \text{CMPY}(R20, R8) :<:1 :\text{rnd}:\text{sat} \\
R11:10 &= \text{VADDH}(R11:10, R13:12)
\end{align*}
\]
Maximizing the control code work/packet

Hexagon DSP ISA improves control code efficiency over traditional VLIW

### Example C code

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

### Traditional VLIW Assembly Code

1. p0 = cmp.eq(r0,#0)
   
2. if (!p0) r2=memw(r0)
   
3. if (p0) jump:nt r31
   
4. r2 = add(r2,#2)

5. r1 = add(r1,r2)

   
6. memw(r0) = r1
   
7. jump r31

**Instr/Packet = 7 instr/5 packets = 1.4**

### Hexagon DSP: Dot-New Predication

1. p0 = cmp.eq(r0,#0)
   
2. if (!p0.new) r2=memw(r0)
   
3. if (p0.new) jump:nt r31
   
4. r2 = add(r2,#2)

5. r1 = add(r1,r2)

6. memw(r0) = r1
7. jump r31

### Hexagon DSP: Compound ALU

1. p0 = cmp.eq(r0,#0)
   
2. if (!p0.new) r2=memw(r0)
   
3. if (p0.new) jump:nt r31
   
4. r1 = add(r1,add(r2,#2))

5. memw(r0) = r1
6. jump r31

### Hexagon DSP: New-Value Store

1. p0 = cmp.eq(r0,#0)
   
2. if (!p0.new) r2=memw(r0)
   
3. if (p0.new) jump:nt r31
   
4. r1 = add(r1,add(r2,#2))

5. memw(r0) = r1.new
6. jump r31

**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
- Architectured 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

Thread 0 Dispatch
T0: { Ld, Ld, Add, Cmp }

Thread 1 Dispatch
T0: { Ld, Ld, Add, Cmp }
T1: { St, Ld, Mpy, Add }

Thread 2 Dispatch
T0: { Ld, Ld, Add, Cmp }
T1: { St, Ld, Mpy, Add }
T2: { Ld, Add, Jump }
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
Hexagon DSP V5: Efficient Architecture

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

Clock Rate (MHz)
- Mobile Competitor: 430-520
- Qualcomm Hexagon V5 (1 thread): 100-267
- Qualcomm Hexagon V5 (3 threads): 300-800

DSP Performance (BDTImark2000)
- Mobile Competitor: 4730-5720
- Qualcomm Hexagon V5 (1 thread): 1810-4840
- Qualcomm Hexagon V5 (3 threads): 5430-14520*

Source: BDTI - For more detailed information see www.BDTI.com. All scores ©2013 BDTI

* - Projected best case score for 3-threads
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
Computer vision offload – ARM/neon to Hexagon DSP

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)**

**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

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](http://www.uplinq.com))

**Hexagon DSP SDK**

- Hexagon C/C++ Compiler
- Hexagon Assembler
- Build Module
- Code Profiler
- Resource Analyzer
- Debug Monitor

**Tool**

- Dynamic Module(s)
- Fast CV Libraries
- Utility API Libraries
- Hexagon DSP-RTOS

**Code**

- Audio Plug-in
- Image Plug-in
- Vision Plug-in
- CPU Offload Plug-in

**Android HLOS**

- Android Examples
- System Test Framework

**Eclipse based Integrated Development Environment**


Qualcomm Technologies, Inc. All Rights Reserved
Thank you

Follow us on:  

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

©2013 Qualcomm Technologies, Inc.  
Qualcomm and Hexagon are trademarks of QUALCOMM Incorporated, registered in the United States and other countries.  All QUALCOMM Incorporated trademarks are used with permission.  Other product and brand names may be trademarks or registered trademarks of their respective owners.  Hexagon is a product of Qualcomm Technologies, Inc.