How to optimize application on Android x86

12th Kandroid Conference

인텔코리아 이 진 용 차장
SSG/DRD/APAC

Legal Notices and Disclaimers

- INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL® PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. INTEL PRODUCTS ARE NOT INTENDED FOR USE IN MEDICAL, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS.
- Intel may make changes to specifications and product descriptions at any time, without notice.
- All products, dates, and figures specified are preliminary based on current expectations, and are subject to change without notice.
- Intel, processors, chipsets, and desktop boards may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.
- Any code names featured are used internally within Intel to identify products that are in development and not yet publicly announced for release. Customers, licensees and other third parties are not authorized by Intel to use code names in advertising, promotion or marketing of any product or services and any such use of Intel's internal code names is at the sole risk of the user.
- Intel product plans in this presentation do not constitute Intel plan of record product roadmaps. Please contact your Intel representative to obtain Intel's current plan of record product roadmaps.
- Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more information go to http://www.intel.com/performance
- Intel, Intel Inside, the Intel logo, Centrino, Intel Core, Intel Atom, Pentium and Ultrabook are trademarks of Intel Corporation in the United States and other countries.
- Material in this presentation is intended as product positioning and *not* approved end user messaging.
- This document contains information on products in the design phase of development.
- *Other names and brands may be claimed as the property of others.
-

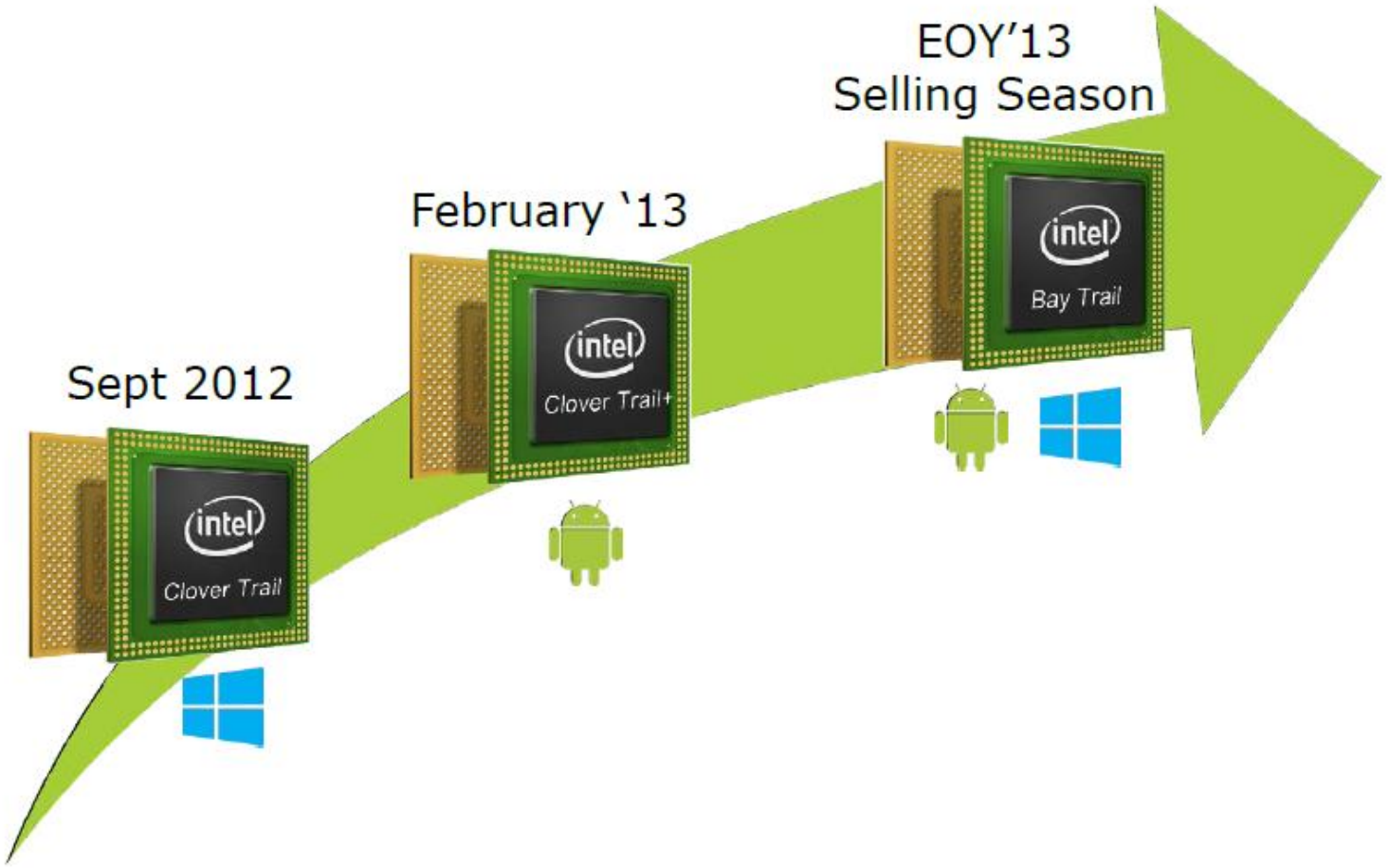
Legal Notices and Disclaimers, cont.

- Hyper-Threading Technology requires a computer system with a processor supporting HT Technology and an HT Technology-enabled chipset, BIOS and operating system. Performance will vary depending on the specific hardware and software you use. For more information including details on which processors support HT Technology, see here
- Requires an Intel® Wireless Display enabled PC, TV Adapter, and compatible television. Available on select Intel® Core processors. Does not support Blu-Ray or other protected content playback. Consult your PC manufacturer. For more information, see www.intel.com/go/wirelessdisplay
- (Built-in Visuals) Available on the 2nd gen Intel® Core™ processor family. Includes Intel® HD Graphics, Intel® Quick Sync Video, Intel® Clear Video HD Technology, Intel® InTru™ 3D Technology, and Intel® Advanced Vector Extensions. Also optionally includes Intel® Wireless Display depending on whether enabled on a given system or not. Whether you will receive the benefits of built-in visuals depends upon the particular design of the PC you choose. Consult your PC manufacturer whether built-in visuals are enabled on your system. Learn more about built-in visuals at http://www.intel.com/technology/visualtechnology/index.htm.
- Intel® Insider™ is a hardware-based content protection mechanism. Requires a 2nd generation Intel® Core™ processor-based PC with built-in visuals enabled, an Internet connection, and content purchase or rental from qualified providers. Consult your PC manufacturer. For more information, visit www.intel.com/go/intelinsider.
- Security features enabled by Intel® AMT require an enabled chipset, network hardware and software and a corporate network connection. Intel AMT may not be available or certain capabilities may be limited over a host OS-based VPN or when connecting wirelessly, on battery power, sleeping, hibernating or powered off. Setup requires configuration and may require scripting with the management console or further integration into existing security frameworks, and modifications or implementation of new business processes. For more information, see http://www.intel.com/technology/manage/iamt.
- No system can provide absolute security under all conditions. Requires an enabled chipset, BIOS, firmware and software and a subscription with a capable Service Provider. Consult your system manufacturer and Service Provider for availability and functionality. Intel assumes no liability for lost or stolen data and/or systems or any other damages resulting thereof. For more information, visit http://www.intel.com/go/anti-theft
- Requires an Execute Disable Bit enabled system. Check with your PC manufacturer to determine whether your system delivers this functionality. For more information, visit http://www.intel.com/technology/xdbit/index.htm

Legal Notices and Disclaimers, cont.

- Intel® vPro™ Technology is sophisticated and requires setup and activation. Availability of features and results will depend upon the setup and configuration of your hardware, software and IT environment. To learn more visit: http://www.intel.com/technology/vpro
- The original equipment manufacturer must provide TPM functionality, which requires a TPM-supported BIOS. TPM functionality must be initialized and may not be available in all countries.
- Intel® AES-NI requires a computer system with an AES-NI enabled processor, as well as non-Intel software to execute the instructions in the correct sequence. AES-NI is available on select Intel® processors. For availability, consult your reseller or system manufacturer. For more information, see http://software.intel.com/en-us/articles/intel-advanced-encryption-standard-instructions-aes-ni/
- No system can provide absolute security under all conditions. Requires an Intel IPT enabled system, including a 2nd generation Intel Core processor, enabled chipset, firmware, and software. Available only on participating websites. Consult your system manufacturer. Intel assumes no liability for lost or stolen data and/or systems or any resulting damages. For more information, visit http://www.ipt.intel.com

Accelerating Intel Atom Processor Based Tablets



Tablets with Intel Inside - 2013

ASUS* MeMO Pad FHD 10”
(Z2560)

ASUS Fonepad* 7”
(Z2420)

Samsung* Galaxy* Tab 3 10.1”
(Z2560)

New Platform for a Great Mobile Experience

Overview: Intel Atom Process Z3000 Series

Performance, Quality and Trust

Quad Core w/ Intel® Burst Technology 2.0

22nm

Intel level performance delivers amazing media experiences

Advanced Imaging and Video

Video capture at 1080p with full HD playback

Advanced 8MP HD camera

Media encode/ decode hardware acceleration

Integrated Image Signal Processor

Powerful and Vibrant Visuals

Intel® HD graphics

Dynamic Graphics Frequency

Open GL* ES 2.0, 1.1 + Extensions

HW acceleration for HTML 5

Display up to 25x16

Wireless, Dual Monitor Display

Great Software Support

Top Google* Play* applications optimized for Intel® Architecture

Comprehensive porting, analysis and optimization tools for developers

Great Mobile Experiences

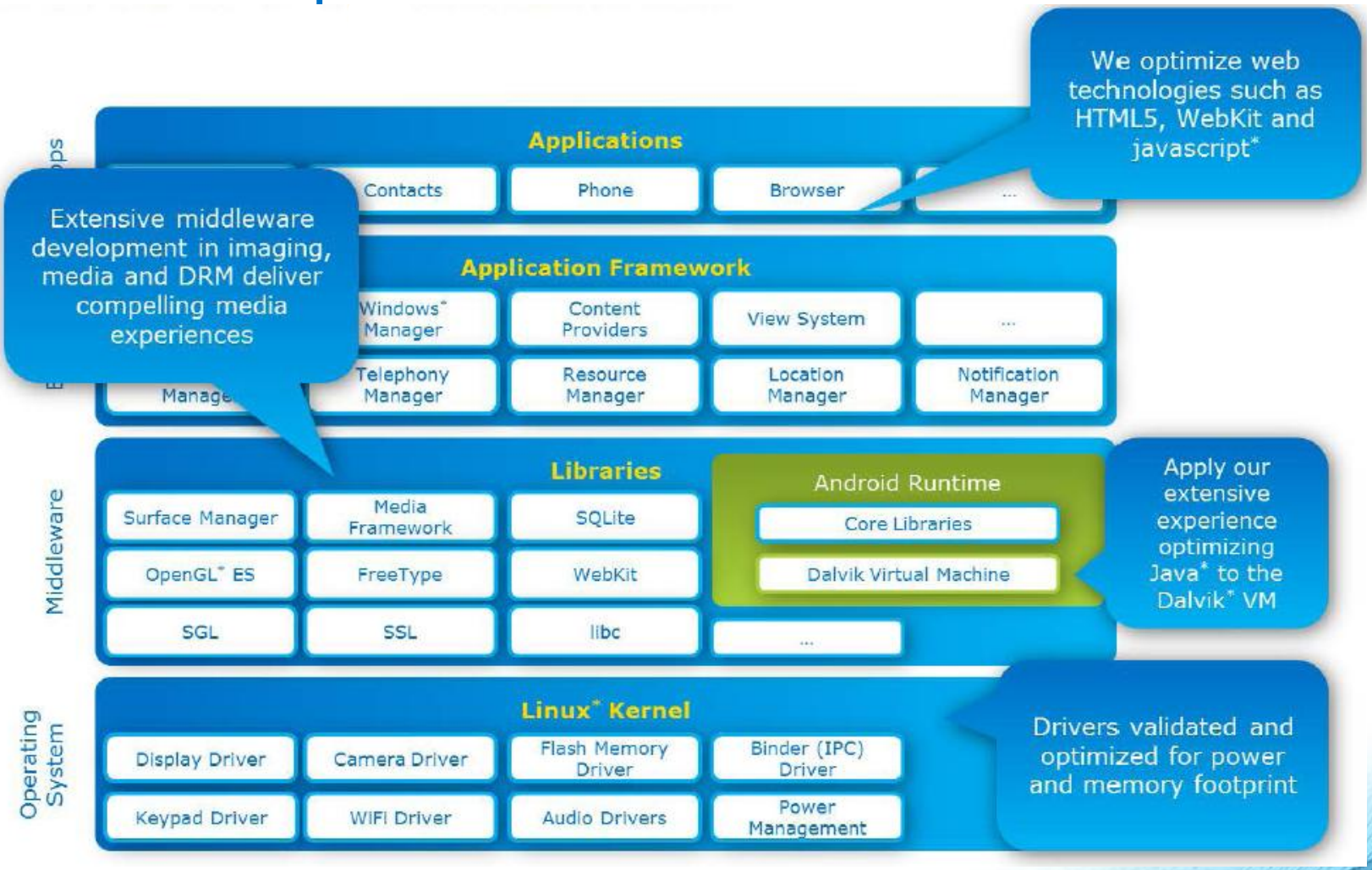
Extraordinary battery life
>8 hrs video playback‡
>3 weeks standby‡
3G/ 4G LTE

<1.3 Pounds†

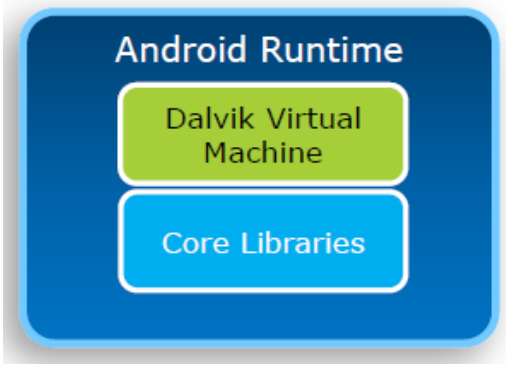
<9 mm Tablets

Fast and Fluid User Experience for Apps, Games, Browsing, Photos, Videos…

Software Optimizations



NDK vs. Dalvik Applications : Intel Platforms Support



Android Dalvik apps

- These will "just work"

Android native (NDK) apps

- Most will run w/o any recompilation…
- Android NDK provides an x86 toolchain since 2011
- A simple recompile using the Android NDK yields the best performance
- If there is specific processor dependent code, porting may be necessary

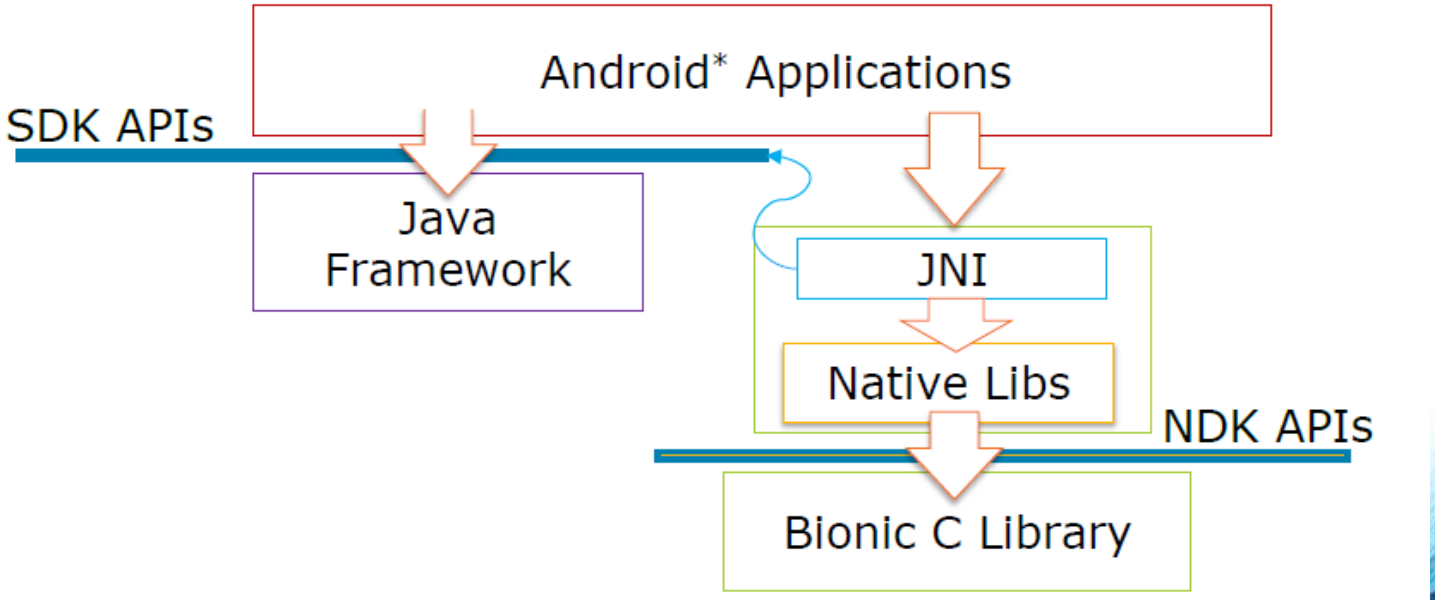
Top Android Market apps

- Intel validates and tests on Intel Atom processor platforms
- (Mix of Dalvik and NDK apps)

NDK Application Development

APP_ABI := all
or APP_ABI := x86

through jni



Packaging APKs for Multiple CPU architectures

Tow options:

- One package for all (“fat binary”)
  - Embed native libraries for each architecture in one APK
  - Easiest and preferred way to go
- Multiple APKs
  - One APK per architecture
  - If you have good reasons to do so (i.e., your fat binary APK would be larger than 50MB)

Fat Binaries

By default, an APK contains libraries for every supported ABIs.

The application will be filtered during installation (after download)

GCC flags

```
ifeq ($(TARGET_ARCH_ABI), x86)
LOCAL_CFLAGS += -O3 –ffast-math –mtune=atom –msse3 –mfpmath=sse
else
LOCAL_CFLAGS += …
```

To optimize for Intel Silvermont Microarchitecture (not available with current NDK r9 standard toolschains) :

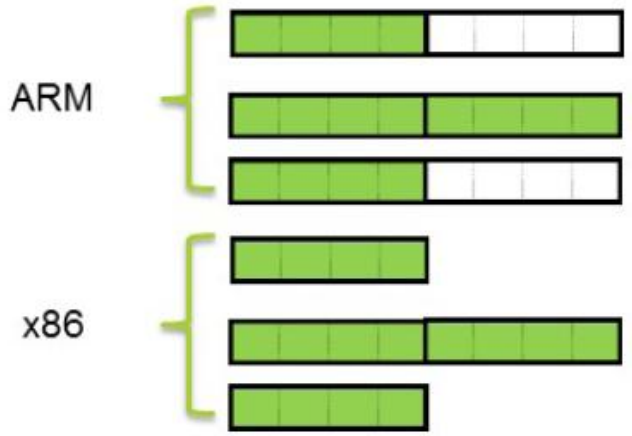
```
LOCAL_CFLAGS += -O3 –ffast-math –mtune=slm –msse4.2 –mfpmath=sse
```

- ffast-math influence round-off of fp arithmetic and so breaks strict IEEE compliance
- The other optimization are totally safe
- Add –ftree-vectorizer-verbose to get a vectorization report

Memory Alignment

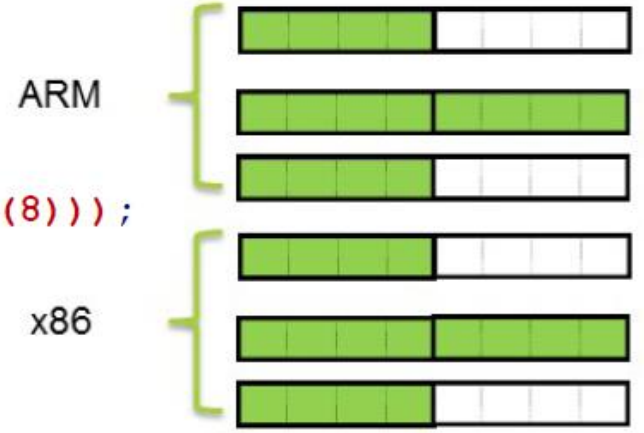
By default

```
struct TestStruct {
  int mVar1;
  long long mVar2;
  int mVar3;
};
```

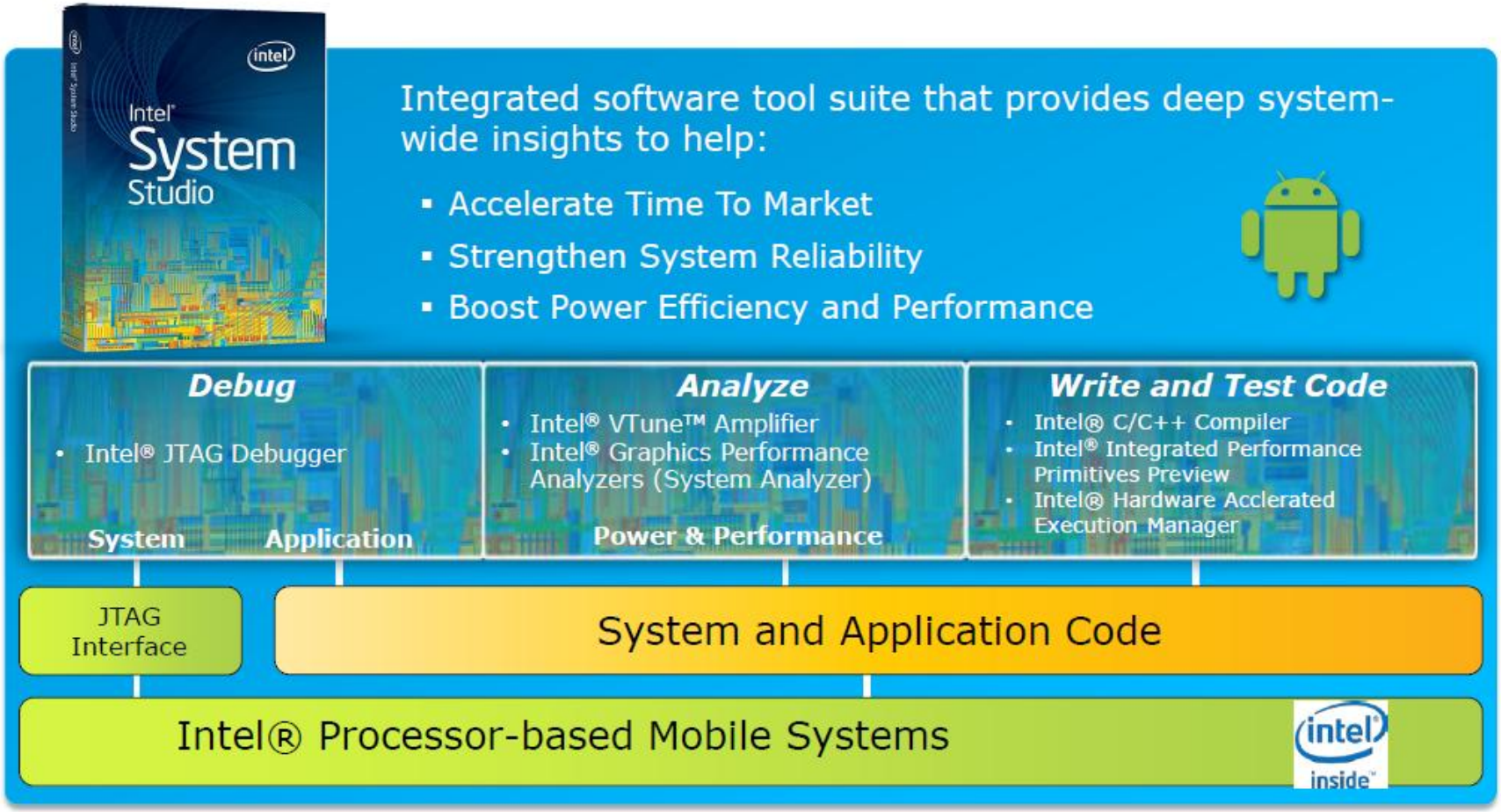


Easy fix

```
struct TestStruct {
  int mVar1;
  long long mVar2 __attribute__ ((aligned(8)));
  int mVar3;
};
```



Intel System Studio for Android Overview



Beacon Mountain Preview v0.5

- Comprehensive Android Development Environment
- For ARM & Intel Atom processor based Smartphones, Tablets
- Tools for design, code, debug & app acceleration
- Helps shorten the Android App Development Cycle

Intel Tools	Third-Party Tools
Intel® Hardware Accelerated Execution Manager Intel® Graphics Performance Analyzers System Analyzer Intel® Integrated Performance Primitives Preview Intel® Threading Building Blocks Intel® Software Manager	Google Android SDK (ADT Bundle) Android NDK Eclipse Integrated Development Environment Android Design Cygwin* (for Windows operating systems)

Free download at
http://software.intel.com/en-us/vcsource/tools/beaconmountain

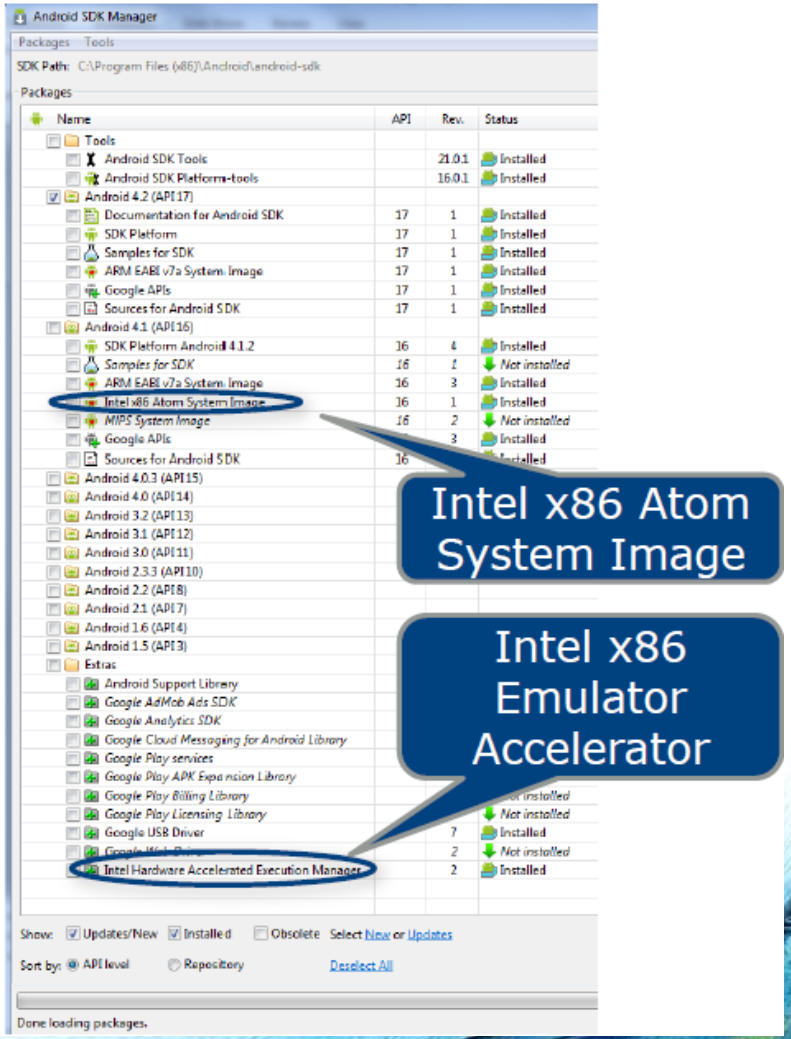
Faster Android Emulation on Intel Architecture Based Host PC

Pre-built Intel Atom Processor Images

- Android SDK manager Has x86 emulation images built-in
- To emulate an Intel Atom Processor based Android phone, install the "Intel Atom x86 System Image" available in the Android SDK Manager

Much Faster Emulation

- Intel Hardware Accelerated Execution Manager (Intel HAXM) for Mac and Windows uses Intel Virtualization Technology (Intel VT) to accelerate Android emulator
- Intel VT is already supported in Linux by KVM



Intel Graphics Performance Analyzer (Intel GPA): System Analyzer

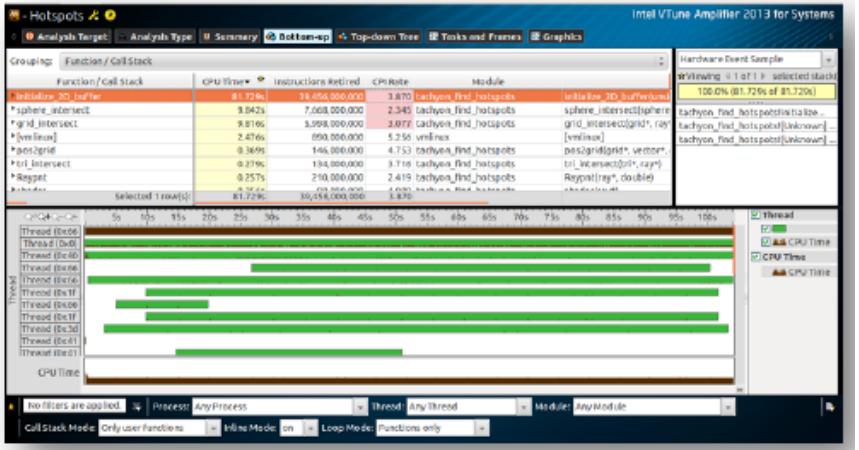
- Profiles performance and Power
- Real-time charts of CPU, GPU and power metrics
- Conduct real-time experiments with OpenGL-ES(with state overrides to help narrow down problems
- Triage system-level performance with CPU, GPU and Power metrics

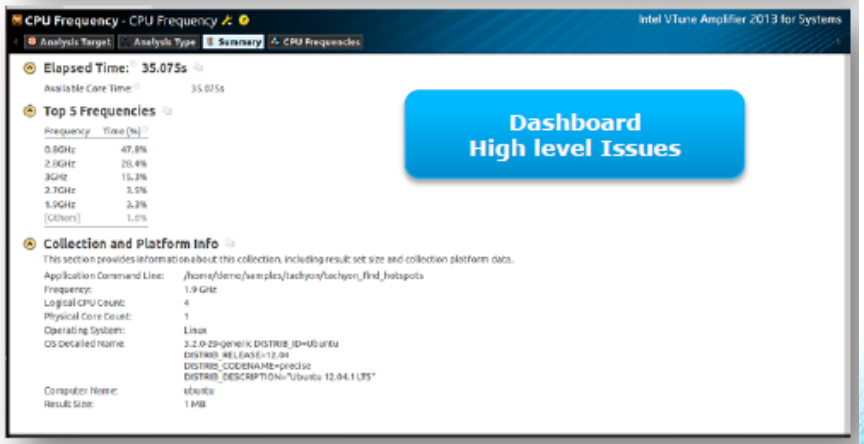
Available freely on
http://intel.com/software/gpa

Intel VTune Amplifier for Android

Performance profiler – Analyzes CPU Performance using PMU

- Event-based sampling for tuning platform performance
- Analyzes hardware events and call stacks
- Identifies Intel® Architecture bottlenecks
  - Cache Misses
  - Branch Mispredictions
  - Important CPU Metrics
- Provides statistical call counts for better data for in-lining decisions
- Powerful filtering to quickly identify cause of performance issues
- Drill-down to the Source Code





Advanced profiling to boost performance

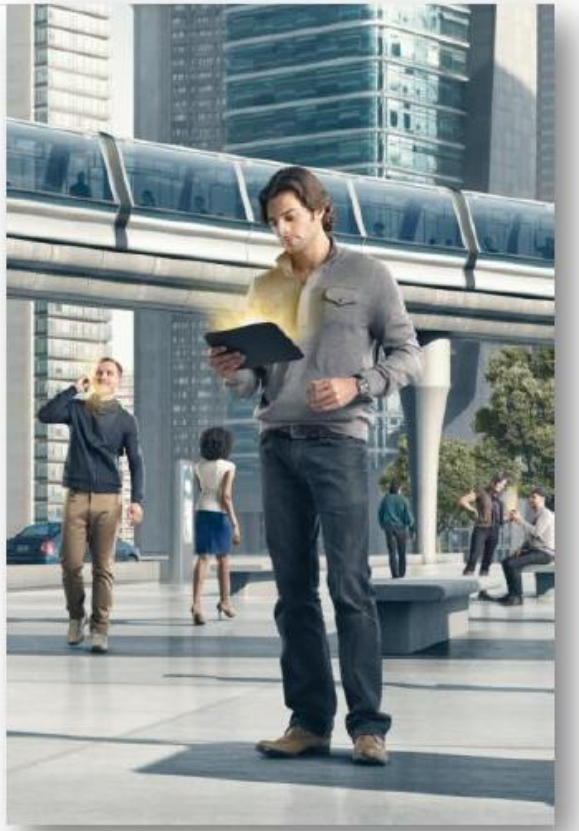
Intel C++ Compiler for Android
Optimized System and Application Performance

High Performance

- Boosts performance with a single recompile
- Binary and source compatible with GNU* GCC compiler
- Intel® Cilk™ Plus - task and data parallelism
- Standards support

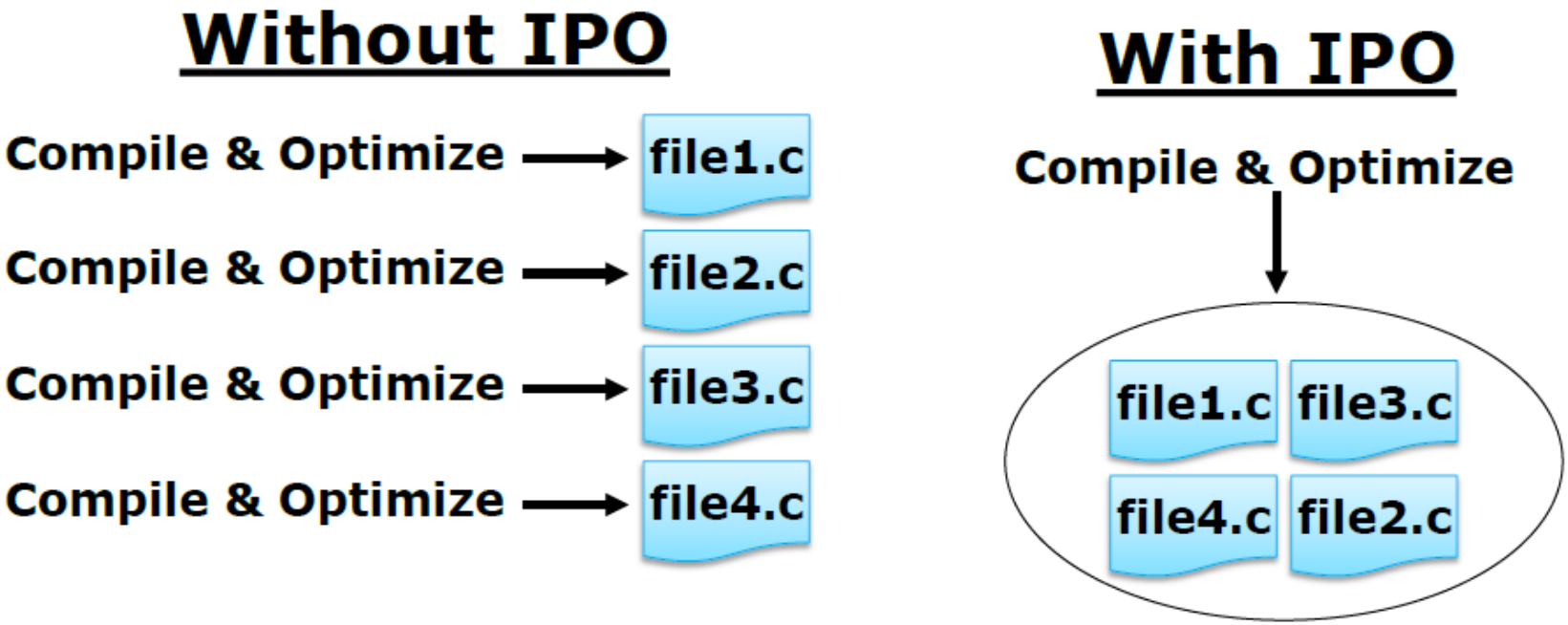
Optimization

- Intel® Atom™ - optimized common libc/libm functions
- Intel Silvermont microarchitecture support
- Vectorization for loops - SIMD
- Interprocedural optimization (IPO)
- Profile guided optimization (PGO)



High performance. GNU compatibility. Standard support.

World Class Leading Compiler Technology
Interprocedural Optimization(IPO)



`-ip`	Only between modules of one source file
`-ipo`	Modules of multiple files/whole application

Intel Integrated Performance Primitives

- **Highly Optimized Building Blocks -** for image processing, signal processing, multimedia, audio, video, speech, computer vision.
- **Applicable Domains -** Several domains contain the hand-tuned functions for Intel® Atom™ processors:
  - Signal processing and data compression, speech coding
  - Image processing, color conversion, frame processing
- **Latest Instruction Support –** Optimizations using the latest instructions for the targeted processor or compiler optimization

Highly Optimized Functions	Cross-Platform & Operating System
Key Function Domains • Image/Frame Processing • Signal Processing **Optimization Goals** • Raw Performance • Platform Throughput • Power Efficiency	**Operating Systems** • Windows* • Linux* • OS X* • Android* **Platforms** • Phone (Intel Atom processors) • Tablet (Intel Atom and Intel® Core™ processors) • Ultrabook™/PC (Intel Core processors) • Server (Intel® Xeon® processors)

Using Intel IPP

- Intel® IPP can be linked with native code:
  - Only static non-threaded libraries provided for Android* OS
- Add Intel IPP functions into your source code:
  1. Include the IPP header files in source
  2. Call *ippInit()* before using any other IPP functions
  3. Call Intel IPP functions in your C/C++ source.

```
#include <ipp.h>
```

```
#include <ipp.h>
int helloipp()
{
    ippInit();
    ……
    ippsCopy_8u(pSrc, pDst, len);
   ..
 }
```

Using Intel IPP

Include Intel® IPP libraries into the NDK building files:

1. Copy IPP libraries and headers into the project folder (e.g. ./ipp)
2. Find Intel libraries required for the app. Check the " *Intel IPP Library Dependencies*" article.
3. Add the required IPP libraries to the "*jni/Android.mk*" file:
   - Declare each IPP library in prebuilt library module

```
include $(CLEAR_VARS)
LOCAL_MODULE := ipps
LOCAL_SRC_FILES := ../ipp/lib/ia32/ipps.a
include $(PREBUILT_STATIC_LIBRARY)

include $(CLEAR_VARS)
LOCAL_MODULE := ippcore
LOCAL_SRC_FILES := ../ipp/lib/ia32/ippcore.a
include $(PREBUILT_STATIC_LIBRARY)
```

   - Add the header path and IPP libraries to the modules calling IPP

```
include $(CLEAR_VARS)
LOCAL_MODULE     := HelloIPP
LOCAL_SRC_FILES  := HelloIPP.cpp
LOCAL_STATIC_LIBRARIES := ipps ippcore
LOCAL_C_INCLUDES := ../ipp/include
include $(BUILT_SHARED_LIBRARY)
```

Intel Threading Building Blocks (Intel TBB)

Specify tasks instead of manipulating threads

- Intel Threading Building Blocks (Intel TBB) maps your logical task onto thread with full support for nested parallelism

Targets threading for scalable performance

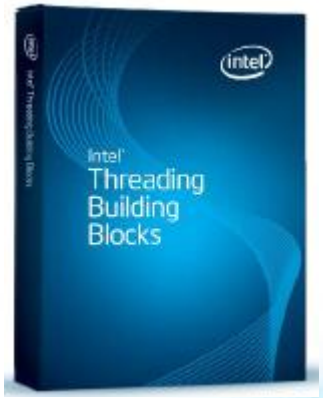
- Uses proven efficient parallel patterns
- Uses work-stealing to support the load balance of unknown execution time for tasks
- Open source and licensed version available on : Linux, Windows , Mac OS X and Android

  Open source version available on :

  http://threadingbuildingblocks.org

  Licensed version available on :

  http://software.intel.com/en-us/intel-tbb



TBB Integration

Android.mk

```
#for including tbb in your project:
include $(CLEAR_VARS)
LOCAL_MODULE := tbb
LOCAL_SRC_FILES := $(TBB_PATH)/lib/android/libtbb.so
LOCAL_EXPORT_C_INCLUDES := $(TBB_PATH)/include
include $(PREBUILT_SHARED_LIBRARY)
#for calling tbb from your lib:
LOCAL_CPP_FEATURES := rtti exceptions
LOCAL_SHARED_LIBRARIES += tbb
LOCAL_CFLAGS += -DTBB_USE_GCC_BUILTINS -std=c++11
```

Application.mk

```
APP_ABI := x86
APP_STL := gnustl_shared
```

Java*

```
System.loadLibrary("gnustl_shared");
System.loadLibrary("tbb");
System.loadLibrary("YourLib");
```

TBB Example

```
#include <tbb/parallel_reduce.h>
#include <tbb/blocked_range.h>

double getPi() {
    const int num_steps = 10000000;
    const double step = 1./num_steps;
    double pi = tbb::parallel_reduce(
        tbb::blocked_range<int>(0, num_steps), //Range
        double(0), //Value
            //function
        [&](const tbb::blocked_range<int>& r, double current_sum ) ->
double {
            for (size_t i=r.begin(); i!=r.end(); ++i) {
                double x = (i+0.5)*step;
                current_sum += 4.0/(1.0 + x*x);
            }
            return current_sum; // updated value of the accumulator
        },
        []( double s1, double s2 ) { //Reduction
            return s1+s2;
        }
    );
    return pi*step;
}
```

World Class Leading Compiler Technology
Intel Cilk Plus

Intel® Cilk™ Plus (Language Extension to C/C++)

Easier Task & Data Parallelism
3 simple Keywords:
cilk_for, cilk_spawn, cilk_sync

Intel Cilk Plus Array Notation
Save time with powerful vectorization

Serial code (left) made parallel with Intel Cilk Plus Keywords. No changes to original code.

```
int fib (int n)
{
 if (n <= 2)
     return n;
 else {
     int x,y;
     x = fib(n-1);
     y = fib(n-2);
     return x+y;
   }
}
```

```
int fib (int n)
{
 if (n <= 2)
     return n;
 else {
      int x,y;
     x = _Cilk_spawn fib(n-1);
     y = fib(n-2);
     _Cilk_sync;
     return x+y;
   }
}
```

Array notation showing simple vector multiplication.

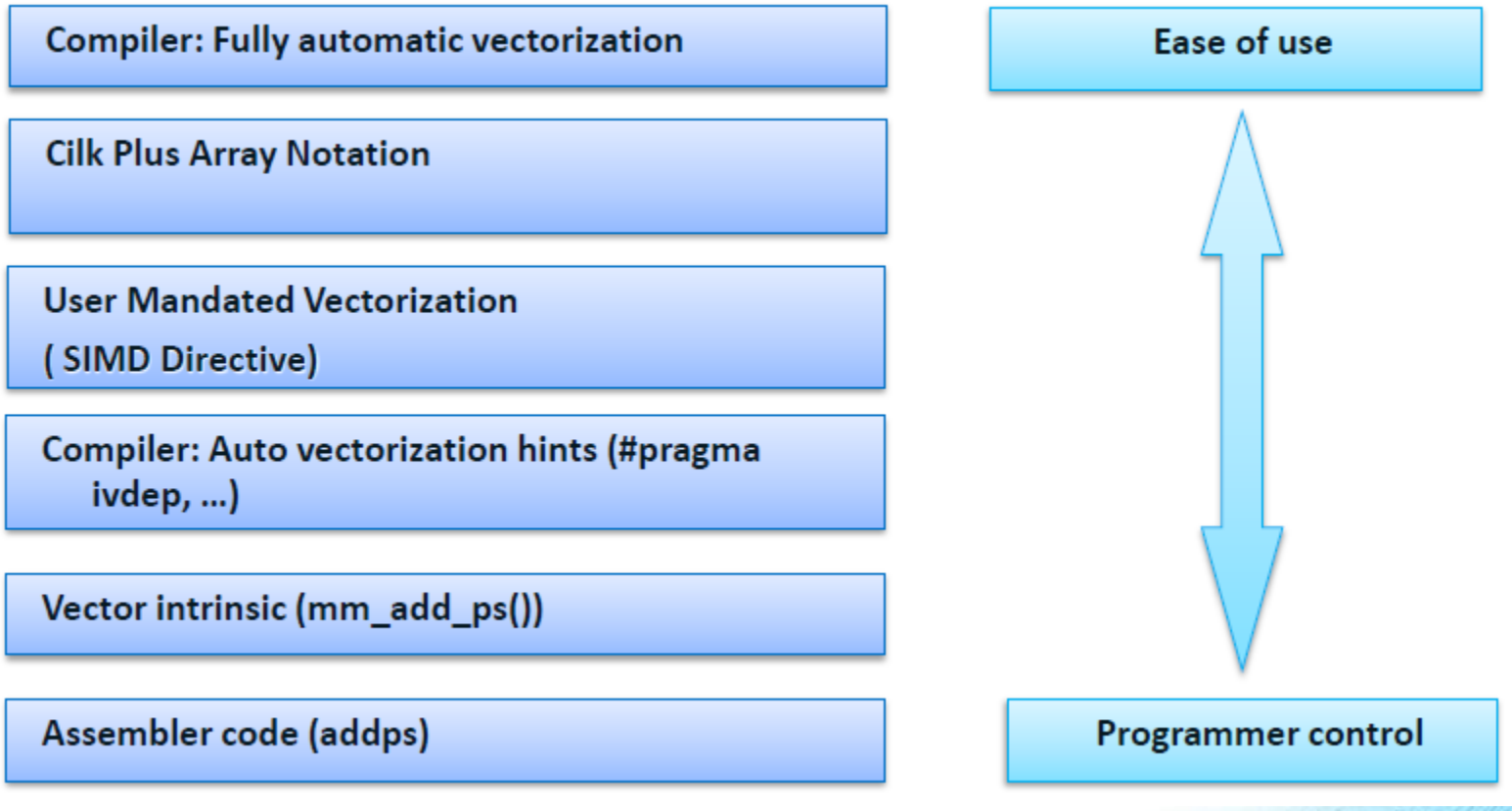
```
a[0:N] = b[0:N] * c[0:N];
```

More sophisticated example of array notation

```
X[0:10:10] = sin(y[20:10:2]);
```

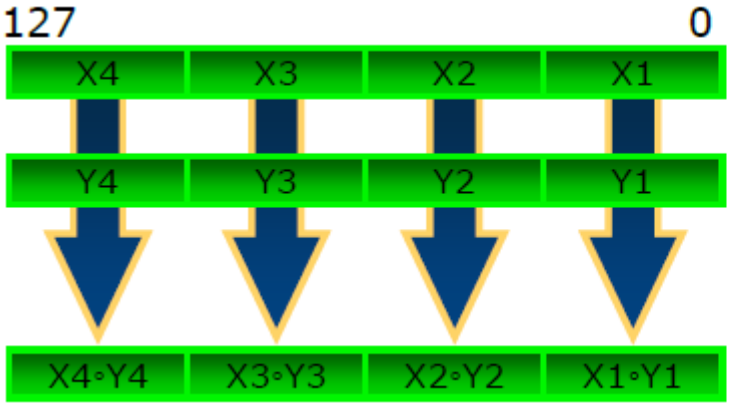
Minimize Software Rework for New Hardware

Approaches to introduce vectorization



Vectorization

SIMD instruction up to SSS3 available on current Intel Atom processor based platforms, Intel SSE4.2 on the Intel Silvermont Microarchitecture



SSSE3, SSE4.2

Vector size: **128 bit**

Data types:

- 8, 16, 32, 64 bit integer
- 32 and 64 bit float

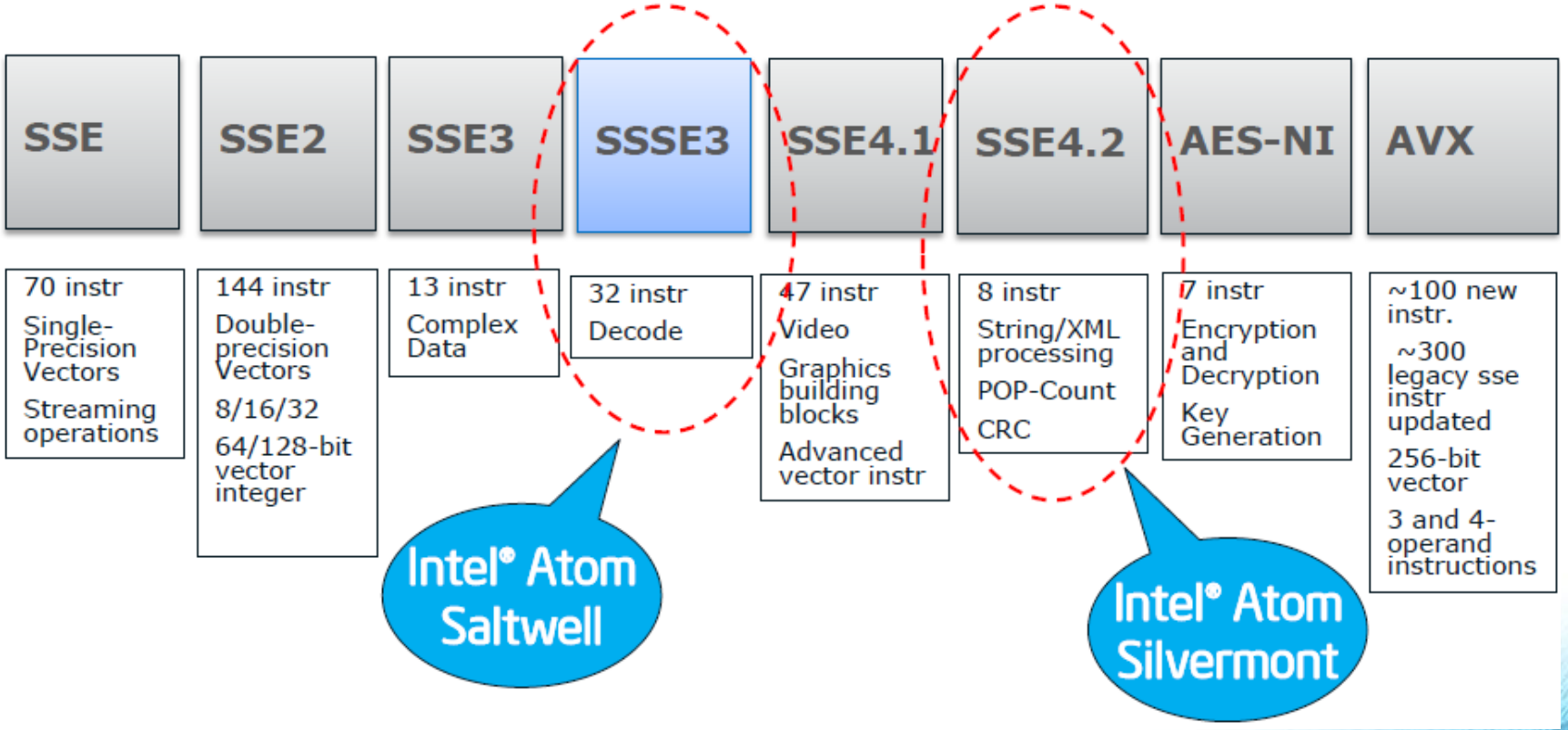
VL: 2, 4, 8, 16

- On ARM, you cat get vectorization through the ARM NEON instructions

Tow classic ways to use these instructions :

- Compiler auto-vectorization
- Compiler intrinsics

SIMD Instruction Enhancements

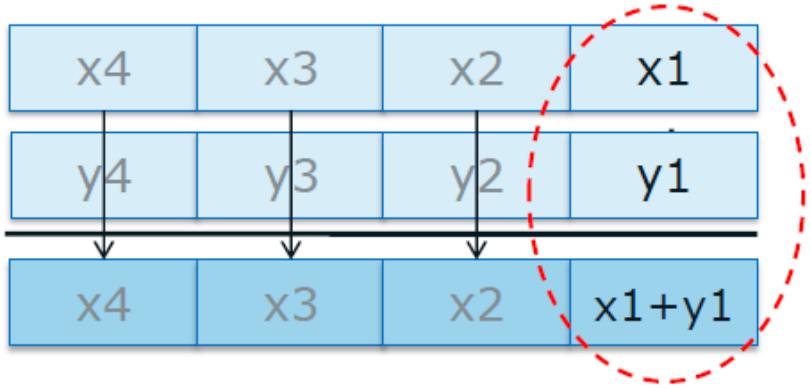


SIMD Instruction Enhancements

addSS Scalar Single-FP Add

Single precision FP data

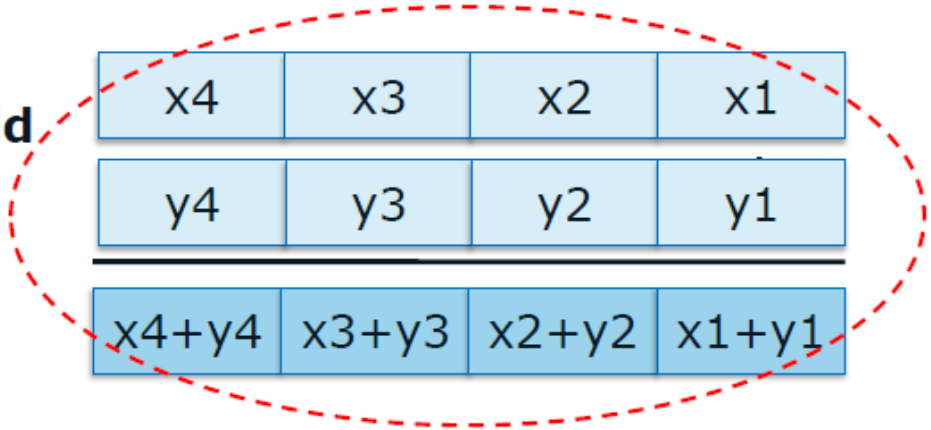
Scalar execution mode



addPS Packed Single-FP Add

Single precision FP data

packed execution mode



SIMD Instructions

- NEON instruction on ARM platforms
- MMX, Intel SSE, SSE2, SSE3, SSSE3 on Intel Atom processor based platforms

- http://intel.ly/10JjuY4 - NEONvsSSE.h : wrap NEON functions and intrinsics to SSE3 – 100% covered

```
//*******  definition sample *****************
int8x8_t     vadd_s8(int8x8_t a, int8x8_t b); // VADD.I8 d0,d0,d0
#ifdef USE_MMX
                #define vadd_s8 _mm_add_pi8    //MMX
#else
                #define vadd_s8 _mm_add_epi8
#endif
//…
```

SIMD Performance

- Latency
  - is the number of cycles after which the data is available for another operation
- Throughput
  - is (the inverse) of number of cycles after issue that another instruction can begin execution.
- Varies for:
  - different instructions
  - 64 and 128 bit version of the same instruction
  - Varied CPUs!

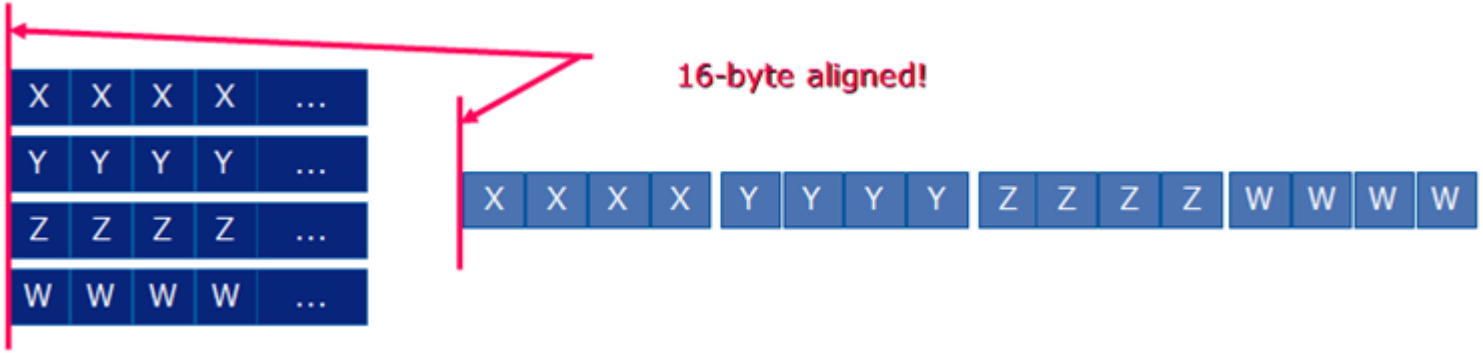
Sample function	64-bit SIMD latencies	128-bit SIMD latencies
Absolute value and Integer "Sign"	1 clock	1 clock
Byte multiply and add	3 clocks	3 clocks
Word multiply high with round and shift	3 clocks	3 clocks
Byte Permute	1 clock	3 clocks
Byte Concatenate with Shift	1 clock	2 clocks
Integer Horizontal Adds and Subtracts	4 clocks	5 clocks

http://www.agner.org/optimize/instruction_tables.pdf

Microarchitectural quirks

Misaligned loads

- Memory alignment is a very important considering in SSE routine.
- In most cases with SSE code, we'll be shootting to ensure 16-byte alignment on a data structure




```
__declspec(align(16)) float a[N];          // static or auto
_MM_ALIGN16 float a[N];                    // compiler macro for static alignment
int* b = _mm_malloc(N * sizeof(int), 16);  // dynamic allocation
F32vec4 c, d[N / 4];                       // Vector Classes are always aligned
```

Microarchitectural quirks

Misaligned loads

- Original code

```
for(i=0; i<16; i++)
{
        a += (float)(*input)*(*window1);
        input++;
        window1++;
}
```

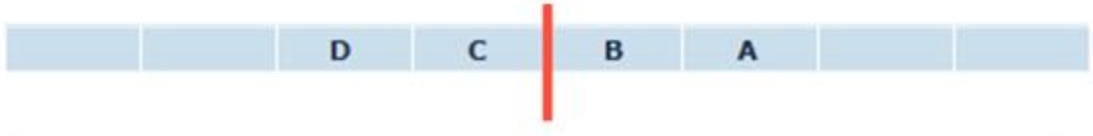
- Aligned loads

```
switch (offset & 3)
{
        case 0: // data at address of 16 Byte align
        {
                __m128 window1_128_0 =  _mm_load_ps(window1);
                __m128 input128_0 =  _mm_load_ps(input);
                a128 =   _mm_mul_ps(window1_128_0, input128_0);
```

Microarchitectural quirks

Misaligned loads

- PALIGNR to avoid cache-line splitting loads



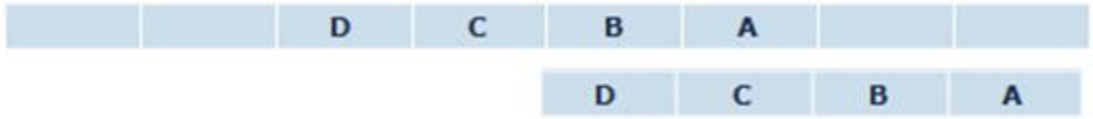
- Load move index -/+ 2

```
__m128i window1_128_a0 = _mm_load_si128((__m128i*)(window1-2));
__m128i window1_128_a1 = _mm_load_si128((__m128i*)(window1+2));
```

window1_128_a0	B	A	0	0
window1_128_a1	0	0	D	C

- Concatenate and shift the result by 8 byte

```
window1_128_0 = _mm_castsi128_ps(_mm_alignr_epi8(window1_128_a1,
    window1_128_a0, 8));
```



Microarchitectural quirks

Blocked Store-forward

- When a store to memory cannot be forwarded to a subsequent load that access some part of the recently stored data.
- When this occurs, the load stalls for several cycles.
- Atom processor supports a very limited number of cases when the data can be forwarded efficiently without stall.
  - The load and store are to the same address
  - For the same size operand
  - From general-purpose register(e.g., eax, ebx, …)
  - Stores of SSE operand(from xmm registers) are never forwarded to subsequent loads.

Microarchitectural quirks

Blocked Store-forward

```
// windowing code
__m128 sum4;
float pSum[4];

sum4 = _mm_mul_ps(_mm_load_ps(&window[0], &b0[0]);
sum4 = _mm_add_ps(sum4,
        _mm_mul_ps(_mm_load_ps(&window[4], &b0[4]));
...
_mm_store_ps (pSum, sum4);  // sum3 sum2 sum1 sum0
int pcm_sample =
        (int)(pSum[0]+pSum[1]+pSum[2]+pSum[3]);
pcm_out[i] = CLIP(pcm_sample);
```

- Stall between the 16-bytre store to the pSum array and the following 4-byte loads from pSum.

Microarchitectural quirks

Horizontal sum in the xmm registers

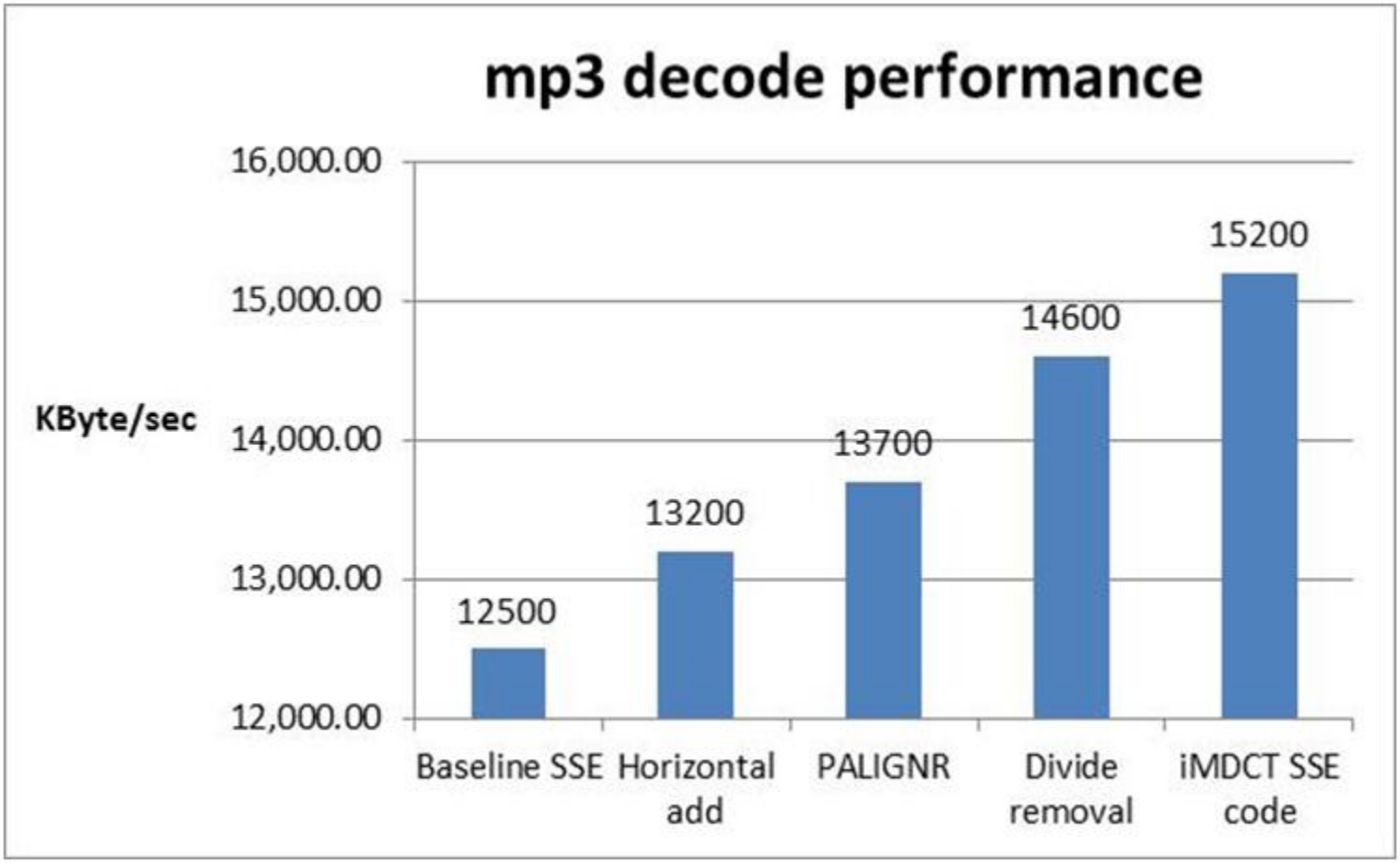
```
// horizontal add of elements from sum4
   // sum4 =  sum4[0]+sum4[1]+sum4[2]+sum4[3];
   sum4 = _mm_hadd_ps( sum4, sum4);
   sum4 = _mm_hadd_ps( sum4, sum4);
   //  Clipping to 16-bit integer range
   sum4 = _mm_max_ss(sum4, lowerBound4);
   sum4 = _mm_min_ss(sum4, upperBound4);

 //  Convert to integer and store to pcm buffer
   pcm_out[i] = _mm_cvtt_ss2si(sum4);
```

sum3+sum2	sum2+sum1	sum3+sum2	sum2+sum1
sum2+sum1+ sum3+sum2	sum2+sum1+ sum3+sum2	sum2+sum1+ sum3+sum2	sum2+sum1+ sum3+sum2

- The horizontal sum can be computed with HADDPS instruction

Microarchitectural quirks



Microarchitectural quirks

Streaming stores

- Intel Atom processor have no L3 cache.
- The L2 cache is typically only 512 KB, so consider whether your data set will fit.
- If not, then your routine may benefit from using streaming stores(aka. Non-temporal store instruction, MOVNTPS or MOVNTQ) that write directly to main memory without polluting the caches.

```
for(j=0; j<w0_8; j+=8)
{
        ……..
        _mm_stream_si128((__m128i*)&rgba[j], rgba8888_0);
        _mm_stream_si128((__m128i*)&rgba[j+4], rgba8888_1);
}
```

Microarchitectural quirks

Streaming stores

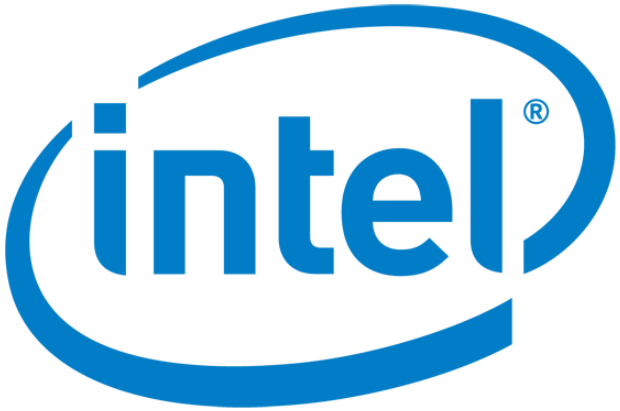
Color Space Conversion	Time(ms)
Baseline VS 2010 compiler	25,430
Intel Compiler /QxSSSE3_ATOM switch	9,350
SSE2 intrinsics	9,520
16-byte alignment	9,000
Non-temporal stores	6,210

Resource

- Intel Software Android Community http://software.intel.com/android
- Beacon Mountain v0.6.1 for Android* http://software.intel.com/en-us/vcsource/tools/beaconmountain
- Intel for Android* Developers Learning Series Landing Page http://software.intel.com/en-us/blogs/2012/11/28/intel-for-android-developers-learning-series
- Android* Application Development and Optimization on the Intel® Atom™ Platform http://software.intel.com/en-us/articles/android-application-development-and-optimization-on-the-intel-atom-platform
- Android* Tutorial: Writing a Multithreaded Application using Intel® Threading Building Blocks http://software.intel.com/en-us/articles/android-tutorial-writing-a-multithreaded-application-using-intel-threading-building-blocks
- Finer Points of using SSE Instructions for Android* media apps on the Intel® Atom™ Platform http://software.intel.com/en-us/articles/finer-points-of-using-sse-instructions-for-android-media-apps-on-the-intel-atom-platform
- Hands-on Lab: Develop, Optimize, Debug, and Tune Applications for Android* https://intel.activeevents.com/sf13/connect/sessionDetail.ww?SESSION_ID=1115

Resource

- Using the Second-Screen API and Intel® Wireless Display from Android* Applications https://intel.activeevents.com/sf13/connect/sessionDetail.ww?SESSION_ID=1134
- Developing Native Applications on Android* and Optimizing for Intel® Architecture https://intel.activeevents.com/sf13/connect/sessionDetail.ww?SESSION_ID=1129
- Accelerating Your Software Development for Android* on Intel® Platforms https://intel.activeevents.com/sf13/connect/sessionDetail.ww?SESSION_ID=1228
- intel compiler download site http://software.intel.com/en-us/c-compiler-android/
- IPP download site http://software.intel.com/en-us/intel-ipp


intel