

Re: 10 bit per channel YUV with alpha

Source:

<http://www.tech-archive.net/Archive/Development/microsoft.public.win32.programmer.directx.video/2007-06/msg00>

- *From:* "tOrakka" <tOr@xxxxxxxx>
 - *Date:* Mon, 4 Jun 2007 09:33:05 +0300
-

"Lee_Nover" <Lee_Nover@xxxxxxxxxxxxxxxx> wrote in message news:op.tqqlsrlx9gasuk@xxxxxxxxxxxxxxxx

method I apply a lot :)
even faster is doing two/four additions (in ASM)
that way the operations can be efficiently pipelined into U and V pipelines, nearly doubling
the performance ;)

What U and V pipes? Aren't those, like, Pentium specific details which aren't really relevant anymore..? Intel has like 3 major architecture designs since then for implementing the IA32.. (PentiumPRO/PentiumII/PentiumIII (that's one), Pentium 4 aka. Netburst (that's second) and Core/Core 2 (that's third..?) which all (where did the PentiumM go..? :) work radically different and don't benefit at all from pairing instructions to U and V execution ALU's which DON'T EXIST anymore in these architecture upgrades. :) :)

Then AMD has their own architecture, which again doesn't benefit much if at all from pairing instructions in specified manner.

Well, good job anyway.

If you really care about performance, the key topics are:

- hide latency
- mind the memory access pattern

How you "hide" the latency? CPU executes basically infinite sequence of instructions, for practical purposes the sequence is of course finite. :) Even though the instructions are decoded and processed into micro-ops, or macro-ops or what not depending on the architecture and can also be executed out-of-order, there are few little bumbs in the road: dependencies being one of them.

Some instruction or operation cannot be finished if one or more of it's inputs isn't computed yet. This creates dependency to the previous computation. If you write your code without understanding these facts, you can easily create situation where single statement in the code can be a bottleneck which stalls everything else.

Also, memory read requests can create arbitrary bottlenecks and dependencies.. if you can issue memory read request and use the result later, it's always a benefit. C/C++ compilers can in theory do this kind of "optimization" but it's best left to the programmer as the compilers are really crappy re-ordering statements, they usually follow the sourcecode more literally than is optimal (for our mutual good, ofcourse, since

Re: 10 bit per channel YUV with alpha

changing evaluation order without changing the semantics can be problematic because C/C++ is *a*/are language(s) which haven't really solved ALIASING very well (it doesn't really even try to address the issue..) I mean, sure, we have a nice clean SSA or AST or similiar intermediate representation of any basic block of the code, but we cannot really re-order beyond specific limits, for example inputs to a function are often pointers and arrays and there you go pretty much. If there was "write only" modifier, it would help a little bit since we already have "read only" mode for arrays etc (const, ring a bill?)

Yada yada yada.. it's all common sense, I don't see the end to this rant to I just put the stop here.

.