XilinxApr1505

[music]

Operator

Hello. I would like to welcome everyone from around the world. Thank you for joining us today for this webcast on "Design Techniques for Best Performance, Power and Cost," brought to you by Xilinx and Tech Online Webcasts. The presenter today is Niall Battson, Technical Marketing Engineer in the newly formed DSP Division at Xilinx. This webcast software allows you to sit back and have the navigation advance automatically. As a user participating in this webcast, you will be able to ask questions at any time during this presentation by clicking on the "Ask a Question" button, and then typing your question in the window that pops up and clicking, "Submit." Niall will be answering these questions at the end of the webcast, please feel free to enter them at any time. Also included in this webcast is a survey; please take the time to open, fill out and submit the presentation survey. You can access the survey at any time in the "Print Documents and Viewings" pulldown menu on the left-hand side of your interface. This survey will also pop open when you choose to close your viewer window, or when the view window closes automatically at the end of this webcast. By

submitting this survey, you will be providing Xilinx and Tech Online with valuable feedback on the subjects covered in this webcast, and also how we can improve the webcast product. And now it give me great pleasure to introduce you to Niall Batson.

Niall Batson

SLIDE 1:

Thank you, ***. Please welcome everybody as *** for this big session now in the Tech Online series of the Virtex-4 solutions we have on our power, performance, signal integrity, SSIO, *** interesting from Doctor Howard Johnson. And today I am going to talk a lot about DSP specifically and what has happened in Virtex-4, and with regard to XtremeDSP Slice, to see how it is going to affect our designs for system engineers or hardware engineers, in regards to performance, power and cost; and how it can really help increase performance by terms of *** in making some and I feel co-processors as a conventional DSP processes. So, let's slide right in here.

SLIDE 2:

The Vertix-4 DSP advantages. Vertix-4 was announced about nine months ago back in September *** fast. I have been working in Vertix-4 for about two years now, looking at designs, looking at how it is going to impact you as hardware designers in terms of

the techniques that you need to use to get this higher performance. You'll have twice the performance of a Vertix-2 Pro in Vertix-4. That's really, really great. I mean, that's fantastic from a hardware engineering perspective because as I increase performance in my DSP functionality, you start to shrink the size. That's going to have a big impact on the cost. Also the power has greatly dropped I've seen by paying a lot closer attention to power. It's *** particularly low power, but we've seen this great improvement, especially in Virtex-4 over Virtex-2 Pro. I have been playing with lots of designs as well. I've seen a massive reduction in the slice count of my designs over Virtex-2 Pro as well, especially in all different types of filters and FFT arrangements. I've seen by exploiting these advantages currently of Virtex-4, I'm getting at least a 50% reduction. As I chose some key examples throughout this session, I try to emphasize that as well. And also, we're a programmable logic company, so we haven't sacrificed any flexibility. So, Virtix-4 with its introduction of the XtremeDSP Slice has yet to do that either. So, there's a lot of flexibility. But I'll point that out in more detail as I go through.

SLIDE 3:

As I said, what we are trying to show you today is a look at

Virtex-4 and see the impact of what this XtremeDSP Slice is going to have on you as hardware engineers; or even in our system engineers listening to this session, what you should tell the hardware engineers the kind of performance, power and size reduction benefits they should be trying to get by exploiting this element inside Virtex-4 correctly. So, as I said, we'll look closely at how we enable that 50% benefit, especially in regard to our competitors, and also increase a much better power reduction. I also want to have at a couple of examples, a digital communications example and a video processing case study at the end, that will take messages up there I am trying -- to say the advantages, to show that a high level, larger functionality will get all the benefits still. The user at the low level of Virtex-4, if he flies correctly, will see much greater benefits at these large system level functions as well.

SLIDE 4:

So, first of all, I'll start off with the Virtex-4 family, and I'll look at the XtremeDSP Slice in detail: I'll give you the history of it, see how its changed and how it is going to impact us, and especially how it's going to change in respect to filter techniques. Everyone always looks for the filter techniques to start with because they're just so rife in DSP systems. So, if you look at [first opus (?)], you can see how that impact other

functionality as well. And that's how we will go into the digital up converter example and [2-D surface].

SLIDE 5:

So, starting off with the Virtex-4 family. As I said, about nine months ago, Giant introduced the Virtix-4 family.

SLIDE 6:

And it did something very different. For the first time ever, and FPG vendor decided to have three sub-families, Splitting out that family into smaller sections, the LX, the FX and the SX family. They all give a slight bias to different kinds of applications: LX is more logic-centric; FX is more systems-centric involving processes and MGTs; and on our DSP design, SX is the one that's most interesting to us. Let's have a look at that SX part.

SLIDE 7:

As you see, Virtex-4 introduces this AFMBL architecture, which is *** of all: the fabric, the multipliers, the memory, the IO, all being in a column-based structure. As XtremeDSP Slices, the horsepower, the work engines if you will, or DSP, edit a number of columns of them in the SX family. The smallest SX part gives me up to 128 XtremeDSP Slices. The largest, 512 XtremeDSP

Slices. And you'll say, "Where do you get 500 megahertz to process speed grade; that's a lot of DSP computational power." For advanced algorithm designers out there, this is going to be very exciting for you. I'm also looking at the fabric count; it's still got 2,650 CLB *** parts. It's all got all the functionality to complement the DSP processes, and also any communication you might need, like the PCI for example.

SLIDE 8:

So, let's kind of have a closer look at that XtremeDSP Slice, and see the impact it's had and how it's slightly different.

SLIDE 9:

Let me bring up the next slide here. I have a few problems bringing up the next slide. It's a wait.

-: Yes, I just moved it for you. Go ahead.

I'm not seeing anything.

-: Yes, that's on my slide now, so you should go ***.

OK. Do you see the extreme DSP? It should be multiply in the register.

-: Now I'm on Slide 9.

***, it's identical.

-: You know what? I've got the audience to Slide 9 --

OK. If you were to Slide 9, you should see the multiply

register. That's the multi-18-by-18 you see in Vertix-2 Pro. And you can use Vertix-2 Pro. That's exactly what you'll see, what you have been using in the past. So that was a good function. We found that we had to sort of bend that functionality. So the most common thing that peoples often do was always have input registers in front on the front of the V2 Pro mult-18-by-18. So, they automatically -- it's like *** -- You'll see they added the input registers.

SLIDE 10:

That was something that people used to use fabric, and it poured that into the Vertix-4 cycle. It was the first step we took. That's fantastic. That's really good because that will fortunately start to reduce the fabric count. You should be able to start to see how the fabric count is starting to decrease.

SLIDE 11:

On our next FOIL, you'll see there will be added an added two FSP. Often in Vertix-2 Pro designs, we started to see that the performance bottleneck was always in the additions, especially in those large multiply accumulate engines where you have like 40-bit add results. This is very important, to start to move it into an embedded functionality, to get the performance gains we

wanted to get. And the system designers and big hardware designers were *** the designs. So we added that, obviously a [typeline] register at the end. And using [chess gates], you can see on the input there. That is going to be very important for the ***, and you'll see how I exploit that as I introduce examples.

SLIDE 12:

We've now added Slide 12. On the next slide, we can see an added functionality. Remember, Xilinx is programmable logic company. We don't make fixed functionality. We need to give you options as designers. So we added these *** not only to multiply ***; maybe you multiply and accumulate; or maybe a multiply and add to a cascade. Cascading is very important in lots of filter functionality. And I'll show how using that dedicated captive, you'll see the PCU, the PCF is extremely important. So we've added that. This introduces a new sort of idea to you as designers. It's an idea of an Opmode. I have to make a decision now between multiply add, multiply accumulate to multiply to scale. And that's decided by what architecture you use, include an Opmode: the idea of changing Opmode to change the functionality of an XtremeDSP Slice.

SLIDE 13:

On our next slide, you'll see the XtremeDSP Slice as it is today. That allotted more Opmode to the added agency. There's the x-max, the y-max, and the zed-max. That's going to be extremely important, using that, to get in the kind of functionality that you might need. There is many different ways you can use this block. I really want to stress to you, it's not just a multiply accumulate engine. It's actually, it can be an accumulated by itself. I could set the Opmode to be a multiplier or multiply it to scale. Or you can constantly change that on the fly.

SLIDE 14:

On the next slide -- I really enjoy this one -- you can probably can't even read it in the audience, but it is trying to stress the point: They have gone over 40 different modes in the XtremeDSP Slice. That is a lot of different modes. So please, don't think of this as just an multiply accumulate engine; it can do far, far more. And you should be looking at it in detail to study how you should exploit your algorithms better in the XtremeDSP Slice. The more it keeps function of the indirect XtremeDSP Slice, the easier one jump will be as a designer to get a lower ***, better power performance and also top performance. This whole block in a *** speed grade under 500

megahertz. That's brilliant. If you keep inside the block, you store the tight line in that will be provided, you are guaranteed to get up to 500-megahertz number. And I'll show how I try my best every time I do a particular application to get as much of the design as possible in to the XtremeDSP Slice, and exploit the cascades. Again, over 40 modes, and it is dynamically operated. A lot of people get a little bit confused with that dynamic operated: "How can I take advantage of it?"

SLIDE 15:

So, the next FOIL, I call it providing sample, probably most familiar to most of you, as we're doing examples in DSP the complex multiplier, the heart and soul of the FSP ***, and adapt to a complex filter for example. Here I'm not using all four multiplications that you might do, and it's *** the complex ***, or maybe three with some addition. But realize as a designer, I thought, "Hang on a second. My day to eight media 100 megasamples per second or slower. This cannot be the slowest speed rate. It needs to be slight, and live up to 400 megahertz. That gives me four clock cycles to play with." Use the idea, it's time to fold down functions into one multiplier and reuse it many times. So, it's very small slice count.

SLIDE 16:

On the next slide I introduce the first clock cycle. And this is specific Opmode required to have the B x D in the multiplication, and also I'm going to be there's no -- I'm just making sure it is attached, because the B x D must be negative, and adding a zero to it. The *** subtraction on the output of the multiplier.

SLIDE 17:

The next cycle is what I do next, at the end of the complex multiplication. Output to the multipliers then. I can now change the functionality of it to be an accumulate, and add to the previous result that I calculated. That gives me my real results by making sure the control logic *** and also sets the correct captures, and enables for me to capture the real result.

SLIDE 18:

Next I'm going to work on the imagining as a change in Opmode again on the third clock cycle, Slide 18. That's great because there's a change in Opmode dynamically, you think, on every clock cycle of that Opmode.

SLIDE 19:

And again in the next FOIL, I *** the fourth clock cycle, we're

back to a multiply accumulate engine again. And this is functionality, exploiting clock cycles to reduce the size of our resources for a particular function. Reducing the function size drops the power and also drops the resources. That's really, really good, and ultimately cost. Using really good techniques that I often don't see hardware designers exploiting. So again, the size is very small. But all that control logic is being used to directly change the Opmode of a DC48 slice. So, how do I generate that control of it? That's a question I also get from a lot of people. And it's really very, very simple. I've seen some customers who specifically do a lot of state machines, and add its statements, and all kinds of more complex state machines to try and do this. I find it much more simpler if we just do a small medley, taking this to require seven bits, and then a counter on the input. That's a really good way of doing it and it only uses five logic slices. But I can guarantee you'll get more performance if I do that. That's a really, really good technique to use, to do these kinds of dynamic Opmode changes. Just a little memory, this distributed memory that we have in logic fabric, and then a counter, each account: one, two, three, four. And then you can just repeat that by using an MA equals to switch it on and off. And here, I'm using four cycles to reduce the size of the function. That's very important; again, reduction in size. And also, it is great to

use the dynamic Opmode to your advantage.

SLIDE 20:

In the next slide, we'll see the XtremeDSP Slice Cascade in full force. Remember as I said earlier, the Virtex family has the XtremeDSP Slices at four *** -- It's four families actually, having them in columns. So those XtremeDSP Slices, they stay on top of each other, and the cascades work their way up the column. So you must exploit the columns to get a lot of the benefits. Think about that carefully. It's very important. I really try to stress that as I look at some of the filter examples I did. There are a few key points to mention here. Few XtremeDSP make one XtremeDSP Tile. Now the tile shares a C input. They show on the FOIL there. It's the only that is registered there. So you can only use one C input for two XtremeDSP Slices. So bear that in mind. Don't try and use the inputs for all the XtremeDSP Slices. And this is where cascading will come in very handy. So you can use one c input for a particular input, and then you start to use the cascades so we free up C inputs. Bear that in mind.

SLIDE 21:

On our next FOIL, we have got to talk a little bit about power as well. Power in Virtex-4 is getting a lot lot better. Do you

see anything about multiplying indirect engines; the yield out there is very *** multiplying indirect engines as consuming as little power as possible; especially if I'm starting to push the clock frequencies up. Power is always related to frequency. If I push up the clock frequency, up flies my power. In Virtex-4, remember, I'm stressing I can get twice the performance I can get in Virtex-2 Pro. So I hope I have improved the power, well, we have in Virtex-4 fortunately. We've gone down to about the 12 for a given multiply-accumulate function. It's about 2.3 milliwatts per 100 megahertz for leading ***. There's a lot of data on the internet at the moment regarding our power numbers. But that's just one little function, isn't it?

SLIDE 22:

On our next slide, you'll start to see how I have started to exploit -- I will do Estimate Stop. Estimate is actual power *** on actual much larger functions. The first slide you can see a 63 Tap FIR Filter. This is a good example. It uses the cascade as I said to implement a fully parallel FIR filter in the *** chain. We also went ahead and did it in the science parts so we could compare the power difference between our smart competitors. We see a significant reduction in power because we are using the cascade columns of these DSP48s. In fact, we use particular filter architectures to really take advantage of

that, so we don't go out into the fabric to consume too much power to the same functionality. We're seeing a 57% reduction in power, much greater than one watt a difference. System engineers are always keen to reduce the power consumption of these parts. And so that has been an extremely important to note. Always push your hardware engineers to get used to columns and take advantage of them without using up extra fabric, so you can get the power down.

SLIDE 23:

In the next FOIL, we look at a slightly different function, a 1024 Point FFT. For this example, I used 16-bit data, very commonly used, and 16-bit twiddle functions for the coefficients is what I began as FFT. Again, it's streaming data, high performance. I'm still seeing a big reduction in power in designing ***, again in that *** territory. The difference in dynamic power won't seem so traumatic, only 7%, mainly because we're dominated in this particular application by the logic fabric. Look at the logic fabric resources in both *** and acknowledge or see that we see that it must be the same. We've got a more efficient solution and it's certainly going to effect the power. But most of the power consumption is coming from the fabric. So that tells me that how to design it. If I can start to use more DSP Slices and less fabric in my algorithm in

general, ***, I can start to really get on top of the direct power and start to bring it down, taking advantage of it down. That is going to be very important when you start doing about this. Look to exploit the DSP48 slice as much as possible. That makes the SX family very, very appealing because its ratio of multiply *** DSP slices the fabric is the most biased towards the DSP48 slice; so, you've got the best shot to getting that power down. The *** when you're looking at families as well.

SLIDE 24:

Now, let's have a look at some filter techniques.

SLIDE 25:

Next slide. This is getting very important. I think it is what is most important to you. Look at how this low level, how this DSP Slice has changed. Now let's look at how it has impacted algorithms and what difference it is making.

SLIDE 26:

It's not filter algorithm on the next slide. This is the family, what I'm sure you've all seen at school many times. And you might have seen a picture like this in your textbooks. You might even carry it on the desk next to you. *** typical sum of products with a *** summation tree at the end. Now, the

question we need to ask though is, "How do we implement this function?" This is just a sort of a signal flow graph we are looking at here. How does verdict score change my implementations, and how would I have to consider building this function based on [e-gravitors] for FIR filters?

SLIDE 27:

In the next FOIL you see, dual-axis sampling a number of coefficients. It is very important to understand because these two key factors in how I am going to start building my first filters. I've built this FOIL an envelope. Let's just say for example I try my hand, I can only use one multiplier to do one MAC FIR filter. That's all I'm allowed for example. That's the envelope I'm looking at, OK? This is a particular *** of filters. I call them sequential FIR filters. They tend to do things of a number of cycles. You can see there is a performance envelope. There is a point where I can support only so many cycles or so many coefficients for a given sample rate. What is interesting though, is in Virtex-4, this envelope has risen due to the potential increased clock speed of the DSP48. Now, I've only 400 megahertz, I can support more coefficients. So, I would expect in general, the applications that haven't increased in sample rate, they will be looking to do these more sequential type FIR filters. So, bear that in mind when you're

looking at Virtex-4. I expect to see these to be more prevalent. Let's have a look at specific implementation now, and in the Virtex-4 MAC FIR filter. In our next FOIL, I've highlighted a filter structure to show you how it could be built. Immediately, you start to see the DSP Slice thing takes advantage of, and how it's going to help me over Virtex-2 Pro. Remember, the *** had to be done in Virtex-2 Pro. Now, it's all embedded in the XtremeDSP Slice. I specified lot mode on the FOIL. You can see the one I used. By dynamically changing the Opmode as well, just one bit of it, so they multiply accumulate every so many cycles -- every 366 in this example -- will reset to another result that these three calculates. I'm creating over 50 slices versus Vertix-2 Pro, by using the XtremeDSP Slice. That's very important. That's going to help reduce my power, reduce my resources, and also any cost as well. Here's the dual-port block memory in dual=port mode, very handy, and a control logic, sitting from the ***. That's going to be expressing the coefficients, and also using a cyclical round buffer in the top part of the memory. Size: down there, very small, very compact. And I can easily achieve the kind of frequencies that I expect. All my analysis here I've been doing is on the slow speed grade. With the marketing numbers you get exposed to, always pursue the high-speed grades. I find, when I work with customers in general, they usually use the slow speed

grade ***. *** probably more realistic geo-environment. *** input my *** I have done, and I tried to design the same filter in Stratix-II.

SLIDE 28:

So, for the next slide you see the Stratix-II implementation. Now, immediately we can start to make some opinion statements on the DSP plot in *** that starts *** slice designs. Now, what's different in that mode? And that means unfortunately, I lose two of the multipliers in *** mode. That is got to be worrying for hardware system designers. Half my potential multipliers if I were to use the whole path ***. That might not be possible. But for a given max unit, it costs me two of the multipliers. That's got to be worried. Sort of hurt. You lose a potential piece, *** engines. And that also forces me to use a second multiplier as a MAC engine as well. It's not possible to do them differently. *** I have to use a few of them here to store my data and coefficients. And I have a register on the output. Again, performance starts to drop rapidly. I'm using an *** FIR Compiler here, the latest version off the web, to get these data points as well. Again, Virtex is much, much smarter here, by pushing the envelope; because we're using ideas we've got and it gives a nice coagularity of the single element, which is very good for c sequential FIR filters. I believe the critical part

was actually in the memory section -- hang on -- in the Stratix-II.

SLIDE 29:

Next foil. Let's have a look at the super-fast situation, that's *** collection. Extremely high data rates coming into you system and there need to be extremely high performance filters. I call these parallel FIR filters, OK? They have their own sort of performance finality lines as well. Over about 225 or 200 or 250, depending on where you are trying to shoot for, for your actual frequency in your parts, you see that the line comes into play. Now I have less than two cycles available to me. If I have got two cycles, I can start to fold the filter and use less multipliers to do more coefficients. Here, it's one coefficient per multiplier you're using, OK? This is a high performance environment. Let's have a look at it has been impacted in Virtex-4, using XtremeDSP Slice.

SLIDE 30:

Here we're looking at -- on our next slide -- this is Systolic FIR filter architecture. We call it a sys-con, but really I guess its most technical name for the mathematicians in the audience would be direct form tray one with one extra stage of pipelining. That's what *** thinks, and this is a good term

that we have been using here *** to describe this structure. It's very, very novice for the XtremeDSP Slice and Vertix-IV because it uses the columns, you see. Use the cascade between XtremeDSP Slice. The non-*** performance, and it's using added chains. That is going to be critical to analyzing Virtex-4, an in your algorithm implementations that you do. Added chains must start to become part of your design algorithms. If you to exploit the exchanges being sliced properly, we can get the benefit that we're talking about: the twice the performance, the lower power. Here in this example, I used 23 *** metric. I used 23 exchanges XDSP slices. That's extremely good. No fabric is being used. Hence, the power story that we have got earlier, just to take an example. That's really good. Added chains are what is enabling us here. We use algorithms to exploit added chain. Again, also look at the two registers to each XtremeDSP Slice. That is very useful because it helps me match up the point values again to the added chain. They were put inside the slice on purpose for this kind of structure.

SLIDE 31:

Have a look at the Stratix-II implementation. Now, we'll deal with a slightly different approach for their DSP solution. They have given through this adder tree structure. The *** things change are the key to getting the performance advantages that

you would want for tomorrow and today's DSP applications. They have gone with a tree. And there really are significant problems in doing the tree. It may be that it's *** simple, but it starts to hurt us in our performance. As you can see, the DSP block is very nicely used up there for multipliers and the added tree there with three others. That's great. It gives me a single result. I still have to combine the results in each DSP block. Now, this has been a very nice thing. It starts at two. I've *** one. They have introduced the tree for adder. And that makes this tree smaller. That's good, there's definitely a nice connection. But still, you can't avoid using fabric adders for this particular example. In fact, any of these parallel structures ***, apart from the four-path one, which might never *** parallel, you have to go into the fabric and start combining results there. Now fabric has its problems because it seems to be slow for large adder change. And unfortunately, the adders are the output of these *** are always the largest, especially the large one; it must be up in the 40 bit territory for these examples. And this is what leads to performance drop-off. Now, you can't pipeline those added chains. And it starts to push the performance up. But that's not the thing. It impacts our resource usage. It impacts our power. And that's not good either. Do you see any way out of this? Using adder trees seems to be very detrimental. At a

chain approach, you start to get at the advantages that we would like. This leaves a situation: what do we do in the in-between? Most of you, I imagine, will be looking at the semi-parallels. For the next FOIL, I start to fill in the gaps between the parallel and the sequential.

SLIDE 32:

These are the grey areas, when I have four multipliers or eight multipliers. It all depends on the kind of speed that your input data rate is coming at. But we are going to push the clock speed up the performance that we have given you. So I -- You know, we are giving you a Ferrari here. Let's use it. And drive it fast. So, let's have a look at the implementation of the Virtex-4 multipliers. So, the four multiplier is *** semi-parallel filter. I'm using HDTV sample rates here. What you have got is free high-performance application. It still affords me a number of clock cycles that is my *** in terms of my 400 megahertz, so it is low speed rates.

SLIDE 33:

Now, here I'd have taken 16 pacts and I folded it by four. So I have taken 16 coefficients, by mapping them down to a 4 multiplier, and the filter gets muted. NED, accumulate your need; that is going to be very important for accumulating the

four results over the four cycles. Look how I have exploited adder chains again here. It's very important. A very good example of the adder chains. Even using the cascade to go with the accumulator at the end. The input data, again we are taking advantage of some of the great features of Virtex-4. The SOL16s, I use them for the memory buffers and the data. They are very useful, the SOL16s. My good friend Jim Chapman in the UK always talks about the advantages of SOL16. You can find a lot of work on our design support site under the Tech Exclusives. Take a look. He's got lots of SOL16s and their advantages. And they're very good here. They take very well in this structure. In fact, a lot of these semi-parallel structures take advantage of a small distributive SOL16 memories; because you have got a number of multipliers, and you are dividing your coefficients by the number of multipliers. So your sort of taps have multipliers. And then they start to get smaller. The faster your project goes, the smaller the memories I need are. We saw un the sequential world that we have used larger memories because they are running out of more cycles. As you push the performance up, their memories get smaller and smaller. We need more memory bandwidth. So *** is very good store a lot of memory bandwidth because you have these tiny little memories that can be quite big or small, but they play, the example here, 4 by 18, 18 bits, four ***. ***. Very small

little memory, we use the slices for that. It looks like we have got good performance again by staying inside the XtremeDSP Slice column as much as possible, using a little bit of fabric to help store my memory coefficients and my data, a little bit of control logic as well.

SLIDE 34:

So, just to prove my point, I went ahead and built it in our software. In the next FOIL you can see an implementation system generator. That's a high level ***. You can find a lot of information out on the web, even *** demand by doctor *** generate a group. He talks about that on our website as well. *** system generator. I just wanted to point out Virtex-4 and ***, and you can build these kinds of functions. This *** is available on the user guide section, the *** on the web for Virtex-4. And you can see how using *** architectures here, the two *** blocks together. You can see its direct implementation looking very similar to my foil on the screen. That's really good. And what's even better is it can grow and shrink on a number of multipliers very easily. Because if you look at the structures they're very *** they're very similar, unlike trees. Trees -- they grow and they get smaller and bigger, and *** on the end aren't as very extendable. Very very nice, the *** very advantageous. If the algorithms start to exploit them. So you

should look to do that.

SLIDE 35:

Next, we'll look at the next foil, at the Stratix 4 multiplier systolic standing parallel FIRs. Have a look at this. This is very nice. The 16 coefficients map very nicely to four mults, which maps very nicely to the DSP block. Again it's good because Altera likes numbers that divide very well by 4. We have single-bit granularities and we don't worry too much about that. 3, 5, 6 is fine. But 4 is what Altera likes. So you can use all of the mults *** most efficient. But there's still a big problem *** fabric again *** finish my result. The accumulator has to go out into fabric. I guess I could use a DSP block only to do an accumulator. But then I'm committing a whole another DSP block to do the accumulate function. That doesn't seem like a very wise use of it. I would use the fabric *** cost me too much, but it's going to hurt my performance again. That's why we're seeing the significant drop-off in Altera's performance in these kind of functions. You can't avoid going into fabric. I have to accumulate over 4 cycles. *** memory *** very nicely here either. The M512s are good. They use the smaller ones. But they're just not small enough *** make up for granularity that the *** 16 so lovely to offer. So the *** 16 *** embedded shifting capability that you would

want in *** M512 they create a little cyclical RAM buffer, very much like *** does with their *** to do the shifting capability in the data core. Again big drop-off on performance. Much more resources. Needs more power as well. All this not looking too good *** Altera. And Xilinx again where you can get sufficient functions, high performance upwards of 400 MHz *** 400 MHz be great. Both of those numbers. It'll help you reduce the size. And remember, the whole design doesn't have to work at 400 MHz. A number of customers often come to me and say well how's the whole chip going to work at 400 MHz? It's not possible. Can't use fabric. It doesn't go that fast. Right, well it doesn't have to. Again in the HDTV example, that semi-parallel filter, you're using 75 MHz *** so the input is quite slow, quite doable, and then they ramp up inside the filter so the internals of that little filter *** before they drop off at the end to take the slow outputs. So what we've got is little islands of 400 MHz going to reduce the size of these functions. So you should look to do that *** interface might be going slower or some *** work layer or other functionality that might be going slower. But the key in these filter structures, anything that uses the XtremeDSP Slice, it's got to push up to the performance that it was meant to be used at.

SLIDE 36:

Right. We looked at three key examples here. And they're really good for emphasizing the differences between the two architectures, between Stratix-II and Xilinx. And look at the advantage of Xilinx. But let's look at something that may be more realistic to you hardware designers in the audience especially in the digital communications environment. Looking at the multi-channel multi-rate FIR filters here.

SLIDE 37:

Let's have a look at the Xilinx implementation. We'll break down *** next slide, 37. Here I've got interpolate by 2, 8 channels. I'm using the UMTS 3.84 megasamples which I can *** three particular options here. Because depending on the speed rate you're going to be shooting for. And I've also totaled up the total number of coefficients that are being required to process it. That's always good to get in mind. Don't think the channels have to be independent. One structure can easily do -- handle multiple channels *** multisection *** see here. You can see the total number of coefficients, over 1500 there. I looked at that and I thought OK 1500 coefficients, let's see if I go to that lower frequency clock how many multipliers would I require. 16 sequential engines there *** have to require, that'd be 16 DSP48s. How about Option 3? The bottom? That's around 430

MHz. It's putting the number of DSP48s down again by pushing the clock frequency and reducing the number of resources I have *** big desire then for us as designers, push that clock frequency up *** three different options *** depending on the speed grade I'm targeting. I went to the Option 3. That's the middle speed grade. I've shown the implementation on the next foil here.

SLIDE 38:

As you can see, on the front end I've got the TDM data. That's the time division multiplex data on the front. That's a bit much. People might say oh how do you get that big *** to work at 400? Well, the answer is it doesn't have to. It's 3.84 MHz, goes up to 15 by timesing it by 8 *** multiplexing. It's much much less than 400. 3 times 8 we're looking at roughly 24 MHz required there. Very slow, very easy to get the *** speeds. And it's the engine in the middle that starts to go up in performance results, ramping up to those high performance numbers of 400 MHz. *** close to the 450 number here *** you can see I've exploited *** again. In fact this is very much like the semipowered up structure we've been looking at earlier. Except we're using block memory to store our data instead of the smaller distributed memory. There's more demand here for that. Because the sample rate was slower, was in that 3.84 MHz

territory *** architectures to the slower data rates. Really good efficiency here. We need a funnel accumulator at the end again just like we saw earlier. The cascaded *** very well here to keep our performance up *** output all my 8 channels to a slower rate. I've totted up the slice *** for you. They're the ones in orange and you can see the performance there *** performance. That's what I like to see. That's great *** hardware designer. Then I put my Altera hat on again, said how would I do in Altera *** exploit their architecture the best.

SLIDE 39:

So the next foil you can see how I approached it from Stratix-II. *** using their filter compiler but *** approaching the problem. Oh, I've got 3 input adders again. That's really advantageous. That does help us in reducing the size of that adder tree. But there's still *** tree, which we've already discussed earlier, is a problem. I do like that *** accumulator. That's quite nice. And then the output trilogy nice down the bottom as well *** still got the same problem. I've got to go into fabric to complete my results. So because I went into fabric I shouldn't start thinking about slower clock frequencies *** they're blocking the highest speed rate was 450 MHz I believe is coming soon. And it's kind of irrespective if the fabric can't keep up. So I went for 295 MHz *** that

requires 20 multipliers. Automatically you need to *** significant number more multipliers to solve the same problem. Because their top frequencies are come down. More multipliers, more power, that's not good, and so that's not so good, and then lower performance as well. Also not so useful.

SLIDE 40:

Next foil *** bit closer at the DSP48 *** I put the performance numbers up I've got there. Significantly larger than Xilinx *** and the performance, as much as I tried, I couldn't get the 295 I wanted. Even by pipelining the adder chains, which is in the fabric, you can pipeline adder chains by breaking them, it cost me a wee bit more resources, as we're seeing here, and also it should boost my performance up, it still didn't give me that 295 I needed. It's not good. So I'd actually have to go back and have more multipliers than 20. At least 22 or 23 maybe *** so the moral of the story is if you use the XtremeDSP Slice correctly you can really start seeing major benefits for you as hardware designers and it keeps the system designers very happy as well. Because they like *** power going down as well as the cost.

SLIDE 41:

To wrap up the whole session, we'll start to look at some

digital up converters and down converters *** sorry and also a video *** this foil you'll see the digital up converter. The classic environment very similar to a gray chip *** the PFIRs, the compensation FIR, the CIC and the *** combining with the *** there. Now you see the CIC. That's always a very good one in the DSP48. I don't have time to go into it but bear in mind it's just a set of accumulators. You can use the XtremeDSP Slice as just an accumulator. So that's what we did.

SLIDE 42:

On the next slide you can see the system generator implementation I used *** we exploited all the techniques in this design I used earlier in this presentation. So please feel free to go back and look at this later to look at the different how I would have done each one. Again, CIC uses accumulators. So I can use that in a DSP48. I did it in V-II Pro and I also did it in Virtex-4.

SLIDE 43:

On the next slide you can see the resource and the performance. Again we're *** from a Virtex-II Pro and a greater than 50% reduction on the fabric because using the XtremeDSP48 to my advantage, I use that properly, using cascades and taking advantage of its performance gains I can really see big

reductions *** lower power, better cost.

SLIDE 44:

Final case study before we finish is the 2-D FIR filters. Again I put the maths up for the mathematicians in the audience. So you'd like to see it. This is a separable 2-D FIR. It is high-performance *** HDTV broadcasting would be where this would be most suitable. 24 taps, re-loadable coefficients.

SLIDE 45:

On the next foil you can see the *** being used. Very nice, it's got *** adder chains again coming to my rescue. Very good *** our algorithms and then the accumulator on the end *** big line buffers *** the lines here and then using a number of block RAMs. I've put the resource at the bottom. Also the equation down there shows you how I figured out *** number of multipliers that I wanted to use. Re-loadable coefficients you can see the distributed memories used there to store them. Each one *** 4 coefficients *** and then the other *** dual port distributed memories to load in the new coefficients as well *** switch over whenever we want to. Video customers often look to reload coefficients quite frequently. So this is a good example for that.

SLIDE 46:

Resources numbers on the next slide. You can see here V-II Pro versus Virtex-4. but the performance again, pushing the twice the performance and also half the size. Again it's very very useful if *** rearchitect my algorithms *** implementations, look for opportunities in my projects to try and do that so I can get the big benefits. Looking to cost-reduce, moving to Virtex-4, think about your algorithms. Think how adder chains can be exploited and how we can get those benefits. If I don't use them I can't get those benefits. And that's not so useful to us.

SLIDE 47:

So that leads me into my conclusion. The key points to remember. If you forget everything else just remember these three key, key points. OK. Very important that you push the clock frequencies up. Those are the kind of speeds that you want. You should expect to get these high performance numbers, 500 MHz, in the fastest speed grade, 400 in the lowest. If you're getting 110 something must be going wrong. Especially on those smaller FIR filter functions. OK so you must be able to push that up. So you'd expect that. If you're not getting that and you're the system designer, you should push your hardware engineers to do that. Or as a hardware engineer, you should be

expecting that. So push the performance up. That leads to reduced cost and reduced size and reduced power. All of that is excellent because the higher the performance the smaller the functions become. Also make sure you use all the DSP48. They're good on the power front, so let's use those. Don't use *** 28 or something. Make sure you exploit them. There's 40 op modes. Lots of them. Over 40 in fact. So you can use it in many different ways. You can also do vowel shifters and a number of other different functions. So even if you have them left over look for other functions to pull in. There's a lot more information on the web on this as well. And finally, think adder chains. Think adder chains. Not the adder tree. The adder tree has got its problems. DSP of tomorrow *** start to look toward adder chains because they have a lot of advantages. That's what *** some of these, especially it's evident here in Virtex-4 *** very important as well. Because there's a lot of detail *** detail and I don't expect you to remember it all. So it's important that you know where to find it.

SLIDE 48:

You can see there's 3 different app notes, 5 different app notes on the web. There's also a lot of other information your FAEs have on this. Please go to xilinx.com/dsp and go to the documentation and find those 3 app notes. And have a look

through them. Read up in detail about those semi-parallel, parallel and MACC *** structures. And they also have in the file *** education classes that can help you about this too. Learning how to use FPGA correctly for your algorithms is going to be very beneficial from a power, performance and cost perspective.

SLIDE 49:

So I think that brings me to a closing, the key points I'd like to say. I think David has something to say before we go into the question time about the feedback form. So I'll *** answer a few questions.

DAVID

Thank you, Niall, for that presentation. At this time we move into question and answer section of the presentation. If you have a question for Niall, please enter it now by clicking on the Ask a Question button and then typing your question in the popup window and clicking Submit. Also please take the time to open, fill out and submit the presentation survey. You can access the survey at this time in the Print Documents and View Links pull-down menu on the left-hand side of your interface. This survey will also pop open when you choose to close your viewer window or when the viewer window closes automatically at

the end of this webcast. Now let's go back to Niall for the Q&A.

NIALL BATSON

*** questions coming in here. There's a general question here, why does your context multiplier *** only work *** Virtex-4 DSP *** 500? Excellent question. Very good point. Most of the time, the marketing messages that you'll be hearing from our company push that faster speed rate. Also our competitor, also push the fastest numbers. But I think in most of the real world examples *** with customers, we tend to go for the slower speed grades. More *** they're usually cheaper. So *** provide an example that was using the lower speed grade. So it only went to 400. If I were to use the high speed grade, I could easily get 500 in that one multiplier complex *** example. So it's important to note again 400 is the glass ceiling we should be pushing towards *** DSP functionality in Virtex-4 in the low speed grades. Easy enough because *** 500 in the high speed grades, but I imagine most of you will be using the low speed grade.

Next question. Little question here, my understanding is that I can implement 4 multiplier *** Stratix-II *** why is that? Excellent point *** 4 multipliers and an adder tree in the

Altera block. And if I switch to multiply accumulate mode I lose two of the adders. So I could still do two multiply accumulates. Now it gives me another option. I could not use a multiply accumulate mode and just use it in multiply mode with their input and output registers and do the accumulator in fabric. That way I wouldn't waste any multipliers. That's a good option *** back to the story of in the fabric. There's enough fabric resources, performance will probably drop, reaching the *** number for the DSP block performance *** faster speed grade or adding slower speed grade to down in the 300s. It's going to be much much harder with that fabric *** so we've got a problem *** using up more resources, other problems there. And Xilinx, we can just stick to one *** and we can stay inside. That's another option and it can be done to not waste multipliers. But I'm now wasting the three adders. So whatever comes or goes I'm going to be wasting silicon. That's never good. Especially from a system designer's perspective.

Another question here. Why is the power difference between Xilinx and Altera less in the 1024 type *** example compared to the FIR filter? It's a very fine question, that, it's quite interesting. As we saw in the FFT example, it was dominated much more by fabric resources versus the multipliers. Again in our example I think we were using up about 16 DSP48s versus 2500

Slices. So most of the power is being consumed by the fabric. And if you go to the other TechOnLines we've had especially *** or go to the web, you'll see that *** have about equivalent power savings or power consumption in the fabric. So *** use the benefits of the DSP48, advantages you have, but it's still better *** and with a static *** system designer I'm certainly looking for the lower power solution. And if I can look to exploit the ***. It also leads me to another point about ratios. If you look at the ratios of the parts, the affect is very attractive *** DSP in terms of horsepower and also in terms of *** numbers of these DSP Slices to logic fabric. If you've got a *** DSP Slices and we can use them all up, we've got less fabric, so the *** than the part that's got a lower ratio. Like Stratix, for example. Stratix-II. It's an excellent question.

Another question here. What tools can I use to take advantage of the DSP solution today and can I *** C codes for programmable DSPs? It's a good question. I get that a lot and *** number of different solutions *** C code question, it's a hotly discussed topic, this is. Always every conference I go to it's always discussed, the next high level *** and there's a lot of interest in developing tools *** C code perspective *** I've yet to see really the winning solution on the C code to programmable DSP *** system generator and is a high level modeling tool. A

really good visual dataflow environment *** can take full advantage of this Virtex-4 architecture in there and *** all my examples *** generator and those reference designs available on the web. But from a C code perspective I think the jury's still out. There are a number of solutions available in the marketplace. And I encourage you to go and investigate those *** as well but the question is which gives the performance you need. System designers, hardware designers, you're all looking to keep the costs down. And also make time to market improvements *** power hits and cost hits. So want to keep performance up. So that's critical. So when you look at these tools, I'd highly recommend you look at the efficiency of them and *** make sure you're looking at the efficiency so you can get the advantages. It's 400 MHz numbers we should be pushing towards.

OK I think that's all we have time for. For the questions and answer. I thank you all for coming today. It's been lovely to talk to you. I hope this has been very beneficial *** start to look towards taking advantage of the Virtex-4 architecture. Especially that DSP48. OK. So back to you Dave.

DAVID

Great. I'd also like to thank everyone for attending today's

presentation of Design Techniques for Best Performance, Power and Cost, brought to you today by Xilinx and TechOnLine Webcast. I would like to remind you to please fill out and submit your surveys. Your survey will open when you choose to close your viewer window or when the viewer window closes automatically at the end of this webcast. By submitting the survey, you'll be providing Xilinx and TechOnLine with valuable feedback on the subjects covered in this webcast and also how we can improve the webcast product. This presentation will be available to all registered users in an on-demand format. You'll receive an email with information on how you can access this on-demand version of the webcast. Thank you again for attending. We hope to have you join us for future online webcasts. For a current schedule of live and on-demand events please go to www.techonline.com for a complete listing. Thank you and have a good day.

End of XilinxApr1505