Presentation Title: Introduction to VisualDSP++® Tools

Presenter Name: Nicole Wright

Chapter 1: Introduction

Subchapter 1a: Module Description

Welcome to the training module for the introduction to Visual DSP++® Tools. My name is Nicole Wright. I'm a staff engineer in the tools group and I will be presenting this module today.

This module will give you an overview of the Visual DSP++ Tools. For demonstration purposes we're going to be using the ADSP-BF537 EZ-KIT Lite® as a target. We're also going to be using Visual DSP++ 4.0 tools. We're going to show you some quick tips on how to analyze your application and fine tune them as well as some information about the tools.

In this module we're going to give you information on the Blackfin ® CROSSCORE ®

Product offerings, we're going to show you how to configure your EZ-KIT Lite. We're going to show you how to set up the hardware and how to configure the tools to use the EZ-KIT Lite. And then we're going to do a quick tour of the Visual DSP++ Tools. We'll take you through creating a project and we'll also demonstrate the plotting feature, which is a useful debugging feature. Then we'll go on and cover performance improvement using features the of the chip. More specifically we'll be using cache, L1 memory and the voltage regulator. Finally we'll be creating an Ethernet application using the features of the tools. We have a template already set up for you to use them.

Subchapter 1b: CROSSCORE Products

The Blackfin CROSSCORE product line comprised of three parts, there's a software component, which is your Visual DSP++ development suite. This contains all of your code generation tools, we provide device drivers and system service libraries. We provide an RTOS Kernel to use which is called the VDK. We have Code Wizards, a flash programmer, we have plotting and advanced code generation debugging tools as well as an automation API. The next portion of the tools offering or the CROSSCORE offering is the EZ-KIT Lite and daughter cards. We have EZ-KIT Lite boards available for the BlackFin 533, 535, 537 and 561. We also offer daughter cards to extend the functionality of the board. We have an audio and video card, we have a card that does high end audio. We have USB-LAN EZ-Extender card, and an FPGA EZ-Extender card. Lastly we have JTAG emulators. Our emulators support USB 2.0, USB 1.1, and PCI and Background Telemetry. Finally I'd like to let you know that there's free support with all of our offerings, and there are no maintenance fees for any of them.

Chapter 2: ADSP-BF537 EZ-KIT Lite Configuration

Subchapter 2a: Hardware Setup

And now I'm going to show you how to plug in the EZ-Kit.

First thing you'll need to do is to plug in your power supply. Once that goes on you'll see some lights blinking on the board. The next thing you'll need to plug in is your USB cable. The USB cable for this demonstration is already been plugged in to the back of the PC. This is a direct USB connection right into your PC. Once this gets plugged in you'll get a message saying that it's found new hardware. All you need to do is click the next button on the hardware wizard and your EZ-KIT device drivers will automatically be loaded on to your system. Now that I've shown you how to set up the hardware I'm going to show you how to configure the tools so you can use them with this target.

Subchapter 2b:EZ-KIT Session Setup

Now I'd like to show you the Analog Devices menu off the Start menu. Off the Analog Devices Visual DSP++ 4.0 there are four menu options, there's the Maintain this installation, a Visual DSP++ Configurator option, a documentation option and the environment. I'd like to show you how to maintain this installation option. This option will bring up a panel and it will show you that you can go to the Analog Devices website. We recommend you go there after installing your tools to check for any updates that may be available. Once you have gotten your updates from the website, you will come back to maintain this installation dialog box and apply your downloaded updates. For this demonstration we've already done that. So let's start the tools and I'll show you how to configure the tools for using the EZ-KIT target. You want to use the Visual DSP++ environment option. When the tools come up it's going to ask you to select a new session. The session that we want to use is the Blackfin Emulators Z-KIT Lite Target. From that session you can see that we've got several EZ-KIT Lites available. The EZ-KIT Lite that we are using today is a 537 via debug agent. And the session name you can change at your own will. We're going to use the default name today. So this will bring up the tools for us. As you can see the target that we're using is already created for us, the EZ-KIT Lite via debug agent. We have a project window, and we have the disassembly window. So so far we have completed setting the hardware configuration for the project, and configuring the tools to use this target.

Chapter 3:The VisualDSP++ Tools

Subchapter 3a: Creating a Project

Now what I'd like to do is take you on a quick tour of the Visual DSP++ tools. First thing I would like to do is we're going to create a project, but we'll have a simple program, just a simple hello world program that everybody is familiar with. You may need to manipulate your windows as you're going along to view them better, I'm going to do file, new, we're going to create a new project. This brings up the project wizard, which will allow you to create your new projects. You can use the directory name here to select any directory you want. I'm going to put my project in a "demo" directory. I'm going to call my project "hello". I'm going to create a standard application, go to the next screen, and it will tell you if your directory doesn't exist, that's fine I want to create it. Now I'm asked what type of output type I am. So it's just asking you what target do I want to create my project for. So, since we had the 537 processor on our target, that's what we're going to use. We hit the 'next' button, now at this point we're not going to add start up code, so we're going to go over this option. Now in the 'finish' screen you're going to see a summary of your choices, so you can double check to make sure that your project is created as you wanted it. Our file name is "hello.dpj", projects end in .dpj file extension. Our directory is in the "demo\hello" directory, it's a standard application. And you can see that our processors type is what we selected and that we have an executable file. Now I'm going to click on the 'finish' button. You can see that we have some folders that have been already created for us. These folders are where you keep your source file, your linker file, and any header files you may create. Today we're just going to create a simple C file. So what we're going to do is 'file' off of the 'new' to create a new file. We're going to just do #include and we're going to include our standard IO library, so we can get our printf. We're going to have a main function, now we're going to put our code for our main, which is just going to be a printf with a typical hello world. And let's close out that function, now we'll save our file. Notice the 'file saved' comes up in your project menu, project directory excuse me, that's where we want to be. So we're just going to call this 'hello.c' and save our project. At this point we have our project but our file has not been added to the project. So let's go ahead and add that file to the project by doing 'project, add to project', and we're going to add a file. You'll also notice off the project menu this is where we build our project, you can select your project options, and set your project to do what you want. You can update your dependencies, export or make file, you can build a batch file, set your configurations, and if you're using source control this is where you set up your source control. The file we want to add is 'hello.c', and you'll notice that once we add it there's a + by our source file to indicate that there's a file in there. So if we expand the folder you'll see the hello.c file. Now what I'm going to do is the next step is to build your file. So you do your project, 'build project'. And you'll note we have the 'build' window down here, and that's going to give you the status of what happens during your build. When your build is completed successfully it's going to load it on to the target. This is a project, an options preference that we have in the tools already. And by default this load upon build is set for you. A few other things you might notice is that we're at the printf statement, and it's in blue, this is to indicate where the program counter currently is. You'll also notice the arrow indicating that as well. We also have the red button which shows that there's been a breakpoint automatically set there for you. This again is another user preference that you can set yourself. So now that we've got everything built, let's just run it and see what happens. I'm going to use the 'run' icon, and if you just watch you'll see the 'hello world' printing in your console window. So so far we've created a project and we've run it just to give you a brief idea of the flow of what you need to do to create an application.

Subchapter 3b: Plotting

Now I'm going to close this project, and what we're going to do is I'm going to show you the plotting feature. I want to close all my source windows, start with a clean slate.
So let's open that project and this is in the default directory where your tools are installed, program files\Analog Devices, everything's off there. The project that we're using to plot is a simple C project that does sort. What you need to do to use the plotting feature is you have to build your project first. So let's do 'project', 'build project'. You can see that our project is built and loaded. So now that we have our project loaded and built what we're going to do is we're

going to configure a plot menu. So we that's off of a 'view' menu, debug windows, and you notice you have disassembly, trace, locals, expressions, just your typical debug windows that you want to see as well as the plot. We're going to create a new plot window, we're going to do a line plot, but as you can see from the pull down there's several other plot types that are available to you. We're just going to call this plot 'bubble' since it's the data from the bubble that we're looking at, the bubble sort. We're going to call our data set 'data set 1', we're going to look at the Blackfin memory and the address that we're going to use is the address the address for our array, which you can see right here in our program it's out_b, I'm just going to put the address right there. Our count, we're going to go for a 128 and our stride is 128 elements in our array that's the count of how many memory spaces, and then our stride 1, then for our data type we know that's an integer, so we want to look at the integer values of that. So the key thing here is you need to remember to hit the add button to add your data set before you leave this dialog box. So we've added our data set, we're going to hit the 'okay' button. And at this point you'll see our plot come up. Right now there's nothing in our array it's all been initialized to zero. What I'd like to do is I'm going to set a break point right before we start the bubble sort because as you can see we have a randomized array function before the bubble sort. So the randomized array will randomize the data in our array, then we'll do the bubble sort and then you'll see that the data has been sorted. To create a breakpoint you can do it one of two ways you can click on the line where you are and then click in the gutter, double click, there you go and you have your break point. Or you could do it from the icon. So we're going to run to this point and that's off of your Debug menu. You can see now that our data has changed, it's all randomized, our variables are all over the board, this is before we do the sort but because we started. Now what we're going to do is I'm going to run the program to completion and you'll see at the end that we'll be all set with our sorted variables. Again debug, run. There you go now you can see the bubble plot, our data is sorted, and this is a very good way for you to look at the values in your memory. This saves me having to actually print out or having it go into the memory window and look at every single item in there and check by hand. It's a good visual indicator of making sure that your data is as you like it. So there you go we've just completed the tour, quick tour of the Visual DSP++ tool, and the plotting feature.

Chapter 4: Performance Improvements using chip features

Subchapter 4a: Using Cache

Now what I'm going to do is we're going to show you the next section of our demonstration, which is how to use features of the chip to improve the performance of your application. I'm just going to close down this application, I want to close my source windows, I'm going to close down my bubble sort. Again you're going to have arrange your windows as you want them to get the look that you want. We're going to open the first project that we want, 'open project'. Again this is a project that has been created for you, it's in the kit and we'll go over that, I'll show you where that is later on. Again this is just a simple variation on our sort. One of the things that I'm going to show you that allow you to see how your project is doing is the statistical profiler. Now let's bring that up, 'tools', statistical profiling, and do 'profile'. There you go. I'm just going to get this running because this is an unoptimized version of this program. Everything is running in external memory so we'll have a little time to talk about things as it's running. So let's build and load our application, again you'll see in the build window you've got some information and you can see that we're loading, and we're completed loaded. One thing I'd like to show you here is in the console window if you right click you can clear the contents of the console window. Since we had some stuff in there I'm going to be looking at the output here in the console window it's a lot easier to see it when it's cleared . I'm just going to run this project. The statistical profiler polls the chip hundreds of times per second to see what is running in your application. So as you can see in our histogram most of the time is spent in our bubble sort since this is a slower sort than the quick sort this is what you'd expect. You'll also notice that we're running very slow because we're not optimized. You can see that we're running by the running in the bottom of your screen here. And again everything is running external memory so this is going to be kind of slow, we've purposely made it this way. Another thing about this statistical profiler is that as it polls the chip it's not intruding on your program. It's running in the background, there was no special code that we had to add.

Here we go we've completed our application. And you can see that our unoptimized application completed in 38 seconds, and it took over 9,000 million cycles to complete. Most of those cycles you can see from the histogram were spent in the bubble sort, 70%. The quick sort, spent some time there.

Then the next part of this demonstration, I'll close some windows so that we've got a much better view of what's going on, we're going to use cache. Which is the reserved memory on your chip, the Blackfin allows you to put instructions in that reserved memory. And the way that we do that is we go to 'project, options' and you'll notice if you scroll down you can set the options here for your compiler, your assembler, your linker and other options. What we want to do is go to the start up code. What we're going to do is we're going to look at the cache and memory protection section. It's very easy to set your instruction cache, you just go right here, select an option, 'instruction cache' and hit okay. And again this will run our project in the reserve memory, which will be much faster then running from external memory as we did before. One thing you need to remember to do when you do this is to rebuild your project. I'm going to rebuild my project for those actions to take effect, again we've done no programming, it's totally all through the GUI. Our histogram has blanked itself out because we're not running anything, and again let's just look at this, it took 38 seconds our first time through. Now let's run this, we should get much better performance this time around. Again you'll notice from the histogram that we're still spending a lot of time in the bubble sort as opposed to quick sort. If we look down in the output window we just completed that program in 3 seconds and just over 700 million cycles, which is much faster then what we had before.

Subchapter 4b: Using L1 Memory

The next thing that the chip allows you to do is it allows you place certain parts of code in internal or L1 memory. So what we're going to do now is we're going to turn off the instruction cache and we're going to put the bubble sort function into L1 memory. I'll show you how to do that. So again we turn off our instruction cache, we're going to go back to the project options, go to our cache and memory protection and we're going to set this back to RAM with no memory protection as we started out. And hit okay. To sort out data into the L1 memory we need to make a slight change to our C code. Since it is the bubble sort that I want to put in there I'm going to put one statement before our bubble sort declaration. We need to change this, segment used to be our old function, now we're using section which the is our section statement. We can see that I'm doing section, L1 code, bubble sort. So I'm saying put the bubble sort function into my L1 memory. Let's space this a bit for viewing so that you have a better view of what's going on. And we'll need to build the project again because by turning off the instruction cash we've changed our project and we've also changed our source code. So again we'll do the rebuild which will link and load our project. The disassembly automatically comes up and shows you where you are. So I'm just going to shut that down because we know we are. Now I'm just going to run our project, and you'll notice some interesting things happening here. Because we put the bubble sort right into that L1 memory, the time spent is a lot less because we're not going on and off between internal and external memory. So the quick sort which is not in that fast memory, takes longer. You'll also notice as well that our cycles that we're completing the project in has gone up a bit, it's just over 3,000 million cycles, and the time to actually run the application was a little longer. You can use any combination of using instruction cache or L1 memory to fine tune your application. With no fine tuning it was 38 seconds. We've gone to 3 seconds with the instruction cache, and back up to 13 seconds with our L1 memory.

Subchapter 4c: Using Voltage Regulator

The next way that you can optimize your chip is to set the voltage and frequency levels internally on your chip via the IDDE or by programmatically. So what we're going to do is we have an application that we're going to use that's going to show you how to do that. And we already have that canned project, we're going to open it up, again this is just a variation on our sort. I'm going to float this in the main window. I'm going to close the disassembly window just to give us some screen real-estate, and I'm going to open up this sort window. Before we look at the code, since this is going to take a while to build, I'm just going to build this application. I'm going to use the 'rebuild all' icon. So with Blackfin 537 you can set the voltage and frequency using the system services library. When you set the voltage the frequency for the chip is set to maximum frequency for that voltage. And that is done as I said by through system services library. What we're doing here in this demonstration is we have an array that we're passing in and we're using 4 voltage levels. And again when we set our voltage to .85 the system service library is automatically going to set that internally to the highest frequency. So since this is going to take a while to run, let's run this because we're doing it at three different speeds, and I'll show you a bit about where the calls are and how that's done. So we're going to show what the voltage is set at, the number of seconds it takes to run, and the number of cycles it takes to run. And again we have our 4 voltage settings that we're going to use. If we scroll down a bit we can see that we're using one call to the system services library to accomplish this. The call is adi_ power_init. Again if you'd like more information on this I suggest you go to the help system and you can look up the specifics of how to use this. For the purposes of demonstration we've set this up for you, and you can see we're starting to get some results. With our voltage at .85 it takes 30 seconds to run this simple sort. And again our cycles are very important to watch because as we're changing the voltage and the frequency our cycles are not changing but the speed at which we run the application is. There you go, so you can see that we cut the seconds almost in half by just a .10 voltage change. But our cycles stayed the same. And again you can see that we're running by the running indicator on the bottom of the screen. There we go our next application, next setting, and you can see here another thing you might want to look at, here's some more, we set command pairs for ez_kit-init this is the ez_kit_init which is again a system service library feature, very nice to use. Okay we've completed our application running. One quick thing to note that when our applications do terminate they terminate in the disassembly window at lib_prog_term and you'll also know that your application has stopped running by the halted message in the bottom of the status screen. Let's just take a quick look at our results, again you can see that they've proved what we've been talking about, as the voltages change and our frequency has been set to the maximum amount for that voltage, our seconds to run the program, have decreased, but yet our cycles have stayed the same.

In this section of the demonstration we've talked about three ways that you can optimize your program. You can use instruction cache, you can use L1 Memory, and you can set the voltage and frequency internally on your chip. Depending on how you'd like your application to run you may choose to use just one of these quick optimizations, or you may use a combination of them, it's totally up to you. But now you have the information on what you need to know for some quick things to try and get your application optimized quickly.

Chapter 5: Creating an Ethernet Application

Subchapter 5a: Using the Project Wizard

For the next part of this demonstration we're going to show you how to quickly get an Ethernet application running. We're going to use the program wizard template that we have created. So what I'm going to do is I'm going to close this application down, and we're going to create a new application using the project wizard. What I'm going to do also again is I'm going to clear out our output window. So we're going to create our Ethernet application using the project wizard. So go to click 'file', 'new', 'project'. This time we're going to have a little different settings then we did before. I'm going to call this project "Ethernet" and I'm going to put it in my "demo" directory under "Ethernet", it already creates that for you directly. Instead of creating a standard application we're going to create a TCPIP staff application using light weight internet protocol libraries, or LWIP, and VDK. VDK is our real time operating system that we use to drive the TCPIP. We're going to go to the next screen, again it's going to ask me if I want to create that directory, I'm going to say yes. We're going to select our processor type, again that's the Blackfin 537. We're not going to use any start up code this time. And now it comes to LWIP Wizard, the light-weight Internet protocol wizard is going to ask you what you're using for your standard target. This will allow it to create the correct device drivers and libraries in your project. Now we've completed our wizard, again we have the summary page of what we have. And again what you want to check for, we've changed our project type so that is reflected here, and notice that even though we've got a different type of project, our file name still has the .dpj extension. What you'll notice in this case when we create the Ethernet applications is that we have some code set up and our libraries are always set up there, we have a linker description file, and the kernel files for the RTOS are already set up for you. One of the things our template does it goes out to the internet and it pings it and it will get an Ethernet address for you or for the board. Before we do that one of the things we need to do is set the internet connection for our board. And there again that's just another cable you plug in to your board, and this is our Ethernet application, and that's our Ethernet cable. And what we're using for this demonstration is we're just going out to a router that just connects to the back of our PC. So we're talking between the PC the router and the board.

Subchapter 5b: Running the Application

So let's build this application, again we're updating everything. You can see how the progress of our build is going in the build window. And you're loading right now in to the chip. I'm going to run the code. And what we've put in here are some clues for you so with Ethernet you know what's going on. We've put in some printf statements that say we're waiting for our link to be established, our link has been established, and that we have an IP address. One quick thing I want to show you in our thread before we go any further, you can see, you'll see where the printf's come from. Then we started the stack for you, we put some kind of Cliff Notes in here for you. There you go you can see right here after the printf for IP address it's kind of giving you a clue this is where you'll want to add your application code. So the template is very user friendly, you'll be able to pick out from where the printf's are, what you add to your code, and do a little debugging in case you have a problem at this end.

Well now we have our IP address but let's test to make sure that our Ethernet connection is working correctly. What I'd like to do at this point, is I'm going to bring up a DOS window. What we're going to do is we're going to use the DOS ping command to go out and transmit data to the board via Ethernet and see how our transmission is going. We're going to use the IP address that's in the console window, you just type 'ping'. And again this is a standard DOS application so you should be able to do this on your machine. So we're going out into our mini little network here and you can see that we are getting data back and forth. So that our Ethernet application is indeed running correctly, we just didn't print something up for you, we actually did do the work and it is working as expected.

So this concludes the Ethernet portion of our demonstration. So so far just to wrap things up a bit, we've gone through, we've shown you what our CROSSCORE product offerings are. We've given you information on how to set up the hardware board and how to configure the tools to use the board. We've also gone through and done a quick demonstration of how to set up an application and a plotting feature. We've moved on to show you how to optimize your application using the product features of the chip. And then lastly we've shown you our Ethernet functionality that we have already in the template.

Now that we've concluded the demonstration portion of this module, I'd like to give you some places where you can go to get additional information. All of the sort projects used today are covered in the Getting Started with the ADSP-BF537 EZ_KIT Lite Manual. The tools manuals are provided in the online help to give you information about specific tools or using the tools themselves. You can also go to the Analog website; www.analog.com to find information on tools, tools update, engineering information. We also have two email support addresses available for you. If you have questions regarding the use of the tools we suggest that you send email to; processor.tools.support@analog.com. And if you have questions regarding the processor that you're using, please send them to; processor.support@analog.com. Or you can simply click the 'ask a question' button in this demonstration.

Thank you for viewing our demonstration.