Component-based DSP Programming

Save Time

Reduce Risk

Meet Schedule

RIDE® and VAB®

The Next Step in DSP Design:
Graphical Programming without Assembly or C

TI Third Party Network Member DSP TEXAS INSTRUMENTS

Hyperception
www.hyperception.com The Leader in DSP

"When Thomas Edison invented the lightbulb, he didn't start by trying to improve the candle. He decided that he wanted better light and went from there..."

For the past 15 years Hyperception has taken the same approach to improving DSP Development, resulting in our Graphical Component-based RIDE and VAB products for use in designing DSP applications.

The Next Step in DSP Design: Graphical Programming Without Assembly or C

Overview

When DSP design engineers sit down to write an algorithm or develop an application, they begin, almost invariably, with a block diagram. Using standard symbols and connecting lines, they refine their thought processes and map out the steps needed to achieve their goals. It's the way in which many engineers do their most creative thinking.

Unfortunately, block diagrams will not, ultimately, create their programs. Moving from a block diagram to an actual implementation involves creating code – lots and lots of lines of code. Traditionally, turning an idea into a workable program has meant struggling with arcane text lines written in Assembly language or C. Working in Assembly language requires advanced training in software engineering. C is not much more intuitive.

Hundreds – perhaps thousands– of great ideas have been lost because of the barrier that textual programming languages impose between the idea and its implementation.

What DSP design engineers need – and what the industry has struggled for years to create – is a methodology that allows them to go from block diagrams to machine-usable object code without intervening textual programming. This article discusses the need for graphics-based development and describes the first viable alternative to text-based programming. It outlines the advantages of a design methodology that can compile graphical representations directly into object code in much the same way that a C compiler translates textual programming into object code.

Background

Historically, textual-based representations for algorithms -- such as Assembly and C -- with their corresponding assemblers and compilers, have been used to produce DSP object code. Development of applications using textual representations as the basis for algorithms has required considerable expertise and generally has involved significant labor and time. Documentation and description of the system has been a separate set of tasks that have added even more time and development cost to a project.

In recent years, the industry has recognized that working in Assembly or C interferes with the creativity that DSP designers can provide. Various tool-makers have sought to obviate the need for textual programming and to create programming schemes that take advantage of the graphical representations that are more intuitive to most engineers. Several concerns have driven the effort to develop graphical DSP programming. Important ones include:

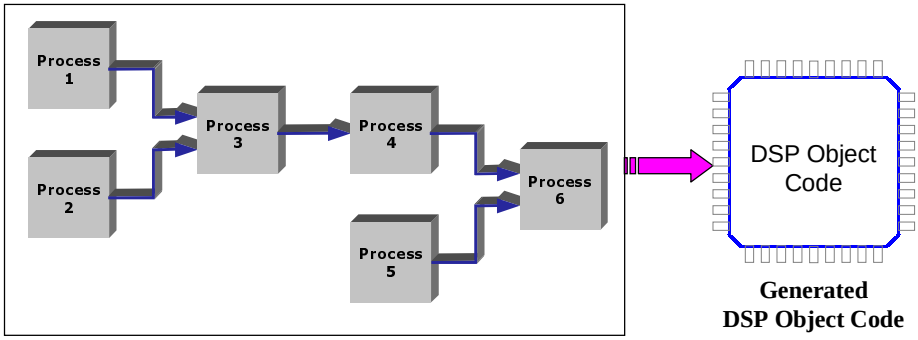
Time to Market. As technology advances and the pace of innovation accelerates, reaching market ahead of the competition increasingly spells success for a new product or product improvement. Conversely, presenting a new product after competitors have introduced similar, albeit inferior, products often means the death of a great idea. The complexities of Assembly or C coding add immensely to overall development time.

Learning Curve Requirements. Not only does writing code in Assembly or C require advanced training, it also limits the DSP talent available. There simply are not enough DSP design engineers skilled in Assembly or C. Design methodology that bypasses source code could allow countless would-be developers to implement their ideas.

Maintainability and Reusability. Frequently, code written by one DSP design engineer is so complex that only that engineer can understand it. Creating an improvement or reusing basic algorithms within different standards or as part of newer applications often requires the involvement of the original designer. If that engineer is not available, hundreds of hours of development work may be lost as a new design team starts from scratch. Because virtually all engineers can interpret block diagrams, graphical programming mitigates this problem.

Efforts to move toward graphical programming have, so far, yielded design methodologies that can interpret block diagrams and convert them into Assembly or C source code. For many engineers, this is a great advance, but it does not solve the underlying problem.

The code created by these methodologies still must go into a textual-based compiler. Modifying an algorithm or debugging the resulting object code requires mastery of Assembly or C. And this extra compiling step – compiling a graphic representation into C, for example, creates opportunities for error and often results in unnecessarily long and complex programs, as well as possibly introducing overhead into the design.



Block Diagram of Application

A block diagram of the system is created which is used to directly create the DSP Object Code for the application.

Hyperception's Development Tool allows direct DSP Object Code Generation from graphical block diagrams

A Step Forward

For more than a decade (quite a long time in DSP years), Hyperception has worked to overcome the Assembly/C barriers to DSP design creativity. Our goal has been to develop programming methodology that would allow DSP engineers to go straight from their graphical representations to optimized object code without the need to create any textual source code whatsoever.

With RIDE® and the introduction of its subset Visual Application Builder, or VAB®, we have enabled direct production of DSP object code from graphically constructed, user-created algorithms to achieve overall software designs for programmable DSPs. Many application areas exist. They include, but are not limited to, those found in digital radio, telecom, image processing, speech & audio processing, control, robotics, and wireless applications.

Development of algorithms with RIDE/VAB is similar to C in that variables, operators, expressions, and functions are used to create other functions, and/or the overall program. The difference is that each of these constructs is expressed graphically rather than textually.

RIDE/VAB can be thought of as a graphical DSP compiler. It allows an engineer to create a DSP algorithm from a graphical design or block diagram. Just as a C compiler turns a textual language (i.e., C) into DSP object code, RIDE/VAB turns a graphical language (i.e. a block diagram) directly into DSP object code.

As with a C compiler, RIDE/VAB permits development of algorithms for a variety of targets. These include industry standard DSP boards already on the market or custom hardware developed by the end-user. The same DSP simulators currently used with textual-based code development also serve for code developed graphically.

This technology addresses each of the three primary concerns discussed above. It reduces development time and helps improve time to market by eliminating tedious programming steps involving source code. It expands the talent pool by eliminating the need for experience and expertise in assembly, C and other textual algorithm development languages. And its innate self-documenting feature, along with the inherent simplicity of graphical design representations, enhances maintainability and simplifies migration from one processor family to another.

Leveraging eXpressDSP™ Real-time Software Technology

This innovative graphical DSP programming technology is based on an open software architecture and works synergistically with eXpressDSP foundation initiatives - Code Composer Studio™ IDE and the TMS320™ DSP Algorithm Standard. It also supports DSP/BIOS™ Kernel graphically. A RIDE/VAB tool, the **Component Wizard™ for eXpressDSP, eases the creation of standardized components and helps designers package their IP compliant algorithms with eXpressDSP.**

In addition to the eXpressDSP connection, which includes support for CCS, the TMS320 DSP Algorithm Standard, and support for DSP/BIOS, **RIDE/VAB allows for a simple method to harness components effectively.** Since **RIDE/VAB supports a component-based method of design**, a user who wishes to add intellectual property to an existing design, may do so by combining existing components to create new components. Alternatively, an engineer may leverage existing traditional code generation tools (C compiler, assembler) to create new components, which may then be used by RIDE/VAB. In fact, RIDE/VAB even allows for part of the design to be completed graphically, with the rest done through conventional means for users who are more comfortable with traditional tools.

RIDE/VAB also supports standard linkable libraries as well as object files created with conventional programming languages. The Component Wizard for eXpressDSP can automatically generate textual (C) base code required to develop a new component that adheres to the TMS320 DSP Algorithm Standard.

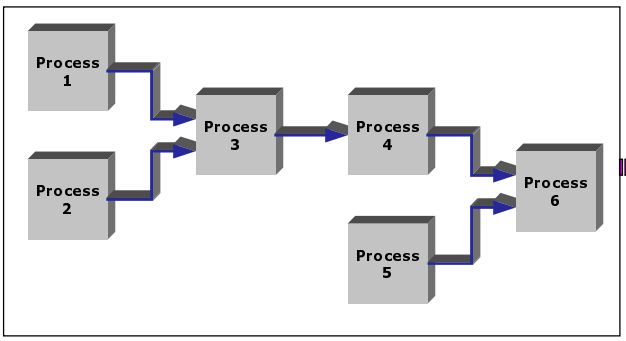
Programming Concerns

Graphical Component-based DSP design also addresses other common concerns about programming methodologies. These include, but are not limited to:

Application Speed and Size. Because RIDE/VAB may be used as a component delivery vehicle, able to sew together components to create an overall algorithm, its programming overhead is quite small. In fact, overall efficiency could even be greater than that achieved through the use of C because C was never designed as a language for DSPs. RIDE/VAB is tailored specifically to the needs of DSP designers.

Embedded Development Issues. In operation, RIDE/VAB closely resembles a C compiler/Linker, which produces DSP object code mapped to a specific piece of hardware. The linker typically uses information in a CMD file to associate physical memory, etc. to the algorithm code segments. RIDE/VAB does the same basic thing, allowing the designer to specify memory maps, etc. that fit a particular DSP target. The graphical design is then used to produce the COFF file (DSP object file), ready for downloading with a debugger, programming into a FLASH memory, etc.

Device Independence. RIDE/VAB supports TI's TMS320™ DSP family, including the TMS320C2000™ DSP Platform, the TMS320C5000™ DSP Platform, and the TMS320C6000™ DSP Platform. Support for TMS320C3x™ DSP Platforms is also available. The ability to move from one DSP platform to another allows convenient migration as DSP applications are moved to newer, more powerful, or less expensive DSPs.



Block Diagram of Application

In this case, a block diagram of the system is created which is literally running in real-time on a DSP target system. The Block Diagram is actually live and interactive, allowing the engineer to modify parameters 'on-the-fly' and observe actual real-time waveforms and data.

Live real-time interactive development on DSP hardware from graphical block diagram

A C6000™ DSP Design Example

So how does one design a DSP algorithm graphically? This example shows RIDE/VAB graphically harnessing the capabilities provided by the TMS320 DSP Algorithm Standard technology, namely a G.723 Encoder/Decoder, for C6000 DSP. The target DSP application makes use of eXpressDSP-compliant algorithms and is executed on a C6000 DSP-based hardware board (DSK6211 or DSK6711).

The application processes an audio input, such as a signal from a microphone. The RIDE/VAB worksheet applies several TMS320 DSP Algorithm Standard-compliant algorithms to the audio stream and then sends it to an audio output so that the results can be heard over a speaker. The data is also uploaded from the DSP for subsequent graphical display so that a visual comparison can be made between the original input signal and the decoded signal.

In using RIDE/VAB to implement this real-time DSP design, the user creates a "block diagram" representation of the algorithm by selecting individual block components through point-and-click. The designer drags each block component onto the worksheet by selecting from either a floating tool palette or a Function Selector tool. A user arranges the block components on the worksheet by dragging them with the mouse.

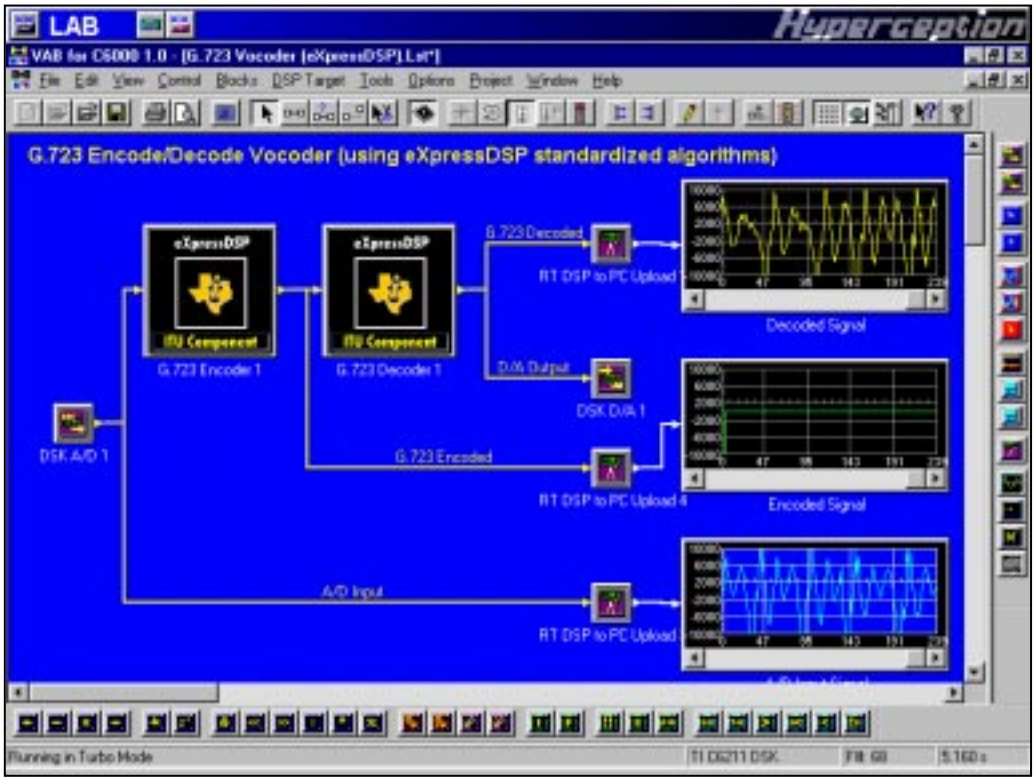
Individual component parameters (such as sample rates, framesize, etc.) can be user-specified by right-clicking on a block icon. The data flow for the algorithm is established by using the mouse to connect the block components together through line connections.

The resulting algorithm may then run directly in real-time on the target within the RIDE/VAB environment. No textual source code (C, Assembly) generation is required. If desired, the graphically designed algorithm may then be 'built' to a standard DSP object file for programming into ROM, FLASH, etc., or downloaded to other targets (including a simulator) via Code Composer Studio.

The figure below shows how the audio input is represented in a RIDE/VAB worksheet through a single A/D block component (located near the left-hand side of the worksheet). The block diagram depicted in the worksheet graphically displays the data flow for the algorithm. The eXpressDSP compliant algorithms used in this worksheet example are two vocoder algorithms that are represented by the G.723 Encoder and G.723 Decoder block components. In fact, these components are the same vocoder algorithms that Texas Instruments includes as part of its eXpressDSP Developer's Kit and they execute in the same processing time. The D/A block component in the worksheet represents the audio output data sent to the D/A converter.

The audio output will depend on whether the G.723 algorithm is used. The worksheet can be user-modified so that the audio input can be connected directly to the audio output (bypassing the vocoder encode and decode block components). The results can then be compared to those achieved when using the eXpressDSP compliant vocoder block components. In this example, differences become apparent when music that has a wide dynamic range is used as an input to the DSP hardware.

Methodology remains essentially the same for C5000™ DSP and C2000™ DSP applications. Any algorithm that can be created textually through eXpressDSP technology also can be created graphically with RIDE/VAB.



Simple G.723 Encode/Decode application graphically designed with RIDE/VAB for C6000

Summary

Until now, DSP design engineers have been hamstrung in their creativity by the limitations of text-based programming languages, notably C and Assembly. RIDE/VAB has eliminated the need for C or Assembly source code by creating object code directly from an engineer's block diagram.

The savings in design/development time as well as the advantages of maintainability and self-documenting nature of a graphical design may help broaden the use and popularity of DSPs in a variety of new applications. The savings in development time will help designers beat their competition to market with the innovations that customers demand.

Hyperception

www.hyperception.com *The Leader in DSP*

For more information, please contact:

Hyperception, Inc.
9550 Skillman LB 125
Dallas, TX 75243

Voice: 214-343-8525 Fax: 214-343-2457
E-Mail: info@hyperception.com