RAPID DEVELOPMENT OF RETINEX ALGORITHM ON TI C6000-BASED DIGITAL SIGNAL PROCESSOR

Juan Zapata and Ramón Ruiz
Departmento de Electrónica, Tecnología de Computadoras y Proyectos
Universidad Politécnica de Cartagena,Muralla del Mar, s/n, Cartagena, Spain

Keywords: Retinex Algorithm, Real-Time Signal Processing, Real-Time Workshop, MATLAB, Simulink.

Abstract: The Retinex is an image enhancement algorithm that improves the brightness, contrast and sharpness of an image. This work discusses an easy and rapid DSP implementation of the Retinex algorithm on a hardware/software platform which integrates MATLAB/Simulink, Texas Instruments (TI) eXpressDSP Tools and C6000 digital signal processing (DSP) target. This platform automates rapid prototyping on C6000 hardware targets because lets use Simulink to model the Retinex algorithm from blocks in the Signal Processing Blockset, and then use Real-Time Workshop to generate C code targeted to the TI DSP board by mean Code Composer Studio (CCS IDE). The build process downloads the targeted machine code to the selected hardware and runs the executable on the digital signal processor. After downloading the code to the board, our Retinex application runs automatically on our target. It performs a non-linear spatial/spectral transform that synthesizes strong local contrast enhancement. The library real time data exchange (RTDX) instrumentation that contains RTDX input and output blocks let transfer image to and from memory on any C6000-based target.

1 INTRODUCTION

With the rapid evolution in semiconductor technology in the past several years, digital signal processing systems have a lower overall cost compared to analog systems. DSP applications can be developed, analyzed, and simulated using software tools (Chacon and Valenzuela, 2004)(Assis de Melo and La Neve, 2004)(Gan and Kuo, 2006).

Due to the fast-paced nature of the digital signal processing applications and to the limited life span of new products, the time to market (TTM) is a very important figure of merit that is often overlooked. The rapid realization of an implementation from its concept to a product is of utmost importance. The scientific community in general, and the signal processing community in particular, have developed a number of methods for the specification of higher level algorithmic concepts and ideas. Two equivalents alternatives are graphical methods and language-based methods. Graphical method includes block diagrams, state diagrams and schemes for the design of virtual prototypes and language-based method includes hardware description languages (HDLs). Simulink uses graphical block diagrams to create models for real-time implementation of applications and then use Real-Time Workshop to generate C code targeted to the TI DSP board by mean Code Composer Studio (CCS IDE). The Retinex is an image enhancement algorithm based on human perception which provides color constancy and dynamic range compression. The algorithm is inspired by the work of Land's model for human vision. This algorithm was initially targeted to process multi-spectral satellite imagery but has found applicability in very diverse areas such as medical radiography, medical ultrasound imagery, aviation safety, and others. This paper presents the rapid prototyping and implementation of Retinex image enhancement algorithm on a platform based on a TI C6000 DSP target and Simulink/MATLAB/CCS development platform.

Zapata J. and Ruiz R. (2007).
RAPID DEVELOPMENT OF RETINEX ALGORITHM ON TI C6000-BASED DIGITAL SIGNAL PROCESSOR.
In *Proceedings of the Second International Conference on Computer Vision Theory and Applications - IFP/IA*, pages 149-152

2 SCHEME OVERVIEW

The key aspect of rapid prototyping is automated code generation. Under our scheme, the algorithm for a given application is initially described with signal-flow block diagrams with Simulink. Simulink is a platform for multidomain simulation and Model-Based Design for dynamic systems. It provides an interactive graphical environment and a customizable set of block libraries, and can be extended for specialized applications. Models built in Simulink can be configured and made ready for code generation. Using Real-Time Workshop, C code can be generated from the model for real-time simulation, rapid prototyping, or embedded system deployment. Figure 1 shows the general scheme for rapid prototyping based on MATLAB/Simulink and Texas Instruments eXpressDSP tools.

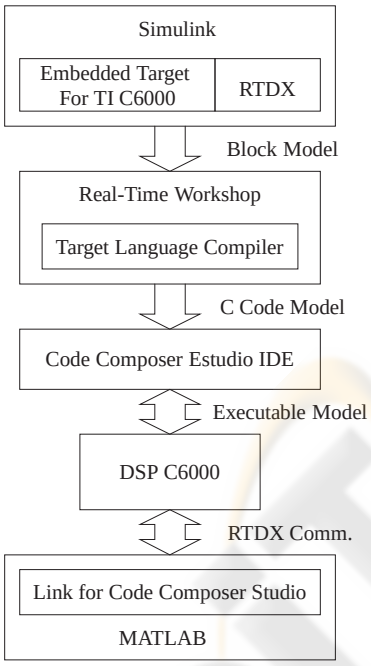

Figure 1: Development scheme of MathWorks and Texas Instruments eXpressDSP tools.

Real-Time Workshop generates and executes stand-alone C code for developing and testing algorithms modeled in Simulink. The resulting code can be used for many real-time and non-real-time applications, including simulation acceleration, rapid prototyping, and hardware-in-the-loop testing. Real-Time Workshop uses target template files to translate Simulink models into ANSI/ISO C code (Gan, 2002) (Chassaing, 2002). The target templates specify the environment on which this generated code will run. Own custom targets can be develop or use the ready-to-run configurations and third-party targets supported by Real-Time Workshop. Fortunelly, the Embedded Target for TI TMS320C6000 DSP (User Guide 3, 2006) consists of TI C6000 target (C6000lib blockset) that automates rapid prototyping on a C6000 hardware targets.

Embedded Target for TI TMS320C6000 DSP integrates Simulink and MATLAB with Texas Instruments eXpressDSP tools. TI development tools pick the C code generated with Real-Time Workshop (User Guide 6, 2006b) for a customized hardware target supported for Embedded Target for TI TMS320C6000 DSP and build an executable file for this target-specific processor. Additionally, one of Real-Time Workshop build options builds a Code Composer Studio project from the C code generated and, therefore, all features provided by Code Composer Studio work to help develop the algorithm or application.

Once target-specific executable is downloaded to the hardware and run it, the code runs wholly on the target and the running process can be accessed only from Code Composer Studio or from MATLAB with two powerful tools: Link for Code Composer Studio (User Guide 6, 2006a) and Real-Time Data Exchange (RTDX).

Link for Code Composer Studio lets use MATLAB functions to communicate with Code Composer Studio and with the information stored in memory and registers on a target. With the links, information can be transfered to and from Code Composer Studio and with the embedded objects, information about data and functions stored on the signal processor can be retrieved. Within the collection of hardware that Link for Code Composer Studio supports, some features of the link cannot be applied.

3 SYSTEM CONFIGURATION

In our scheme, the building process is initiated from Simulink, with a model or algorithm. Next, the Target language Compiler from Real-Time Workshop builds a program automatically for real-time application in C6000 environment. Using the make utility, Real-Time Workshop controls how it compiles and links the generated source code. Data can be sent, or received to the application through the RTDX channels.

3.1 Real-Time Workshop Configuration

Real-Time Workshop analyzes the block diagram and compiles an intermediate hierarchical representation

in a file called `model.rtw`. The Target Language Compiler reads `model.rtw`, translates it to C code. Real-Time Workshop constructs a makefile from the appropriate target makefile template. The make utility reads the makefile to compile source code, link object files and libraries, and generate an executable image, called model (UNIX) or model.exe (Windows). Figure 2 illustrates the complete process. The box labeled "Automated build process" highlights portions of the process that Real-Time Workshop executes.

After generating the code, Real-Time Workshop generates a customized makefile, `model.mk`. The generated makefile instructs the make system utility to compile and link source code generated from the model, as well as any required harness program, libraries, or user-provided modules. Real-Time Workshop creates `model.mk` from a system template file, `system.tmf` and where system stands for the selected target name. The system template makefile is designed for a specific target environment. Exits the option of modifying the template makefile to specify compilers, compiler options, and additional information used during the creation of the executable.

Real-Time Workshop creates the `model.mk` file by copying the contents of `system.tmf` and expanding lexical tokens (symbolic names) that describe a model's configuration. Real-Time Workshop provides many system template makefiles, configured for specific target environments and development systems.

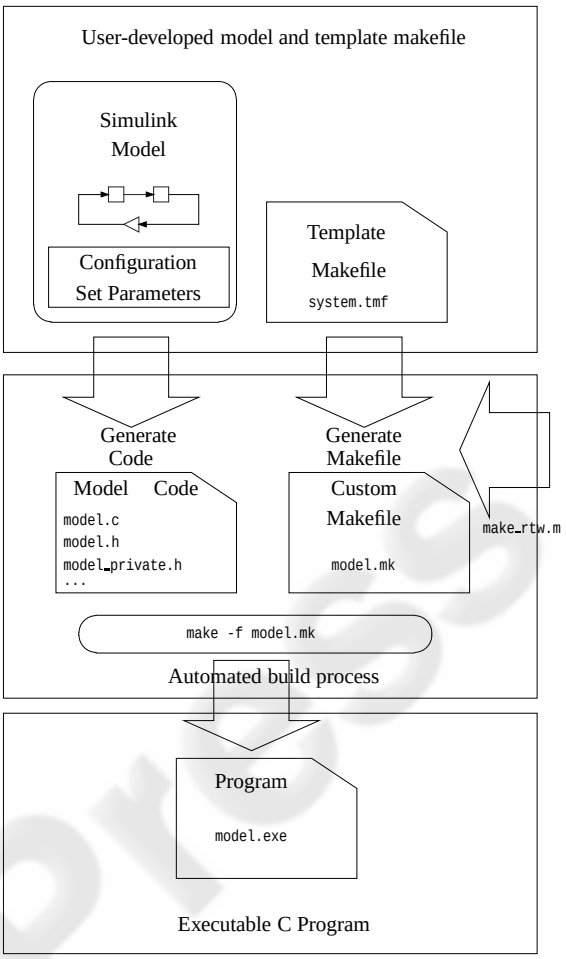

Figure 2: Automatic build process for Real-Time Workshop.

4 DEVELOPMENT OF RETINEX ALGORITHM

4.1 Retinex

The general form of the center/surround retinex is similar to the difference of Gaussian (DOG) function widely used in natural vision science to model both the perceptive and receptive processes. Expressed mathematically, this take the form

$$R_i(x,y) = logI_i(x,y) - log[F(x,y) * I_i(x,y)] \quad (1)$$

where $I_i(x,y)$ is the image distribution in the ith color spectral band, "*" denotes the convolution operation, $F(x,y)$ is the surround or kernel function, and $R_i(x,y)$ is the associated retinex output.

$F(x,y)$ is a Gaussian Filter defined by

$$F(x_1,x_2) = k\ exp[-(x_1^2+x_2^2)/\sigma^2] \quad (2)$$

where σ is the standard deviation of the filter and controls the amount of spatial detail that is retained, and k is a normalization factor that keeps the area under Gaussian curve equal to 1.

It is a direct result that color constancy (i.e., independence from single source illuminant spectral distribution) is reasonably complete since the image distribution, $I_i(x,y)$, value can be expressed

$$I_i(x,y) = S_i(x,y)r_i(x,y) \quad (3)$$

where $S_i(x,y)$ is the spatial distribution of the source illumination and $r_i(x,y)$, the distribution of scene reflectances, so that

$$R_i(x,y) = log\frac{S_i(x,y)r_i(x,y)}{\overline{S}_i(x,y)\overline{r}_i(x,y)} \quad (4)$$

where the bars denote spatially weighted average value. As long as $S_i(x,y) \approx \overline{S}_i(x,y)$, then

$$R_i(x,y) \approx log\frac{r_i(x,y)}{\overline{r}_i(x,y)} \quad (5)$$

4.2 Prototyping Retinex On Ti C6000-based Dsp

Once the retinex model was simulated by passing of different images and its performance was checked then the real-time C code for a specific target was generated through Real Time Workshop. Before two channels for RTDX communication were added. In this process, the executable application was built and downloaded automatically by CCS IDE in the DSP target. At last, real-time digital data was send for digital processing through two programs based on RTDX instrumentation. Processed images were received for the other channel in the same way for visualization with different tools of MATLAB.

This is not an impediment to effective use of tools provided for Code Composer Studio IDE. Moreover, these tools may sometimes be necessary to use, following the traditional practice of development of real-time applications using Code Composer Studio IDE exclusively. Code Composer Studio provides tools for configuring, building, debugging, tracing and analyzing programs. Texas Instruments DSP's provide on-chip emulation support that enables Code Composer Studio to control program execution and monitor real-time program activity. Finally to point out that the C code generated by this rapid prototyping platform is not the most efficient. To obtain a better code and increase the performance, there are techniques to improve and modify the C code generated. However, these techniques are not rapid and easy pricessly because designers must manually optimize the generated code in the code Composer Studio IDE. Alternatively, the code can be optimized by modifying the corresponding blocks of Simulink model and using others blocks from the preoptimized C62x and C64x libraries. When the code is generated, the Embedded Target for TI C6000 DSP produces function calls to preoptimized assembler implementations of the blocks, increasing the efficiency and performance of critical zones of real-time application.

5 CONCLUSION

We have successfully implemented a real-time version of simple scale Retinex image enhancement algorithm using a digital signal processor. This methodology allows an easy portability a other similar processors. We have discussed in this paper a design methodology of retinex algorithm based on cutting edge like MATLAB/Simulink, Code Composer studio IDE, and EVM/DSK hardware based on C6000 DSP of Texas Instruments. The paper described and illustrated the most important point to consider during the application of this technology. This technology lets develop and validate digital signal processing designs from concept through code, in a typical professional vision design simulation implementation. Moreover, the above discussion illustrated the use of the rapid prototyping system is a fully automated program building process, where the system is tested prior to the generation of the executable file. This saves tremendous amount of time, besides reducing the hardware cost. In the continued effort to train more DSP engineer, this type of technology incorporates an added profit to the formation of new experts in this knowledge area and can help to speed up the learning curve and implementation of real-time DSP applications. Another important benefit is that it avoids low level hardware work that can be tedious and very time consuming and therefore designers can focus their efforts in another important aspects of design of real-time applications. This work describes the steps needed to write and RTDX host application using MATLAB and the Developer's Kit for Texas Instruments DSP. Finally we illustrated this process with some applications presented in this paper and its feasibility is proved.

REFERENCES

Allensworth, D. (2002). How to Write an RTDX Host Application using Matlab. Technical report, Texas Instruments.

Assis de Melo, M.A.; Leonardi, F. and La Neve, A. (2004). Digital Signal Processing with Matlab and DSP Kits. In *Digital Signal Processing Workshop, 2004 and the 3rd IEEE Signal Processing Education Workshop. 2004 IEEE 11th*, pages 15–18.

Chacon, M. and Valenzuela, I. (2004). Fast Image Processing Application Development Scheme for the DSK C6711 using Matlab and Simulink. In *Digital Signal Processing Workshop, 2004*, pages 79–83.

Chassaing, R. (2002). *DSP Applications Using C and the TMS320C6x DSK*. John Wiley & Sons, New York.

Gan, W. (2002). Teaching and Learning the Hows and Whys of Real-time Digital Signal Processing. *IEEE Transactions on Education*, 45(4):336–343.

Gan, W.-S. and Kuo, S. M. (2006). Teaching DSP Software Development: from Design to Fixed-point Implementations. *IEEE Transactions on Education*, 49(1):122–131.

User Guide 3 (2006). *Embedded Target for TI TMS320C6000*. The MathWorks, Inc.

User Guide 6 (2006a). *Link for Code Composer Studio Development Tools*. The MathWorks, Inc.

User Guide 6 (2006b). *Real-Time Workshop*. The MathWorks, Inc.