C-Based Hardware Design Platform for a Dynamically Reconfigurable Processor

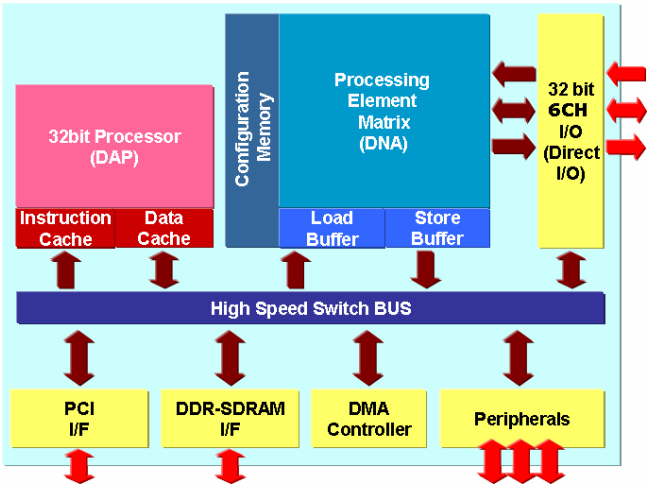
Phil Mulholland, IPFlex Inc. (phil@ipflex.com), Keisuke Ide, IPFlex Inc. (ide@ipflex.com), and Tomoyoshi Sato, VP and CTO, IPFlex Inc. (tomosat@ipflex.com)

C-Based Hardware Design

DAPDNA, a dynamically reconfigurable processor architecture, and its design tool, DAPDNA-FW, are a platform that offers C-based hardware design. In an example of image processing application, C-based compiler produced the best possible performance for a DAPDNA processor (166M pixel/s at 4 bytes of CMYK/pixel for basic 3x3 image filters) without manual optimization.

DAPDNA-2

DAPDNA-2, a processor using the DAPDNA architecture, contains a 32-bit RISC processor (termed DAP) and a dynamically reconfigurable two-dimensional matrix of 376 heterogeneous 32-bit processing elements (termed DNA), as shown in Figure 1. The processing elements (PE) consist of ALUs, RAMs, delays, counters, and I/O buffers. The reconfigurable array has four banks of configuration memory, and can switch among them in one clock cycle. DAPDNA-2 is fabricated with a 0.11-um standard-cell CMOS process and contains 12 million gates including 32 16-kB SRAMs. It runs at 166 MHz, performs 28 billion ALU operations and 9 billion multiplications/s, accesses 64-bit wide DDR-SDRAMs at 21 Gbps, transfers data through six 32-bit I/O channels (termed Direct I/O) at 32 Gbps, and dissipates a typical 2-3 W with maximum 7 W. DAPDNA-2 is well suited for stream processing such as multimedia, network, and cryptography.



DAPDNA-FW II

DAPDNA-FW II is a design tool for DAPDNA-2 as shown in Figure 2. Its major components include a compiler for a C-like language (Data Flow C, jointly developed by Celoxica of the UK) and a graphical design tool (DNA Designer), in which data streams are created by connecting PEs and predefined libraries.

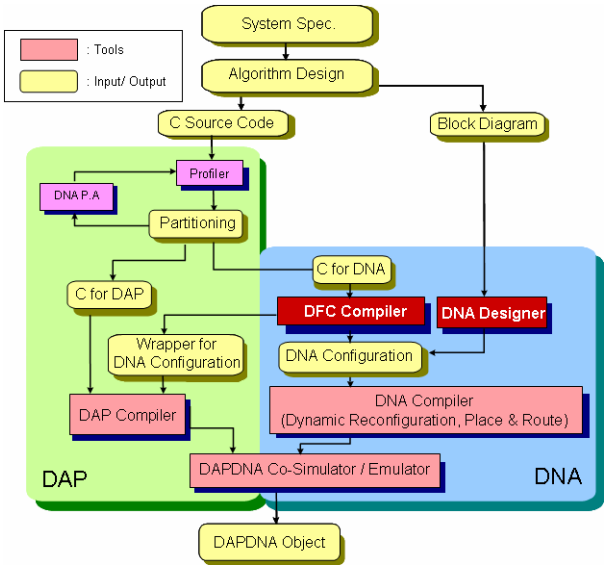


Figure 2: DAPDNA-FW II Overview

DFC Compiler

Data Flow C (DFC) is a variation of the C programming language designed for data flow processing architectures. It is largely based on ANSI C and Handel C. By using DFC, the user can directly control the algorithm translation onto the DAPDNA-2 hardware. Trade-offs and alternative translations can be quickly developed and evaluated by re-writing sections of the DFC code. The level of abstraction between DFC and the data flow hardware is much like the abstraction between C and a conventional DSP, and allows the user a great deal of control on the processing resource usage. Compiler overhead is kept minimal; in general, overhead affects only resource usage and not performance.

The DFC compiler translates C features directly to hardware features: e.g., a "for" loop translates to a set of counters, while external array reads or writes translate to streaming data sources or sinks. The translation is not always one-to-one, as some statements are sensitive to their context, but the translation is always predictable and consistent. DFC adds new constructs for hardware features that do not translate directly to the C language. For example, an array can be declared that acts like a tapped delay FIFO buffer. Other hardware features are available through a simple function call interface, such as a hardware-implemented bit reverse operation. DFC adopts a basic timing model, making the resulting hardware performance completely predictable.

The output of the DFC compiler is a function that DAP calls, like any other C function. This function, however, is executed on DNA, resulting in the corresponding high

performance. The DFC compiler achieves this in the following steps: 1. P&R the DFC code onto DNA, 2. create a C wrapper function which configures DNA and executes the algorithm within DAP's C code.

As an alternative, the DFC compiler can produce logical netlists instead of physical configurations. This logical configuration can be optimized and P&Red using DNA Designer. In the final stage of implementation, the DFC compiler can be used to create a C wrapper function for the modified configuration and the original DFC function.

DNA Designer

DNA Designer is a graphical design tool and cycle accurate simulator.

DNA Designer allows various PEs to be directly instantiated and then programmed via a set of parameter values. In addition, the user can use pre-designed or create his own parameterized and reusable macro blocks. All blocks can be freely placed and connected with drag-and-drop operation. This model-based development style would be familiar to Mathworks' Simulink tool users. (It should be noted a DNA Blockset is also available to provide the same DAPDNA development capabilities for Simulink.) Through the use of pre-programmed library macros, combined with custom design when necessary, the full performance of the DAPDNA architecture is extracted in an easily repeatable fashion.

All of the unique features of DNA are available for design and simulation, including the use of dynamic reconfiguration and Direct I/O. Using dynamic reconfiguration, the user can time-partition an application to reuse the same hardware resources. DNA Designer allows these configurations to be selected or sequenced using a switch/case metaphor. Using the Direct I/O, an application can be partitioned across multiple chips, or connected to external data sources and sinks.

Performance Figures

In an example of image processing applications, DAPDNA-2 processes 3 bytes of RGB data or 4 bytes of CMYK data per clock (equivalent to 667M monochrome pixel/s) in basic 3x3 image filters. The result was measured for 800x600 24 bit RGB images, with input/output frames stored in external DDR-SDRAM memory. Filter parameters do not affect the performance, and a 3x3 Mean filter, 3x3 Laplacian filters and 3x3 binary edge filters all give the same result. DNA processes the data as follows: an input data stream gained from external memory, followed by a series of line buffers, tapped delays, multiply operations, and a final summation followed by an output stream to external memory. All operations are pipelined and the 3x3 pixel operations are parallelized, resulting in a one-pixel-per-clock process. Such image filters can be designed graphically using DNA Designer and specific image processing library macros, or programmatically with DFC.

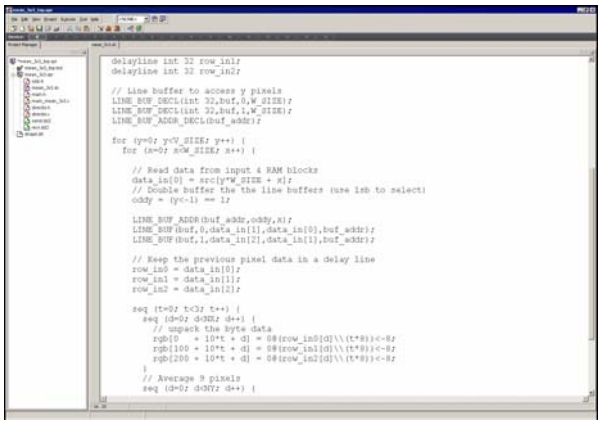


Figure 3: Data Flow C Source Code for a 3x3 Mean Filter

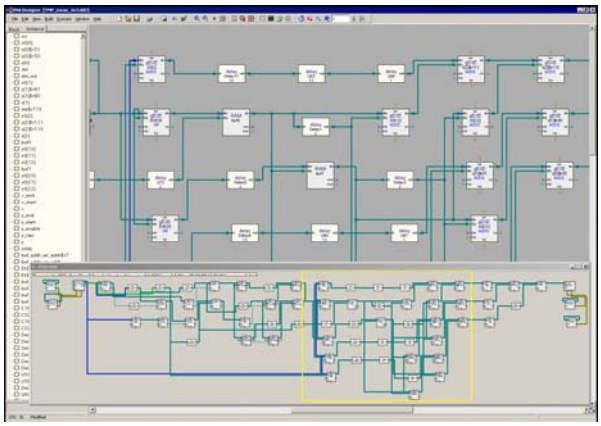


Figure 4: Output of DFC Compiler (Logical View)

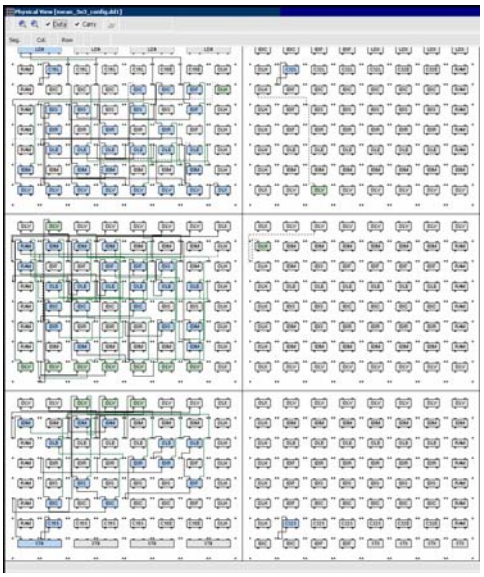


Figure 5: Output of DFC Compiler (Physical View)

Image filters described above produced similar result in DFC compiler and in DNA Designer. However, more complicated applications, e.g. motion estimation in video encoding, may benefit from optimization in DNA Designer. Typically, the user would develop in DFC first, and then optimized using DNA Designer to achieve better resource usage.