Implementation of Dynamically Reconfigurable Processor DAPDNA-2

Tomoyoshi SATO, Hiroyuki WATANABE, Kosuke SHIBA
IPFlex Inc.
Sun felista Meguro, 2-27-1, Kamiosaki, Shinagawa-ku,
Tokyo, 141-0021 Japan

INTRODUCTION

Application requirements for high computation power are increasing and becoming difficult to meet. Computer architectures based on program counters, the basis for RISC or DSP processors, have been used for a long time along with the other technologies, such as FPGAs or ASICs, for acceleration using parallel data processing. However, today the NRE of ASICs has become expensive and the TAT is getting longer than before. FPGAs have been used a one solution to the issues of NRE and TAT. However it is not always possible to meet the target performance, as they are limited by the complexity of the circuit generated by synthesis tool.

In order to solve these problems, we've developed the DAPDNA-2 as a high performance processor solution using dynamically reconfigurable technology, which, using parallel data processing, can provide more flexibility and higher computation power. The performance by DAPDNA-2 is very close to that of ASICs' and it is easy to achieve customers' application requirements in a short development period. The design has been implemented with Fujitsu 0.13 um CMOS technology. The gate count is approximately 12 million gates and the clock frequency is 166MHz. We have seen exceptional performance results in typical applications when compared with Intel's Pentium IV running at 3 GHz.

DAPDNA ARCHITECTURE

The major functional units are shown in Figure 1. DAPDNA stands for Digital Application Processor based on Distributed Network Architecture.

The DAP is designed as both a sequential execution unit and a controller for the DNA, the data processing engine that consists of 376 different type PEs. The DAP has five pipeline stages and it is possible to execute one instruction per cycle, and includes 8 Kbytes of instruction cache and 8 Kbytes of data cache. When more performance is required, the DAP utilizes the DNA by switching in a bank of internal pre-loaded configuration data on to the DNA. This is done in order to change the network structure and functionality of the PEs on the DNA and perform parallel data processing with the DNA. This dynamic reconfiguration can be done in one cycle. The DAP can operate independently of the DNA or it may suspend operation until it gets the result of parallel data processing by DNA.

The DNA consists of four double buffered data input and output buffers, six segments of PE matrices and four banks of configuration memory, as shown in Figure 2. There are several types PEs, as can be seen in Figure 3. Provided they have been pre-loaded with configuration data, configurations can be switched in one cycle. When more configuration data is required, configuration data can be loaded in from external DDR-SDRAM memory, as a background process. Once configured, all of the activated PEs on the DNA can be executed simultaneously without continuously feeding instructions. Generally in an application "C" source code we can easily find many looped expression or repeating processes. Most of these can be replaced as DNA configuration data. Especially in the case of parallel data processing algorithms these repetitive processes can be accelerated to up to 50 times higher performance when compared with Pentium IV at 3 GHz. In addition processing is done with a power consumption of only 2 or 3 Watts (see Figure 4.) In addition the DAPDNA-2 guarantees performance at 166 MHz regardless of the application mapping or fitting.

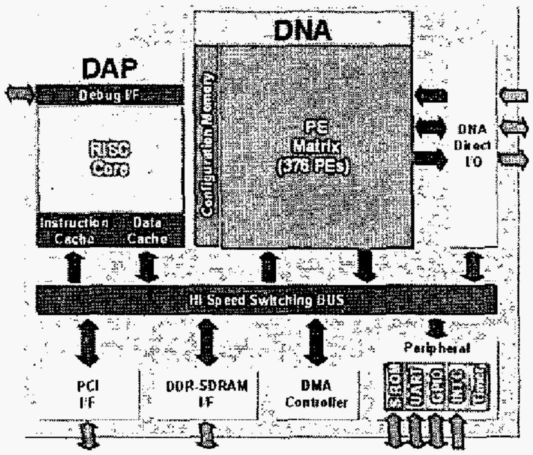

Figure 1

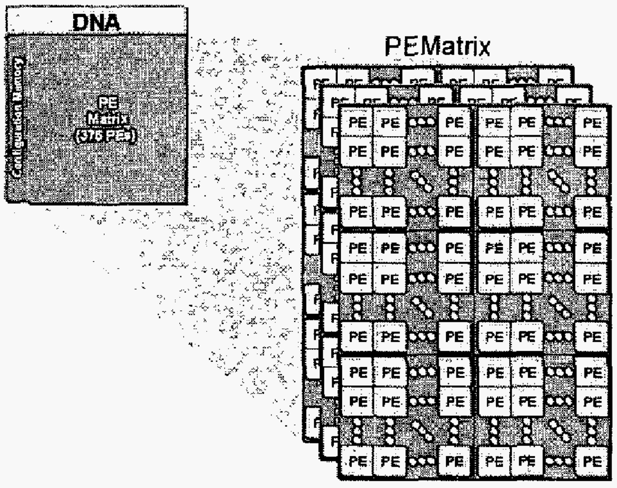

Figure 2

Cat.	PE Name	#	Functionality
Data Process	EXE	168	32 bit 2 inputs / 1 output execution element Variety of execution is done with two pipelined stages: 1. pre-processing stage (shift, mask, multiplier) 2. ALU stage (various arithmetic and logical operations) Includes 56 multipliers (16bit×16bit->32bit)
	DLE	136	32 bit 2 inputs / 1 output delay element Configurable delay length
	RAM	32	DNA internal memory element: 16KB (16KB×32 =512KB) Keeps content over multiple configurations
Data I/O	C16E	12	Address generation for LDB/STB access General purpose 16 bit counter
	C32E	12	Address generation for memory space General purpose 32 bit counter
	LDB	4	Input channel to DNA including double buffer
	STB	4	Output channel from DNA including double buffer
	LDX	4	Input channel to DNA from DNA Direct I/O
	STX	4	Output channel from DNA to DNA Direct I/O
Total		376	

Figure 3

Performance on Image Processing Algorithms

Algorithm	Pentium 4 (3.06GHz)	DAP/DNA-2 (166MHz)	Performance Multiple	PE Usage (%)
FFT (1024 pt) (us)	122.0	3.072	40	46
3x3 Average (M pixel/sec)	15.5	664	43	18
7x7 FIR (M pixel/sec)	3.0	166	55	27
Binary Pixel Expansion / Contraction (M pixel/sec)	32.7	664	20	14
Error Diffusion* (M pixel/sec)	11	304	28	43
Error Diffusion** (M pixel/sec)	6.2	110	18	33

* Floyd & Steinberg method
** Javis, Judice & Ninke method
Note: Simulator result. Simulation conducted by IPFlex.

Figure 4

DAPDNA-FW II ARCHITECTURE

An overview of the DAPDNA-FW II (FW II), the design tool for DAPDNA-2, is shown in Figure 5. When using the FW II, the first step is to compile an applications "C" source code and then to identify which sections are the bottleneck. The second step is to use the Profiler and to determine some candidate hot spots to be accelerated with the DNA. In the third step, you should compile the hot spots using DFC (Data Flow C), with some possible manual modifications to ensure the target performance. When the user requires much finer tuning, more elaborate parallel data processing can be implemented with the DNA Designer graphical design tool. The FW II also provides a Matlab/Simulink block set for DSP users. Some portability is possible for those users with a large amount of existing IP. We also provide a cycle accurate simulator for the DNA in order to get the same performance results between simulation on FW II and actual execution on the device. If necessary, it is possible to get an estimation value of the power consumption of the device for any given configuration.

APPLICATION MAPPING RESULT

Figure 6 shows an example of an application mapping. Usage of over 95% of the hardware resources by a mapping is possible by performing a manual design. The physical arrangement of the PEs is such that there are six segments, each a two dimensional array of PEs.

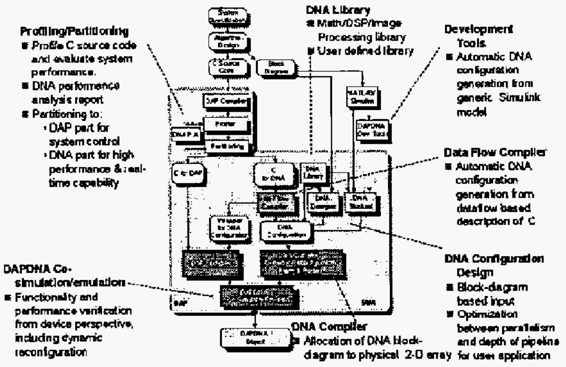
Profiling/Partitioning
Profile C source code and evaluate system performance.
DNA performance analysis report
Partitioning to:
DAP part for system control
DNA part for high performance & real-time capability
DNA Library
Math/DSP/Image Processing library
User defined library
Development Tools
Automatic DNA configuration generation from generic Simulink model
Data Flow Compiler
Automatic DNA configuration generation from dataflow based description of C
DNA Configuration Design
Block-diagram based input
Optimization between parallelism and depth of pipeline for user application
DAPDNA Co-simulation/emulation
Functionality and performance verification from device perspective, including dynamic reconfiguration
DNA Compiler
Allocation of DNA block-diagram to physical 2-D array
C Source Code
Block Diagram
DAP Compiler
Profiler
Partitioning
C to DAP
C to DNA
DNA Library
DNA Designer
Wrapper for DNA Configuration
DNA Configuration
DAPDNA Object
DAP
DNA

Figure 5

Every PE has two independent 32 bit input buses and one 32 bit output bus. Using these busses any PE in a given segment can be connected with any other PE located in the same segment. The bus structure provides flexibility to realize such sophisticated connections by allowing any PE to connect a fixed horizontal bus and it can choose one of either 8 x 32 bit horizontal X or Y buses as its output and any one of 8 x 32 bit vertical buses as its input. Multiplexers between the vertical and horizontal busses then complete the connection.

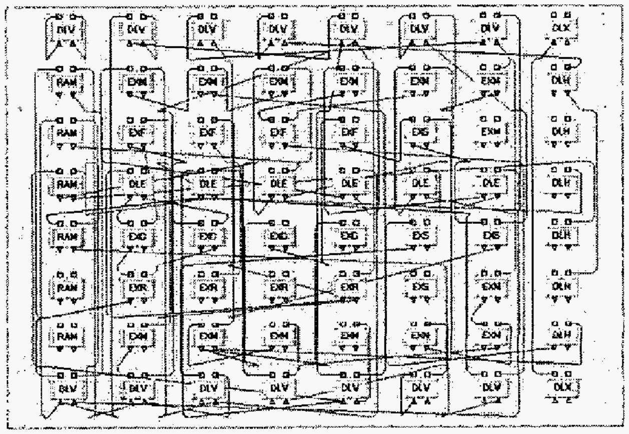

Figure 6

NEXT CHALLENGE

Recent deep sub-micron semiconductor process helps to realize this kind of new architecture. In given application areas, it obviously shows many advantages compared to existing technologies. We need to continue development of compilation technology in order to seek better solutions to shorten TAT and to make application design much easier. Unfortunately today there are some gaps between manual design and automatic design using a compiler. However we believe in that the issues involved in software and hardware trade-off are decreasing. Software requirements can change the hardware architecture to bring it closer to the ultimate goal.. With our application first policy we are aiming to create what we would like to call a "seamless environment", where the user feels little gap between the design of both software and hardware. Finally, in the near future, in order to meet various application requirements better than can be done with hardware design based on our current coarse grain architecture, the dynamic reconfiguration technology itself will be extended to a fine grain architecture.