



Cellot Inc

Cellot

FPCL

Field Programmable Cell Logic

From Idea to Product & ASIC

Using the FPCL as a Super Computer

CONFIDENTIAL

The information, data, drawings and the like contained in this document are proprietary to Cellot. The disclosure by Cellot of information contained herein does not constitute any license or authorization to use or disclose information, ideas or concepts presented. No part of this document may be disclosed or used for any purpose other than the review by the recipient. Information in this document is subject to change without notice and does not represent a commitment on the part of Cellot Inc.

Reconfigurable / Adaptive Computing Systems, Super Computers

The FPCL device can implement not only common hardware, but also complicated applications / complicated functions normally implemented utilizing a processor type of hardware (e.g. CPUs, DSPs). The FPCL execution time is expected to be faster than that utilizing a very fast processor by an order of magnitude as the processor activates its commands one by one while the FPCL hardware is working in parallel on thousands of commands. For example: if the FPCL implements 2000 commands in a single clock and its frequency is 250 MHz, the equivalent command rate is 500 GHz (or 500,000 MIPS). Combining several FPCL devices to work in parallel, will lead to enormous computing power.

The application can be downloaded into the FPCL devices and the needed super real-time computing power is achieved. Nevertheless, even a small number of FPCL devices can convert a regular computer into a super computer.

The fractal like structure enables the FPCL device to be converted into a single cell, which is nothing but a Random Access Memory that can be mapped into a CPU memory.

The base for such a supercomputer implementation is a collection of several FPCL devices each of which is to be programmed to carry out a required operation. A host computer, which has a memory, stores the code that must be loaded into each device so as to allow it to operate. While one device is executing the application, a second is connected to the host as normal memory allowing previous results to be read there from or for new code to be loaded thereto, and at the same time the code for effecting the required connections and loading the required data may be loaded into the third device. Upon completion of an application by the currently running device, the states of each device changes, and the host is connected to the device that has just finished its task for reading the result and loading a new implementation.

By such means, each implementation may be effected in hardware which is much faster than can be done in a normal CPU using software, whilst the downloading of new data to the devices, which is time-consuming, is done in parallel with the operation of a different device and therefore represents a transparent operation not demanding any overhead in real-time.

This technique can be useful when a very fast real-time computing is needed (as in configurable computers). The host computer will change the task to be implemented as needed on one device, while in the other device it fetches the result of the previous instruction.

The FPCL Development Environment already has the ability to create FPCL objects - the foundation stones for an FPCL application – in runtime. These objects may be integrated into high-level language (such as C++) enabling the engineer to create, manipulate and use the FPCL components in this high-level language so as to enable automation of the described technique.

The same concept of changing the application by the running application itself can be implemented not only using a controller, but as part of an application implemented on a single device. To simplify the explanation – imagine the FPCL device is built out of several devices (the fractal-like structure enables this) use one as a controller and implement the above procedure. Such technique may be applied to evolvable hardware.