



Cellot Inc

Cellot

FPCL

Field Programmable Cell Logic

From Idea to Product & ASIC

**The FPCL as a Digital Signal
Processor**

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.

Utilizing the FPCL for DSP Applications

Cellot's innovative FPCL technology has been designed to support "complex multi-million gate functionality" in a single chip. Its simple and fast developer environment along with the ability to use high level language such as C++ in order to develop applications utilizing the FPCL along with its massive computing power make it a perfect solution for Digital Signal Processor application. Below is an example of FFT implementation.

An Implementation Example: FFT

Several FFT implementations have been investigated for different sizes and different lengths, for speed optimization and for size optimization, which shows the great advantage of the FPCL technology. Below is the result for a 1024 points, 16 bit complex (16 bits real, 16 bits imaginary) FFT implementation. Investigating some algorithms, it has been found that the Cooley-Tukey radix-4 algorithm is best for speed, while the Cooley-Tukey radix-2 algorithm is best for size. The latest could implement the whole FFT in a few cells, but would take a long time for the transform. If time-sharing is executed, dozens of thousands of transforms could be performed, but the time for any transform would be counted in seconds.

The implementation presented below has been designed for speed optimization therefore the radix-4 algorithm has been chosen. This implementation is a "straightforward" implementation result and we may implement the FFT in less real-estate (same transform time) or faster (same real-estate).

Please note that the transform time presented IS NOT the best transform time the FPCL can achieve even with this "straightforward" implementation, but is the transform time achieved for the number of cells mentioned. The transform time is divided by two each time the real estate is duplicated.

Based upon the Radix 4, Cooley-Tukey Algorithm and no accuracy compromise, a sample of the indicative results for the 1M Bytes chip is presented below. For other chip sizes the Transform Time is directly linked to path delay.

As the FFT can be implemented utilizing core only, the results are provided for such case. Anyway, as normally common PLD do not implement the FFT using core only but need the assistance of memories and DSP blocks (or multiplier or multiplier-accumulators) which may be added into such PLDs, some results are shown in case the FPCL utilized such accessories. Again, the transform time is divided by two each time the resources are duplicated.

FFT TRANSFORM TIME (Micro-Seconds)	NUMBER OF K Bytes USED	REMARKS
4.0	450	FPCL Standard Cells Implementation. One butterfly.
2.5	450	FPCL Full Custom Implementation. One butterfly.
24.5	80	FPCL Standard Cells Implementation.
15	80	FPCL Full Custom Implementation.
16.4	20	Using 4 Hardwired Multiplier-Accumulators and Standard Cells.
10.2	20	Using 4 Hardwired Multiplier-Accumulators and Full Custom Cells.

Better transform times may be achieved. For example, using 24 multiplier-accumulators and 150 K Bytes (or 900 K Bytes, utilizing core only) Full Custom implementation will lead to 1.25 microseconds transform time. This transform time is divided by two each time the resources are duplicated.

There is room to achieve better results, which have not yet been investigated:

- Compromising accuracy in the same manner other providers do,
- Using 16 bits Floating Point so the multiplication is a lot simpler,
- Optimizing the indicated implementation,
- Investigating other algorithms possibly better suited for FPCL (WFFT),
- Using LNS (Logarithmic Number System).

Combination of a DSP and FPCL

The FPCL is designed with a built in controller used for the “Built In Test and Auto Recovery” feature of the device. The performance needed from this controller is poor, nevertheless, as the controller has access to all the cells in the FPCL device it can be used as part of the user application. If such controller is replaced by a stronger Digital Signal Processor, the device resulted can have the benefit of the two methodologies: using both processor and hardware for the best advantage of the user.