

# Fast Reconfiguration Through Difference Compression

Irwin Kennedy  
Division of Informatics  
University of Edinburgh  
Edinburgh, EH9 3JZ  
iok@dcs.ed.ac.uk

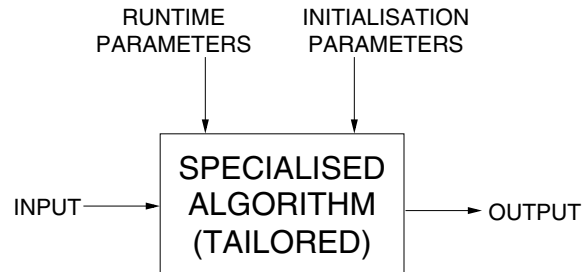
## 1. Introduction

Advances in the configurable logic fabric's architecture, together with the increasing hard-wired integration of commonly used cores such as giga-bit I/O transceivers, multipliers and processors suggests that statically configured FPGA platforms will continue to become more competitive and therefore gain further market share at the expense of the ASIC. In addition, the Semiconductor Association predicts that the percentage area of memory in a System on a Chip (SOC) will continue to increase, with 70% of a SOC area devoted to RAM by 2005. However, it is not certain whether it is valid to combine these observations and extrapolations to predict the demise of the ASIC as some FPGA vendors believe. One of the main arguments made against this prediction is the size of the silicon area gap between an ASIC solution and an FPGA solution is largely attributed to the area inefficiency of the FPGA configurable logic fabric. Dynamic partial reconfiguration can help significantly reduce this area inefficiency gap.

## 2. Reconfiguration Architecture Domains

The frequency and the application of reconfiguration can be captured by three broad domains listed below. To illustrate each of the use cases, an example will be given from the domain of cellular communications.

- Localisation: A Custom Computing Machine's (CCM) functionality may depend on its locality and environment. For example, the radio baseband processing unit in a mobile phone may be required to operate with several different air interfaces depending on its geographical location. The frequency of reconfiguration for localisation is low, perhaps occurring only once in a machine's lifetime, or, as the device is physically relocated to a different geographical location.
- Specialisation: An important advantage FPL based CCM's have over their ASIC alternative is the ability



**Figure 1. Algorithm Implemented on a Reconfigurable CCM Fabric**

to dynamically tailor their functionality to suit the precise task to be performed. An ASIC design is built to a rigid specification, usually dimensioned by what are called the worst-case operating conditions. For example, the precise implementation required of a Forward Error Correction (FEC) algorithm will vary with time and can be constructed from a set of values on the fly as illustrated in Figure 1. The base parameterisable circuit forms part of the initialisation parameters, run time parameters are fed in from other systems and the input data stream is used as an additional mechanism for triggering an implementation change. The algorithm may be optimally implemented by changing its implementation according to the data rate and run-time parameters such as channel conditions could result in a more approximate (and hence simpler) implementation of the algorithm.

- Dynamic Circuit Switching: An extreme form of circuit specialisation is to change the functionality of an area of the fabric entirely. This paradigm is so powerful in its own right that it is distinguished here from specialisation. An example from wireless communications might be a base station dealing with many voice

calls and no data calls; high-rate turbo decoders used for data calls in a rigid ASIC implementation would go unused, but in a reconfigurable solution, the turbo decoders could be switched out of the fabric and replaced by convolutional decoders for voice call processing.

One of the central aims of dynamic reconfiguration is to perform useful work by making the most efficient use of the silicon resources available. In applications where many thousands of specialisations per second are necessary to have the optimal algorithm implementation, the time to reconfigure the fabric would need to be much less than a milli-second, otherwise the benefit of specialisation is lost. In fact, the three use cases of reconfiguration illustrate that the required speed varies substantially with the application being targeted. This is an important point.

### 3. Reconfiguration Architecture Design Space

The two main architectural innovations proposed in the literature to speedup reconfiguration are configuration compression [1] and the multi-context configuration store [2]. Configuration compression by Hauck et. al, presents an algorithm which reduces bitstreams for the Xilinx Virtex FPGA by around 75%, and requires the addition of a decompression unit to the configuration architecture, speeding up reconfiguration by reducing the amount of data that must be loaded on chip. The multi-context configuration store has multiple memory cells per bit to be configured, forming configuration planes, enabling rapid switching between planes and hence active device configurations. The two innovations approach extreme points in the tradeoff space between silicon area and reconfiguration speed. The multi-context solution achieves several orders of magnitude speedup in reconfiguration time at a significant cost in silicon area, and the configuration compression technique achieves a small speedup of 2-3X for only a small increase in silicon area. The following subsection examines the architectural design space which spans these two approaches and suggests a technique to exploit it.

#### 3.1. Exploiting Bitstream Redundancy

There is high resource redundancy in any particular circuit instantiation on an FPGA. It follows that this applies equally to the bitstream — only a small fraction of the bits in a reconfiguration bitstream are important, the value of the majority of bits does not affect the circuit (don't care values). Compressing only the required changes to the existing configuration memory and then storing them on chip ready for the reconfiguration time is a new architectural design space in which a tradeoff between reconfiguration time and silicon area exists. With relatively straightforward compression techniques such as runlength encoding, the on-chip

decompression unit can be made very small to facilitate parallelisation of the decompression process. The architectural space allows the level of parallelisation, and the amount of on-chip RAM to be tailored, hence providing control over the tradeoff between speed and area. Storing changes on chip significantly reduces the time during which the fabric isn't processing data due to reconfiguration, and since only the changes are stored rather than the configuration of every resource bit, it makes more efficient use of onchip configuration RAM compared to the multi-context device.

### 4. Future Work

The ultimate goal of future work is to demonstrate the advantages offered by improving and fully utilising reconfiguration over the static equivalent.

A method of measuring the benefit of adding configuration memory to an FPGA will be investigated, ie a way to determine the optimal point between a multi-context device and a single context device. Typical operational characteristics, maximum and minimum reconfiguration times, system requirements etc. all determine such a point. Physical layer processing in a cellular phone base station is chosen as the domain of study because of its excellent qualities for such a metric to be tested. Its complex requirements, changing operational environment and computational load will also be used as a case study for exploring designs able to take full advantage of the flexibility offered by a reconfigurable CCM, and a comparison made with the static equivalent.

### 5. Acknowledgements

This work was sponsored by Lucent Bell Labs and the Engineering and Physical Sciences Research Council.

### References

- [1] **Zhiyuan L. and Hauck S.**, Configuration Compression for Virtex FPGAs, *IEEE Symposium on FPGAs for Custom Computing Machines*, 2001.
- [2] **Tau E., Chen D., Eslick I., Brown J. and DeHon A.**, A First Generation DPGA Implementation, *FDP 95, Canadian Workshop of Field-Programmable Devices*, May 29-Jun 1, 1995.