Soft Circuitry

Gordon Brebner
Department of Computer Science
University of Edinburgh

February 1998

Contents

What this talk is about:

- A software-oriented approach to configurable circuitry
- Producing soft circuitry (brief)
- Soft circuitry as a system component
- Virtual circuitry
- Mobile circuitry: Circlets

Software-oriented approach to circuitry

Starting point:

- seek common factors between stored circuitry and stored programs
- apply long-established compiler/operating system principles

Differences between circuits and programs:

- parallel v. sequential
- two-dimensional v. one-dimensional
- tighter v. looser structure
- synchronous v. asynchronous

Key similarity between circuits and programs:

- both are essentially state machines computing outputs from inputs

How to produce soft circuitry

[Controversial] How *not* to produce soft circuitry:

- try to use traditional (map, place, route) tools
- try to write compilers for high-level programming languages

How to produce soft circuitry:

- probably need a new kind of high-level representation

Pending this advance, experiment with low-level representations. Work at Edinburgh:

- have SPODE: block/interconnect definition within C control structures
- currently merging SPODE with Guccione's JERC: Java-based dynamic reconfiguration system
- investigating genetic algorithms for circuit design

Soft circuitry as a system component

Soft circuitry has a hardware environment:

- implemented on particular FPGA technology
- might interact with memory or processors

and it has a software environment:

- interaction with programs written using various paradigms
- managed by operating systems

Amongst other things, this necessitates standard interfaces:

- at the physical level
- for bit-level interaction
- for functional interaction

Virtual circuitry

Basic ideas:

- virtual circuitry is an analogue of virtual memory
- the central component is:
  - the Swappable Logic Unit (SLU)

  which is the analogue of a page or segment
- note: circuitry cannot just be chopped up into equal-sized pages for virtual memory management

As an interface to the world of operating systems, an SLU:

- executes within a supportive environment

As an interface to the world of programming, an SLU:

- roughly corresponds to a function or procedure

Supportive environments for SLUs

Two particular environments have been designed and investigated so far:

- the ‘sea of accelerators’
- the ‘parallel harness’

(there is also a ‘raw’ environment for system SLUs)

Supporting the environments raises two familiar issues:

- placement
- routing

The novel twist is that:

- placement and routing has to be done dynamically by the operating system

Dynamic placement and routing

For the sea of accelerators:

- placement is the issue: two-dimensional version of page management for virtual memory

- no routing, since SLUs are independent

A key bonus for placement is:

- an FPGA technology allowing direct reads and writes to registers located within SLUs

For the parallel harness:

- placement is very regular and so easy

- routing between SLUs in the harness is the issue

Adam Donlin will talk about routing issues, in a more general soft circuitry context

Mobile circuitry

Soft circuitry offers various novel possibilities. One is:

- portable circuitry in a network computing environment
- that is, applets expressed as circuits

The name 'circlets' has been invented to describe these applet circuits

First prototype:

- SLURP: the SLU Remote Procedure server
- manages a collection of executable/executing SLUs for clients

This directly leads on to a system for managing a collection of circlets for a network computer user.

Circlet issues

There are two main issues for circlets:

- how are circlets represented?
- what execution environment is provided for circlets?

Possible (graph-based) circlet representations:

- raw configuration data (not portable)
- multiplexor/LUT circuit description
- boolean logic circuit description
- higher-level circuit description

Current favourite:

- fine-grain abstract array architecture
- simple function blocks with nearest-neighbour interconnect

Challenges ahead

The biggest challenge currently is:

- finding examples where soft circuitry is faster than a program

Integration of configurable logic with memories and processors is the saviour on the horizon.

Assuming that the first challenge is cracked, then interesting issues are:

- finding the best high-level way to describe soft circuitry
- making soft circuitry into an integral system component
- finding efficient portable representations of soft circuitry
- talking about circuit/program co-design as well as hardware/software co-design