Dynamic Specialisation of XC6200 FPGAs by Partial Evaluation

Nicholas McKay
Tom Melham
Kong Woei Susanto
Dept. Computing Science
The University of Glasgow, U.K.

Satnam Singh
Xilinx Inc.
San Jose, California, U.S.A.

Abstract

We describe preliminary results of dynamically specialising Xilinx XC6200 FPGA circuits using the partial evaluation method. This method provides a systematic way to manage the complexity of dynamic reconfiguration in the special case where a general circuit is specialised with respect to a slowly changing input. We describe how we address the verification and run-time support issues which are raised when one modifies a circuit at run-time.

1 Introduction

Imagine a decryption circuit with two inputs: the key and the data to be decrypted. The key (a few bytes) changes infrequently with respect to the data (megabytes). Imagine at run-time being able to specialise this circuit every time the key changes to decrypt only for the given key. This would incur a run-time cost, i.e. the calculation needed to specialise the circuit description and then reconfigure the device. But in return it computes a circuit with a shorter critical path, allowing data to be decrypted faster. This paper describes a project which is developing technology to achieve exactly this kind of fine grain dynamic circuit specialisation.

Rather than solving the general problem of how to perform dynamic synthesis, we have selected a special case of dynamic reconfiguration which is easier to solve. In particular, we are researching how to dynamically specialise circuits systematically by taking a general circuit and some data known at run-time and then using this to *transform* the general circuit into a specialised circuit. By trying to solve this simpler problem, which has useful structure and properties, we hope to get insight into how to solve more general problems in the area of dynamic reconfiguration.

Instead of devising a totally new methodology for dynamic circuit specialisation, we have borrowed existing ideas in the areas of off-line constant propagation from HDL compiler technology and from partial evaluation [1] techniques developed for the run-time specialisation of software.

While we have not done any work which is specifically aimed at software radio, we do hope that this research may have some applications in this field.

2 Dynamic Hardware Synthesis

The project is based around the technology of Field Programmable Gate Arrays (FPGAs). These devices consist of an array of cells that can implement a variety of logic functions plus some interconnection network. Configuration information is downloaded onto the FPGA to set up the cells and interconnection network to realise a specific circuit.

The project is using the XC6200 family of FPGAs. These combine very fast reprogramming speeds, the ability to perform partial re-programming, and high performance. These features are exploited by our research.

The XC6200 chips can have their configuration state mapped onto the address space of the host system, so that reconfiguration under software control is as simple as assigning to variables in a program. This allows the dynamic reprogramming of subsections of the FPGA, even while the remainder of the chip is running. Circuits may be swapped into and out of the FPGA at will and at high speed. An analogy with virtual memory is appealing, and we call this technique *virtual hardware*.

However in addition to swapping in static, pre-compiled circuits, we also wish to synthesise circuits *dynamically*, on a need to use basis, before downloading them to the FPGA at run-time. For example, consider the above example of a device designed to decrypt a data stream with a given key. For each new session key, a specialised circuit can be dynamically synthesised which decrypts the associated stream with the relevant key. This circuit will be smaller and faster than a general circuit which stores the key in a register.

In this approach, the important question is when the cost of calculating new configurations are amortised over sufficient time to make the approach worthwhile. Consider a data stream consisting of a specialisation parameter followed by n data items.

In the encryption example mentioned above the specialisation parameter would be the key and the data items the message. Now suppose:

Ts	Time to synthesise hardware
Tp	FPGA programming time
Tc	Cycle time for specialised hardware
Tg	Cycle time of general purpose device

Tk Time to load specialisation parameter

We are concerned with the ratio:

$$\frac{dynamic}{conventional} = \frac{(T_s + T_p) + nT_c}{T_k + nT_g}$$

One aim of this research will be to identify applications in which n is sufficiently large and Tc/Tg sufficiently small to make our approach worthwhile. For example, a circuit realising the DES ('Data Encryption Standard') [2] algorithm has the very useful property that under specialisation the combinatorial logic associated with the generation of the key schedule and transposition stages can be converted to a sequence of inverters, which themselves can then be absorbed into a modified S-box. In this case 768 gates will have been replaced by wires and (in a design without pipelining) 16 gates delays removed from the critical path; Tc will be substantially smaller than Tg.

3 Verification

While dynamic synthesis seems appealing, it represents a major verification problem. How do we know if the dynamically generated circuit works as intended?

Even when current synthesis techniques are used, conventional verification relies on simulation—taking hours or days and often requiring human input and checking. Clearly such an approach is impossible in a system that will dynamically generate hardware and then use it for just a few milliseconds before discarding it.

An alternative is to use formal methods to verify the synthesised circuit. Here, again, user-guided methods are obviously inappropriate. Automatic techniques, for example those based on model checkers, can verify small to medium sized circuits; but even these still take far too long to execute for 'in the field' verification of synthesised circuits. In general, it is not feasible to perform a post-design verification of dynamically generated hardware, where 'post-design time' means a tiny gap between synthesis and downloading onto the FPGA.

We have chosen to verify the synthesis algorithm itself. The above decryption example provides an illustration of how we proceed. First, we use conventional techniques, complemented by formal methods, to convince ourselves that a general circuit which stores the key in a register works as intended. We can then use formal techniques to prove that the specialised circuit synthesised from any given key is a legitimate replacement for the general circuit loaded with that key. This is a correctness property of the synthesis algorithm, which we need prove only once for all possible input keys.

4 Results to Date

In the first stage of this project we have been implementing a very simple form of partial evaluation that corresponds to run-time constant propagation. This is analogous to a technique used in functional programming [1], by which a general design can be specialised in the presence of partly known inputs. By propagating known values at run-time, we can transform cells implementing logic functions like AND and OR into cells that just route wires, avoiding the delay incurred by going through the interior of the function block.

Using this technique we have partially evaluated several circuits, including adders, multipliers and FIR filters. The speed-up results for an 8-bit by 8-bit multiplier are shown in Table 1.

	times 1	**times 2**	**times 8**	**times 85**
p.e.	23.5 MHz	23.5 MHz	23 MHz	16 MHz
non-p.e.	14 MHz	16.5 MHz	16 MHz	15 MHz
	times 126	**times 128**	**times 170**	**times 255**
p.e.	13.5 MHz	20 MHz	16 MHz	11 MHz
non-p.e.	11 MHz	14.5 MHz	13.5 MHz	11 MHz

Table 1 Speed comparisons of a partially evaluated and a non-partially evaluated 8-bit by 8-bit parallel multiplier

Our verification methodology is based on theorem proving using the PVS theorem proving tool. We have formalized and verified all possible cell specialization transformations. Currently, we are at the stage of formalizing the partial evaluation algorithm. This will give us confidence in the correctness of the transformations that are applied to the general circuit. We will also apply formal proofs and simulation techniques to the general circuits and show that the specialisations preserve their functional behaviour.

5 Acknowledgements

This work is part of a project funded by EPSRC (project reference GR/L38530) and the United Kingdom Ministry of Defence (MoD), managed by Satnam Singh (Xilinx Inc.), Tom Melham (University of Glasgow) and Derek McAuley (Microsoft Research Labs, Cambridge).

6 References

[1] N. D. Jones, C. K. Gomard, and P. Sestoft, *Partial Evaluation and Automatic Program Generation*, Prentice-Hall, 1993.

[2] National Bureau of Standards. *Data Encryption Standard* (DES), Technical Report. April 1997.