Using Reconfigurable Hardware to Speed up Product Development and Performance

Adrian Lawrence[1], Andrew Kay[2], Wayne Luk[1], Toshio Nomura[2], Ian Page[1]

[1] Oxford Parallel, Oxford University Computing Laboratory, Oxford OX1 3QD, UK
[2] Sharp Laboratories of Europe Ltd., Edmund Halley Road, Oxford OX4 4GA, UK

Abstract. Harp1 is a circuit board designed to exploit the rigorous compilation of parallel algorithms directly into hardware. It includes a transputer closely-coupled to a Field-Programmable Gate Array (FPGA). The whole system can be regarded as an instance of a process in the theory of Communicating Sequential Processes (CSP). The major elements themselves can also be viewed in the same way: both the transputer and the FPGA can implement many parallel communicating sub-processes. The Harp1 design includes memory banks, a programmable frequency synthesizer and several communication ports. The latter supports the use of parallel arrays of Harp1 boards, as well as interfacing to external hardware. Harp1 is the target of mathematical tools based upon the Ruby and occam languages, which enable unusual and novel applications to be produced and demonstrated correctly and rapidly; the aim is to produce high quality designs at low costs and with reduced development time.

1 Introduction

The performance of many computationally demanding tasks can be improved by the use of matching parallel hardware. But the design of correct circuits is notoriously difficult and time-consuming. Harp1 is a circuit board designed to demonstrate that mathematical methods can overcome some of these problems.

Figure 1 gives a broad overview of Harp1. The architecture of Harp1 is inspired by the theory of Communicating Sequential Processes (CSP) [1]. The board contains two communicating elements: a transputer and a Field-Programmable Gate Array (FPGA) chip. Both elements are programmable and are capable of implementing many parallel processes internally; in particular, the FPGA hardware may be configured with almost complete freedom for major parallelism when appropriate. A board implements a CSP process, and may be used alone, in arrays, or as a component in a diverse system.

Harp1 provides external memories for the FPGA and the transputer. Each is also provided with external communication resources. The FPGA is normally clocked by a programmable frequency synthesizer.

Harp1 may be exploited using conventional methods, but it becomes far more powerful when coupled with mathematically based compilation tools. Both a variant of occam(TM) [3], Ruby [2] and a combination can be compiled directly

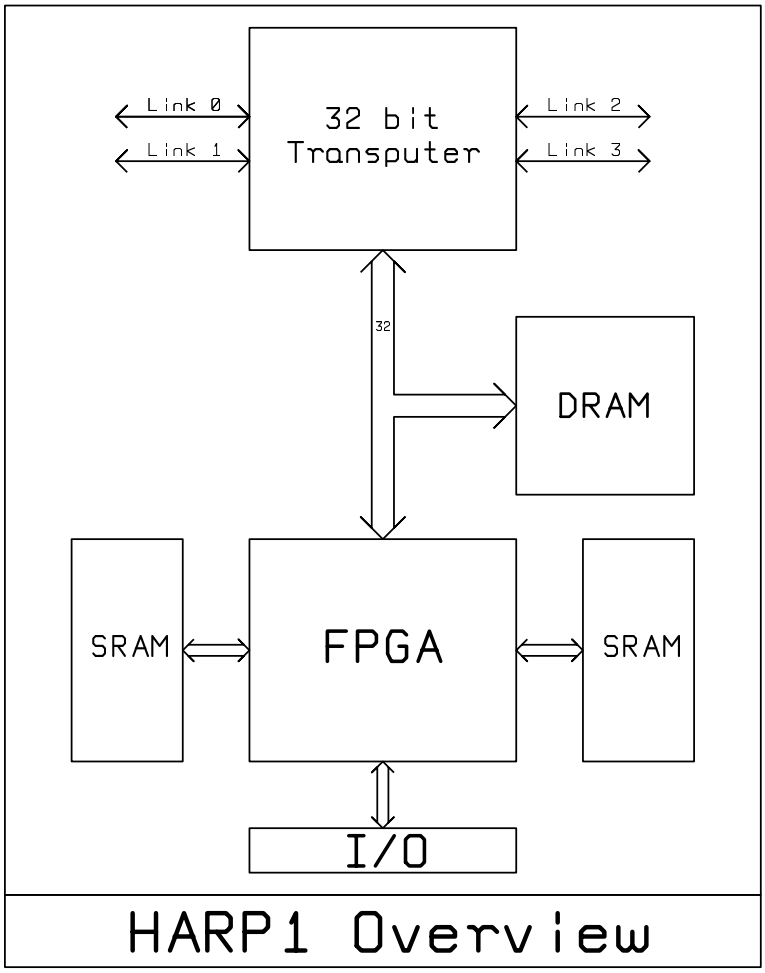

Fig. 1. Overview of Harp1

into the FPGA hardware. This allows correct hardware to be produced from high level descriptions very rapidly. Systolic arrays, for example, can be efficiently implemented on the FPGA.

The flexibility of Harp1 makes it useful in a variety of situations. It is a suitable development platform for FPGA designs or for prototyping application-specific integrated circuits. It may be used to accelerate a program: the speed-critical parts may be mapped onto the FPGA while the rest of the program runs on the transputer; different cost-performance trade-offs between hardware and

software can be explored. Another use of the FPGA is to hold custom hardware for algorithms that require non-standard data size, or operations that are not included in the instruction set of microprocessors.

Here we give a brief overview of the hardware: aspects of the compilation tools have been described elsewhere (see [5], [6], [7], [8], [10], [9], [11] and [12]).

2 Field Programmable Gate Arrays

Field Programmable Gate Arrays are now widely used in electronic design. They provide directly programmable hardware and promise to change the way in which design is approached in a revolution similar to that engendered by the microprocessor. Programmable hardware has of course been available for many years, not least various forms of memory. Various PLDs (Programmable Logic Devices) have been used for several decades in implementing state machines and "glue" logic, among other things. But the available devices have tended to have restricted architecture, and to be rather small.

The last decade has seen a significant change with the introduction of a variety of field-programmable gate arrays, as well as an evolution of some PLDs into much larger devices with extended architectures. The term FPGA is normally understood to apply to programmable devices with a regular array of cells with distributed routing. But the term is sometimes also applied to "super-PALs" which may also be called CPLDs (Complex PLDs): these usually have what is essentially a large cross-bar switch as the major interconnection mechanism.

Some FPGAs may only be programmed once: these are often based on anti-fuse[1] technology. But others are based on SRAM, and may be programmed or *configured*[2] repeatedly.

There are a variety of FPGA architectures available, and no one style has emerged thus far as obviously superior for all purposes. Some have a hierarchical sub-structure for the cells and routing. However, there is one broad distinction that can be made: the architectures are either *coarse-* or *fine-grained*. This refers to the capability of a single cell. A fine-grained FPGA tends to have a very large array of relatively simple cells. An example is the AT6000 series [21]. But the FPGA used on Harp1 is coarse-grained. It is usually a Xilinx XC3195 [17].

The FPGA on Harp1 consists of an array of cells called CLBs (**C**onfigurable **L**ogic **B**locks). Each CLB contains two latches, and a function generator. The internal connections within the cell and the lookup table in the function generators are determined by configuration bits held in an integrated SRAM. This allows an individual cell to implement quite complex combinational and sequential elements. The routing resources allow the cells to be connected as required, at least in principle. In practice, the problem of routing a congested design is the major obstacle in obtaining the highest performance. The particular intercon-

[1] An *anti-fuse* makes a connection when it is "blown".

[2] Configuration here means the programming of the FPGA functions and interconnection.

nections required in a design are determined by further configuration bits held in the same integrated SRAM.

This ability to sculpt hardware to fit a computation is significantly different from the programmability of a microprocessor. In particular, the microprocessor has a fixed instruction set[3]. In contrast, an FPGA can be configured to fit a particular algorithm. We may even combine the two approaches, and compile a specialised microprocessor into the FPGA with a restricted instruction set chosen to suit the particular case. Often a computation falls naturally into two parts, one of which matches a microprocessor implementation, while the other can use special hardware. Harp1 provides just the right closely coupled environment for that situation.

Precisely because an FPGA is programmable in something like the same sense as a microprocessor, it is already becoming very widely used. Eventually it should become a commodity item, although prices are still very high. There is often a performance penalty: it is seldom possible to obtain the ultimate speed from a general purpose architecture as from a custom design. This is especially true of routing: a large number of programmable interconnections usually significantly slow signals. The anti-fuse technology has an advantage here, but is not reprogrammable.

The type of reconfigurable SRAM-based FPGA used in Harp1 provides an ideal testbed in which to explore compilation of algorithms directly into hardware. The methods apply to the design of any hardware, but the flexibility and reusability of the FPGA are exactly what is needed for a prototyping environment.

Although Harp1 was designed as a testbed for compilation techniques, it is useful in its own right. The performance of the FPGA is respectable, and is very adequate for many purposes.

3 Architecture

Harp1 is a size 6 TRAM [16] circuit board. A 32-bit transputer, usually a T805, controls the system, and is the relatively conventional processor. The processor has 4Mbytes of external DRAM memory which is also accessible to the FPGA via DMA. The transputer is responsible for configuring the FPGA, and controlling the FPGA clock. The four links are available for communication with other devices in the standard way.

Two banks of fast SRAM are associated with the FPGA. Communication resources beyond the 4 transputer links are provided: a connector provides a further 17 fully programmable signals. This permits high bandwidth interaction with external devices. It also provides a configurable port for connection to other hardware, particularly useful when prototyping applications that may need to control special hardware. Figure 2 gives a simplified description of the architecture of Harp1.

[3] Or micro instruction set.

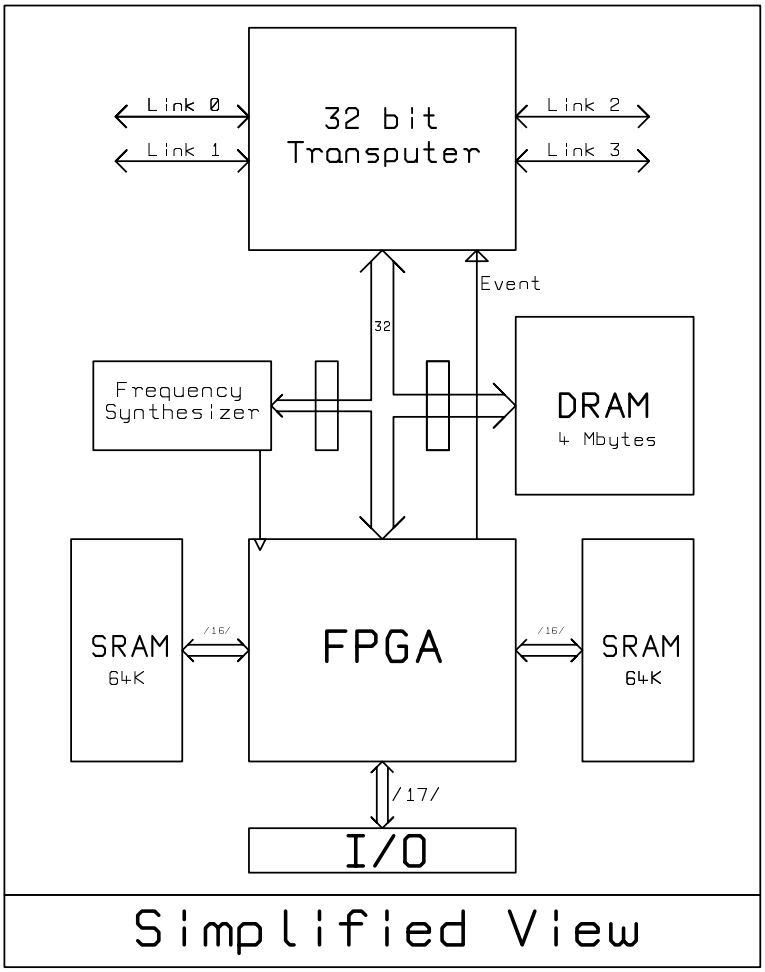

Fig. 2. Simplified view of Harp1

The configuration of the FPGA is carried out by the transputer whenever necessary. It is also possible to read back the loaded configuration when permitted: a security feature can be invoked to disable this option.

An important measure of the performance of an FPGA configuration is the maximum clock frequency. If a design has many layers of deep combinational logic, the propagation time, especially that involving inter-cell routing, is often the bottleneck that constrains maximum performance. Thus Harp1 has a high performance and very flexible clocking circuit, including a frequency sythesizer.

The FPGA can be clocked from various sources, including a single-step circuit. When the frequency synthesizer is selected, a very large range of frequencies with fine resolution is available. The range is normally of the order of 1 to 100MHz, with a resolution of better than 125kHz. This allows us to benchmark configurations, permitting investigation of compiler strategies, routing and placement algorithms and related matters. The frequency synthesizer is normally controlled by the transputer, and uses a Phase Locked Loop.

The FPGA may be used as a co-processor, dynamically reprogrammed when needed by the transputer. Or it may have a single application-specific configuration loaded at boot time.

In most parallel systems communication is paramount. Harp1 meets this need by connecting the FPGA directly to the transputer bus, and also by allowing it to activate the transputer *Event* channel. Thus there is a high bandwidth highway between the processors, and the ability for either end to signal the other at any time. In fact, we can implement many CSP channels between multiple processes spread across the transputer and the FPGA in this way.

The boards may be used alone, or as part of parallel arrays. Although designed primarily to support the compilation of Handel (a variant of occam) and of Ruby directly into hardware, the FPGA may also be programmed by standard commercial tools.

4 Applications

A few prototype applications have been developed to run on the Harp1 board. In most cases the applications were intended to exercise the compilation technology rather than the hardware itself. For this reason we have concentrated on functional complexity rather than speed or usefulness. The list is not complete.

The simplest application used the FPGA as a pattern matcher for a fixed string (with wildcard matches). It returns the set of addresses in RAM at which the pattern can be found.

A spelling checker was developed which checked the existence of words in a compressed, miniature dictionary (of 6000 English words) stored in SRAM. Here the FPGA is used as a "spelling coprocessor", looking up whatever word the transputer requires. An extension to the transputer part of the system allows close matches to be found for incorrectly spelt words.

By extending the data structures and improving the search strategy of the spelling checker, a morphological analyser was developed. This application is capable of splitting words into their constituent parts, and could be used as part of a natural language parsing system. For example, the word "uneatable" might be represented by three tokens denoting "un-(prefix)", "eat(verb)", "+able(suffix)".

An OCR kernel (optical character recognition) was implemented using the flexibility of the FPGA to provide the unusual binary operations used in the specific algorithm. This allows rapid searching to be done even at low clock speeds. The task of the kernel is to take a normalised input and compute a

similarity measure with each entry in the database of characters (in this case, Japanese Kanji characters), keeping the best few for further analysis.

A merge sorter and a heap sorter were implemented to compare their relative efficiencies in space and time. The merge sorter works well, performing close to one memory access per clock cycle and achieving close to the theoretical minimum number of cycles for this algorithm. Sadly, the complexity of the heap sorter meant that it would not fit on the FPGA. This problem could be cured by improving the optimisation methods used by the compiler, by using a larger FPGA on Harp1 or by using an array of Harp1s.

Other applications of Harp1 include real-time digital filtering [22] and video motion tracking [23].

5 Summary

Because Harp1 is almost totally programmable, it is useful in a range of situations. The applications above demonstrate its utility as a development platform; it also has a place in education and familiarization with FPGA technology. The I/O connector means that it is easily used as a prototype for custom designs. It provides a powerful computational resource, particularly for algorithms which do not "fit" a microprocessor architecture. Systolic arrays, for example, can be efficiently implemented on an FPGA. More routine applications are as a "custom" interface to unusual hardware, or where a single piece of equipment needs to be reprogrammed to suit different hardware standards. By using the frequency synthesizer and programming the FPGA to pass the signals through to the external connector, Harp1 could even be used as a (exotic) signal generator.

Acknowledgements

Thanks to Bob McLatchie for suggesting the title of this paper. The support of a JFIT grant (IED/46/91/014) is gratefully acknowledged.

References

1. C A R Hoare. *Communicating Sequential Processes.* Prentice-Hall International, 1985.
2. G Jones and M Sheeran. *Circuit Design In Ruby.* In J Staunstrup (ed.), *Formal methods for VLSI design,* North–Holland, 1990.
3. *occam 2 Reference Manual.* Prentice-Hall International, 1988.
4. W R Moore and W Luk, Eds. *More FPGAs.* Abingdon EE&CS Books, 1994.
5. W R Moore and W Luk, Eds. *FPGAs.* Abingdon EE&CS Books, 1991.
6. I Page and W Luk. *Compiling occam into FPGAs,* In [5].
7. W Luk and I Page. *Parameterising Designs for FPGAs.* In [5].
8. I Page, W Luk and H Lau, *Hardware Compilation for FPGAs: Imperative and Declarative Approaches for a Robotics Interface. Proc. IEE Colloquium on Field Programmable Gate Arrays – Technology and Applications*, Ref. 1993/037, pp. 9.1-9.4, IEE, February 1993.

9. J He, I Page and J P Bowen. *Towards a Provably Correct Hardware Implementation of Occam.* In G J Milne and L Pierre (eds.), *Correct Hardware Design and Verification Methods*, Lecture Notes in Computer Science, 683, pp. 214–225, Springer-Verlag. 1993.
10. I Page. *Parametrised Processor Generation.* In [4].
11. W Luk, D Ferguson and I Page. *Structured Compilation of Parallel Programs into Hardware.* In [4].
12. W Luk and T Wu. *Towards a Declarative Framework for Hardware-Software Codesign.* In *Proc. Third International Workshop on Hardware/Software Codesign*, pp. 181–188, IEEE Computer Society Press, 1994.
13. A E Lawrence. *HARP1 User Manual.*
14. A E Lawrence. *HARP1 (TRAMple) manual, volume 2: Manufacturing Pack.* 1992. Confidential.
15. A E Lawrence. *Analysis of a Phase Locked Loop: Expanded issue for HARP1 documentation.* 1991. Confidential.
16. *Dual-in-Line Transputer Modules (TRAMs).* Inmos Technical Note 29, *Transputer Development and iQ Systems Databook.* 2nd Edition, 1991.
17. *The Programmable Logic Data Book.* Xilinx Inc., 1993.
18. *The Transputer Databook.* Inmos document 72 TRN 203 02, Third edition, Inmos, 1992.
19. *The T9000 Transputer Hardware Reference Manual.* Inmos document 72 TRN 238 01, First edition, Inmos, 1993.
20. M S Jhitta. *Introduction of a New FPGA Architecture.* In [4].
21. *Configurable Logic Design and Application Book.* Atmel Ltd., 1993.
22. M Aubury and W Luk. *Binomial Filters.* To appear in *Journal of VLSI Signal Processing.*
23. M Bowen. *Tracking Moving Objects in Video Images.* Project Report, Oxford University Computing Laboratory, 1994.

This article was processed using the LaTeX macro package with LLNCS style