Riley-2: A Flexible Platform for Codesign and Dynamic Reconfigurable Computing Research

Patrick I Mackinlay[1], Peter Y.K.Cheung[1], Wayne Luk[2], Richard Sandiford[2]

Imperial College of Science, Technology and Medicine,
Exhibition Road, London SW7 2BT.

Abstract: The paper first proposes requirements for an ideal platform for codesign research. A new board developed at Imperial College, the Riley-2, is shown to meet these requirements. It is a PCI based board consisting mainly of four dynamically reconfigurable FPGAs and an embedded processor. A VHDL model of the main Riley-2 system, including all components except the PCI interface, is described. Two design routes, one based on VHDL with parametrised hardware libraries and the other based on a novel codesign language called Cedar, have been developed for Riley-2. Finally, an image processing application running on a PC with the Riley-2 and a Quickcam camera, is described.

1. Introduction

Electronics systems containing application specific hardware working alongside an embedded microprocessor are now common place. However, to design such systems quickly and reliably, while exploring the design space sufficiently to ensure near optimal solutions presents many new challenges. The problem is further complicated by reconfigurable hardware such as FPGAs, some of which can be dynamically reconfigured. This paper describes the design of an experimental platform, known as Riley-2, that supports our research in codesign and reconfigurable computing at Imperial College. It will also describes the tools that we have been developing to assist the design and analysis of such systems. We hope that the collection of tools would form a framework that facilitates rapid design exploration, evaluation and validation.

[1] Department of Electrical and Electronic Engineering,. (p.cheung@ic.ac.uk, p.i.mackinlay@ic.ac.uk)
[2] Department of Computing. (w.luk@ic.ac.uk, r.sandiford@ic.ac.uk)

2. Platform requirements

An ideal platform for codesign and reconfigurable computing would allow us to investigate both embedded and PC acceleration applications. Such a platform would comprise the following:

1. A general purpose processor (possibly used in embedded systems), with a modest amount of memory.
2. An adequate amount of reconfigurable resources, comprising a number of FPGAs with local memory. This would allow investigation of partitioning across FPGAs.
3. A flexible interface to the reconfigurable resources, allowing fast, partial and runtime reconfiguration of the reconfigurable resources.
4. A flexible and extensible external IO interface.
5. A fast host interface.
6. A complete model of the system, which allows detailed examination of the interaction between the software and the dynamically changing hardware.

There are many FPGA-based computing machines which could be used in codesign research, however most fall short of the above requirements. For example the Riley board (Riley-2's predecessor) [1] only has one FPGA, has a slow interface to its host and does not support partial configuration. A more recent PCI-based XC6216 board, whose architecure is outlined in a previous paper [2], only has a single FPGA and no processor.

3. Riley-2 overview

Riley-2 is a successor to an earlier system, known as Riley [1], designed by Hewlett Packard laboratories, Bristol. It is designed to be used in a Pentium PC with a PCI interface. A photograph of Riley-2 can be seen in Figure 1.

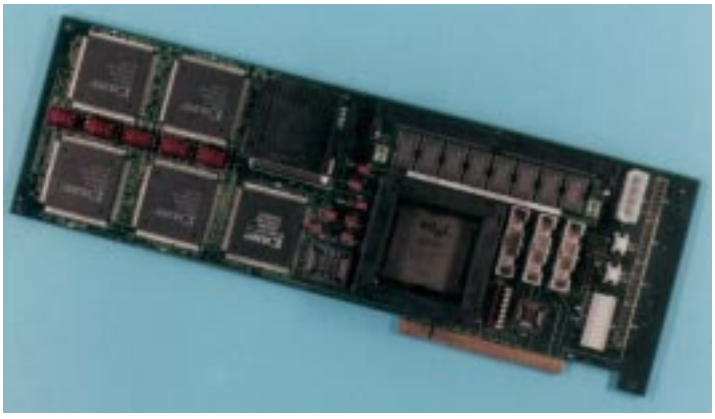

Fig. 1. Riley-2 board

Figure 2 shows a block diagram of the Riley-2. The design has two main buses, the microprocessor's local bus and the reconfigurable resource (RR) bus.

The local bus connects the components meeting requirement 1 of the above list. The i960JF core is a 33 MHz integer RISC core used in many embedded system designs. The shared memory is implemented with a single 72 pin SIMM, currently seating a 16 Mbyte 50 ns BEDO DRAM, allowing a burst throughput of 88 Mbytes/sec. The boot ROM, not shown in the diagram, is a 256 Kbyte FLASH ROM, which contains the boot-up sequence and initialises the PCI interface.

In accordance with requirement 2, the RR bus connects four reconfigurable resource units, each unit containing a Xilinx 6216 FPGA and a 512 Kbyte fast local memory with a 32 bit data bus.

The RR interface along with the 6216s [3] provides the i960JF core with a flexible interface to the reconfigurable resources (requirement 3). The 6216 has the following features:

- Advance microprocessor interface with direct read/write access to the reconfigurable resources. All registers (6216 internal and user defined), SRAM control store memory and local SRAM are mapped onto the microprocessor address space.

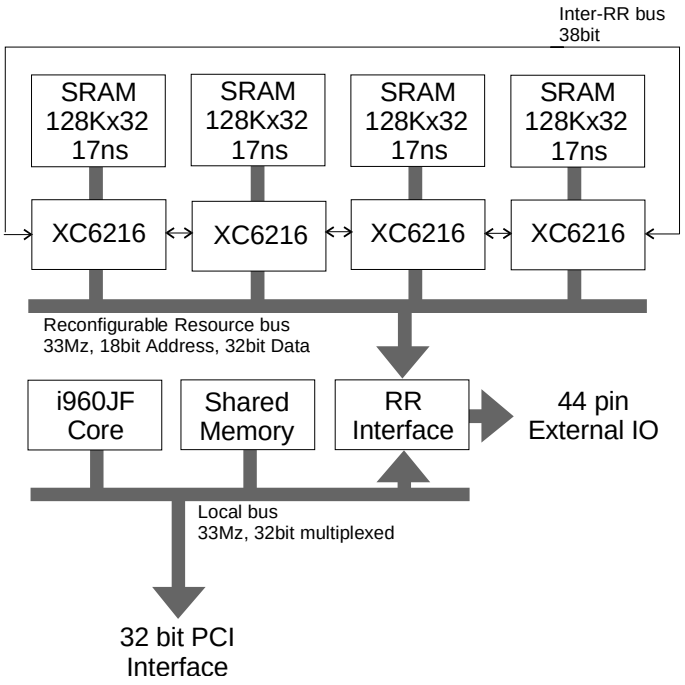

Fig. 2. Riley-2 block diagram.

- Advanced dynamic reconfiguration capability with a high speed CPU interface, unlimited and partial reprogrammability.

The implementation of the PCI interface gives the host programmer flexibility by mapping the i960JF's shared memory and the reconfigurable resources into the host processors address space. The processor, Intel's i960RP [4], contains a 32 bit PCI interface in the same chip as the i960JF core. It has inbound and outbound transfer queues which can support sustained burst transfers of up to 132 MBytes/sec (requirement 5). Actual speeds will depend on the host's PCI clock and on its chipset.

The RR interface also has a 44 pin external IO connector, which provides access to any external device, such as a video source. Since the RR interface is implemented with an FPGA, the IO interface can be changed and is very flexible (requirement 4).

4. Cosimulation

One important aspect of a codesign environment is the possibility of cosimulation, as outlined in requirement 6. Both hardware and software can be simulated together with proper interface between them. To facilitate cosimulation of the Riley-2 System, we have developed a functional simulation model in VHDL for the i960 microprocessor. The model handles i960 machine code level instructions produced by the gnu *gcc960* compiler. In addition, we have also developed a VHDL model for the rest of the Riley-2 system (except for the PCI interface), including models for all the memory modules and the 6216s. A commercially available VHDL simulator is used for cosimulation.

The i960JF core model from the Riley system [1] was modified for the Riley-2. This was a relatively simple task since the processor core is the same. Due to the complexity of the microprocessor core, the following compromises were made. Only the functional behaviour of the instruction is modelled. Not all instructions and processor modes are implemented, however these can easily be added if needed. Pipeline timing and architecture is only approximate, since Intel does not publish the detailed internal architecture of its chips. The local bus model does not take account of setup and hold times. However, the following functionality was modelled correctly: the local bus protocol, the internal data and instruction caches, the register file and scoreboarding hardware, and the on-chip memory. Hence, the model only gives an approximate cycle-by-cycle model, but the bus traffic is modelled correctly.

The RR interface is actually an XC4013E whose functionality is synthesised from VHDL. The model is the actual code used to create the 4013E configuration.

The following 6216 functionality is modelled: Microprocessor interface; State accesses; Map register; Cell hierarchy; Cell functionality. The 6216 model can be run on its own with a given 6216 configuration (CAL file), or the entire Riley-2 system can be simulated, with i960 code parsing a CAL file. Wildcarding has not yet been fully tested, since the current tools do not create any CAL files that use this feature. However, the VHDL code has all the necessary features to support wildcarding. The

model allows us to look in detail at the bus traffic during execution of a program. The exact details of configuring and executing FPGA designs can also be easily extracted. The detail in the 6216 model allows us to simulates FPGA designs that change dynamically. However, because of the complexity of the model, simulation is very slow and memory hungry. In most cases it is only practical to simulate part of the 6216.

5. Design Tools

In developing design tools for Riley-2, we aim to meet several challenges, including:

1. Efficient utilisation of FPGA resources;
2. Systematic description and compilation of hardware, software and host programs;
3. Versatile facilities for simulating system operation at different stages of the development process;
4. Appropriate strategies for partitioning data for local and global memories, and for partitioning hardware and software;
5. Performance enhancement by reconfiguring FPGAs at run time.

The following provides an overview of how the first three challenges can be met using parametrised libraries and the Cedar system. Some work on meeting the remaining challenges, such as design tools for exploiting run-time reconfigurability, can be found elsewhere [5].

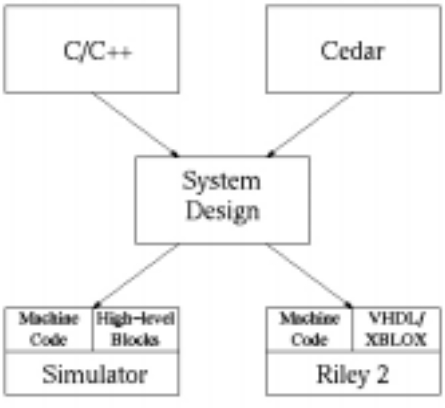

Fig. 3. High level compilation route

5.1 Overview

The first challenge is to enable efficient utilisation of FPGA resources. Our approach allows the user to optionally guide placement in a high-level language; for instance we have employed user-defined attributes in VHDL for this purpose. This development route is supported by a package of commonly-used library components

that have been optimised for the 6200 series [6]. They can be customised by parameters such as the size of input and output operands, the separation between the bits of an operand and the number of pipeline stages. In particular, the ability to specify the separation between bits allows the interfaces of connected components to be aligned. This alignment leads to a more predictable placement and reduces the amount of routing resources required.

The second challenge is to systematise description and compilation of hardware, software and host programs. Our approach supports a style of hardware/software codesign, in which both the hardware and software parts of an application are defined in a single source file. This infrastructure is built on top of the VHDL route explained above, and there is a simple but effective mapping from Cedar to the VHDL libraries or to Xilinx XBLOX components (Figure 3).

The software components running on the I960 and the PC host are usually written in C or C++, and the hardware components are described in a language called Cedar. Cedar is loosely based on C, but has extensions to deal with parallelism and communication; its semantics clearly defines the cycle-by-cycle behaviour of a design and is similar to that of Handel [7]. Cedar achieves portability by using a target-specific interface, itself written in Cedar, to provide a standardised set of channels with which to communicate with the processor and other components of the board. At the most abstract level, therefore, a design can be specified in a completely imperative manner using C and Cedar. This method treats the software and hardware as parallel processes of equal status that communicate using a message passing scheme.

Our third challenge is to provide facilities for simulating system operation at different stages of the development process. At the early stages, a design for Riley-2 involving C and Cedar can be tested using a high-level simulator that we have developed. If necessary, sections of a design may be optimised by redefining them using register transfer level Cedar or by the direct use of Riley-specific features such as Well Known Address accesses. Further refinement is possible via the creation of custom VHDL libraries, which can then be included into the main Cedar design. When all hardware descriptions have been compiled into VHDL, their behaviour can be obtained using commercial VHDL simulators. After FPGA programming files have been generated, detailed interactions of the Riley-2 system can be explored using the VHDL model of Riley-2 outlined in the previous section.

Further details of our tools will be given in the next section, when we consider a simple example.

5.2 Example

We illustrate the design route with an example which performs a threshold transformation on an image. In this example, the threshold value is specified at compile time and the hardware process stores just one pixel. Two fragments of the codesign file are shown in Figure 4 and Figure 5, which contain Cedar and C code respectively. The Cedar part enters an infinite loop in which a pixel is read from the `original_channel' channel and the processed value is returned via the `processed_channel'. The code is encased in a `using' directive which tells the compiler to use the interface called `riley6216channels' to provide these channels. The software in Figure 5 uses the matching communication primitives to send each pixel of an image to the hardware and reads back the processed result.

```
typedef unsigned int pixel : 8;
using riley6216channels
{
  pixel original_pixel;
  while (true)
  {
    input  (processor.out [original_channel],
            original_pixel);
    output (processor.in [processed_channel],
            original_pixel<threshold?
black_pixel:white_pixel);
  }
}
```

Fig. 4. Simple Cedar threshold program

```
for (y=0; y<height; y+=1)
{
  for (x=0; x<width; x+=1)
  {
    output (threshold_fpga.in  [ original_channel],
picture[y][x]);
    input  (threshold_fpga.out [processed_channel],
newpicture[y][x]);
  }
}
```

Fig. 5. Inner loop of software program

The implementation of Cedar is based on the one-hot encoding method, which relies on a small set of token-passing control components [7]: the presence of a token in a circuit indicates that the corresponding statement is being executed. A library-based strategy is adopted in the design of the Cedar compiler. From the source code, an abstract syntax tree is generated which is then transformed into modules

independent of the chosen implementation technology [8]; these modules provide the interface to the back-ends which target technology-specific libraries.

The Cedar compiler currently possesses two back-ends: one converts the high-level modules into Xilinx XBLOX format, while the other translates it into VHDL that uses the 6200 libraries. The language provides a mechanism to include designs written in other languages into the output of these back-ends. Such designs are treated by Cedar as external functions that can be called in the same way as true Cedar functions. These functions can be bound, at the user's request, to built-in operators such as addition and multiplication. This external function facility is especially flexible, since it supports polymorphic arguments and return values, multi-cycle functions and, if necessary, allows the function to be managed by a dedicated Cedar process. For example, the language can describe interfaces to multiplier libraries that are pipelined or combinational, that operate on signed or unsigned numbers, that are parametrised or of fixed size, and that require the two inputs to be the same size or allow different bit widths.

The other components of the high-level modules are mapped onto standard 6200 libraries when applicable, and onto Cedar-specific VHDL entities otherwise. In particular, control logic is mapped using a combination of gates and custom libraries such as the IOPRIM block, which implements the (symmetrical) input and output message passing primitives. Similarly, parallel statements are implemented by the PAR block, which collects tokens from parallel processes and releases a token when they have all finished.

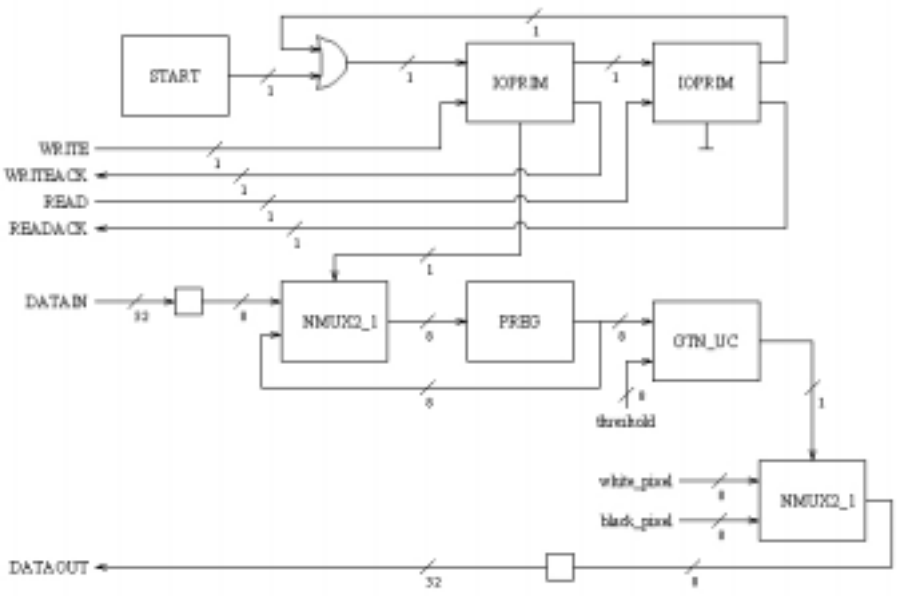

Fig. 6. Pictorial representation of VHDL circuit description produced from Figure 4.

The VHDL code for the threshold example is illustrated in schematic form in Figure 6. The riley6216 channel interface uses a wrapper that consists mainly of buffers and a small amount of bus control logic, and has the input and output ports shown on the left of the diagram. The control logic that was generated for the design is shown in the top half. The START block is used to generate a token after reset, while the two instances of IOPRIM are responsible for controlling the transfer of data, and the OR gate implements the loop. The data path is shown below the control logic; the PREG and leftmost NMUX2_1 together form the 'original_pixel' variable, while the other NMUX2_1 implements the condition `original_pixel<threshold ? black_pixel : white_pixel'. GTN_UC (Greater ThaN Unsigned Combinational) is the library that was chosen to implement the comparison operator.

Other applications, mainly for image and video processing, are currently being developed using the design tools described above. The Cedar compiler has also been used in producing designs for other FPGA systems, such as the EVC-1 board from Virtual Computer Corporation.

6. Applications

Support for Windows 95 has been developed in the form of a plug-and-play device driver and a simple monitor program. Using these, a demo program was written which performs some image processing form a video source. The application setup is shown in Figure 7. The design continuously grabs frames from the QuickCam, processes them on the Riley-2 and displays the result in a window.

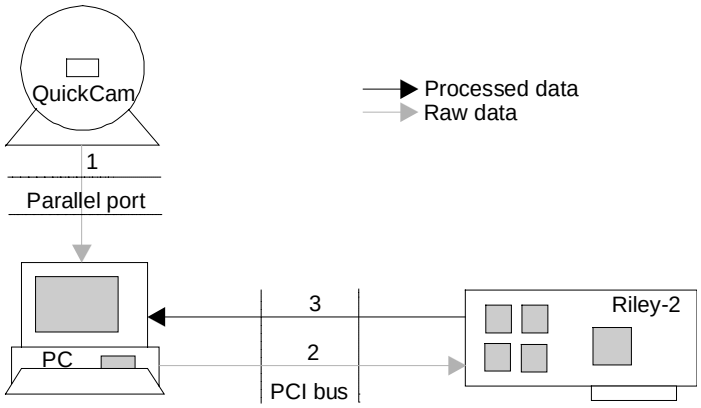

Fig. 2. Image processing application setup

The application can display either a normal picture (Steps 2 and 3 in Figure 7 are not performed), an inverted picture, a Gaussian filter or various edge detection pictures.

This is achieved by configuring three of the 6216s with these designs, a simple inverter, a filter/edge detector, a filter/edge detector followed by a thresholding

function. The latter two are based on a design outlined in a previous paper [2]. All four designs could have been fitted in a single 6216, but for test purposes three chips were used. Currently the frame size is only 320x240, 64 gray levels and the application achieves a very low frame rate. This is mostly due to the limitations of the QuickCam, since there is no visible difference in frame rate when the application is displaying a normal or a processed picture.

7. Conclusions

Many experimental platforms have been designed to support research in reconfigurable computing in the past. Riley-2 is perhaps unique in providing the necessary architecture to experiment with practical issues concerning dynamic partial reconfiguration and hardware/software codesign. For example, accessing the static memory on each 6216 from the PCI or local bus provides a number of interesting alternatives. One could configure each 6216 such that the SRAM device appears permanently on the memory map of the local bus. Alternatively, one could use the demand-driven model where the 6216 are dynamically configured to provide accesses to the SRAM when necessary. These two alteratives represent a possible trade-off between hardware resource and reconfiguration time. A number of demanding real-time image processing applications are currently being implemented on the Riley-2 in order to evaluate the strength and weakness of such a computing model, the efficiency of the cosimulation software, and the effectiveness of our design tools.

Acknowledgements

The support of Hewlett Packard Laboratories Bristol, the UK Engineering and Physical Sciences Research Council (research studentship and contracts GR/L24366 and GR/L54356) and the UK Overseas Research Student Award Scheme is gratefully acknowledged.

References

1. P.Y.K. Cheung, W. Luk, Patrick Mackinlay, “Hardware Cosynthesis for the Riley System”, IEE Colloquium Digest on Hardware-software cosynthesis for reconfigurable systems, Febuary 22, 1996.\
2. W. Luk, N. Shirazi and P.Y.K. Cheung, “Modelling and optimising run-time reconfigurable systems”, in Proc. IEEE Symposium on FPGAs for Custom Computing Machines, 1996.
3. The programmable logic data book published by XILINX.
4. i960RP Users Manual.
5. W. Luk, N. Shirazi and P.Y.K. Cheung, Compilation tools for run-time reconfigurable designs, in Proc. IEEE Symposium on Field-Programmable Custom Computing Machines, K.L. Pocek and J. Arnold (editors), IEEE Computer Society Press, 1997.
6. W. Luk, S. Guo, N. Shirazi and N. Zhuang, A framework for developing

parametrised FPGA libraries, in Field-Programmable Logic, Smart Applications, New Paradigms and Compilers, R.W. Hartenstein and M. Glesner (editors), LNCS 1142, Springer, 1996, pp. 24-33.

7. I. Page and W. Luk, Compiling occam into FPGAs, in FPGAs, W. Mooreand W. Luk (editors), Abingdon EE&CS Books, 1991, pp. 271-283.
8. G. Brown, W. Luk and J.W. O'Leary, Retargeting a hardware compiler using protocol converters, Formal Aspects of Computing, Vol. 8, 1996, pp. 209-237

1 P.Y.K. Cheung, W. Luk, Patrick Mackinlay, “Hardware Cosynthesis for the Riley System”, IEE Colloquium Digest on Hardware-software cosynthesis for reconfigurable systems, Febuary 22, 1996.

2 W. Luk, N. Shirazi and P.Y.K. Cheung, “Modelling and optimising run-time reconfigurable systems”, in Proc. IEEE Symposium on FPGAs for Custom Computing Machines, 1996.

3 The programmable logic data book published by XILINX.

4 i960RP Users Manual.

5 W. Luk, N. Shirazi and P.Y.K. Cheung, Compilation tools for run-time reconfigurable designs, in Proc. IEEE Symposium on Field-Programmable Custom Computing Machines, K.L. Pocek and J. Arnold (editors), IEEE Computer Society Press, 1997.

6 W. Luk, S. Guo, N. Shirazi and N. Zhuang, A framework for developing parametrised FPGA libraries, in Field-Programmable Logic, Smart Applications, New Paradigms and Compilers, R.W. Hartenstein and M. Glesner (editors), LNCS 1142, Springer, 1996, pp. 24-33.

7 I. Page and W. Luk, Compiling occam into FPGAs, in FPGAs, W. Mooreand W. Luk (editors), Abingdon EE&CS Books, 1991, pp. 271-283.

8 G. Brown, W. Luk and J.W. O’Leary, Retargeting a hardware compiler using protocol converters, Formal Aspects of Computing, Vol. 8, 1996, pp. 209-237