Field Programmable Gate Array Technology for Robotics Applications

P.H.W. Leong and K.H. Tsoi
{phwl,khtsoi}@cse.cuhk.edu.hk
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, NT Hong Kong

Abstract

Field programmable gate arrays (FPGAs) are integrated circuits which can be user customised to implement arbitrary digital functions. Modern FPGAs combine generalised logic resources with programmable interconnect, microprocessors, networking, multipliers, memories, delay/phase locked loops and other cores to provide a versatile system on a chip (SoC). In this review paper, we describe a typical FPGA architecture and discuss how they can be used advantageously in robotics. Compared with conventional technologies, FPGAs offer high speed, low power, reduced development time and cost. Examples are given including applications in logic replacement, rapid prototyping, control, planning and sensors.

1 Introduction

A field programmable gate array (FPGA) is an array of logic gates in which the connections can be configured by downloading a bitstream into its memory. They offer a compromise between microprocessor/DSP and application specific integrated circuit (ASIC) based systems, offering better performance than the former and shorter development times and lower cost than the latter. In this paper, we review the architecture of FPGA devices and describe some of the advantages that they offer for robotics subsystems including system integration, control, task planning, machine vision, neural networks and fuzzy logic.

As illustrated in Figure 1, an FPGA consists of a number of logic cells (LC) which can be interconnected to other logic cells and input/output cells (IOC) via programmable routing resources (RR). The LC consists of user-programmable combinatorial elements with an optional register at the output. Using such an architecture, subject to FPGA imposed limitations on the circuit's speed and density, an arbitrary circuit can be implemented. The actual design is described via a configuration bitstream which specifies the LC functionality and their interconnection.

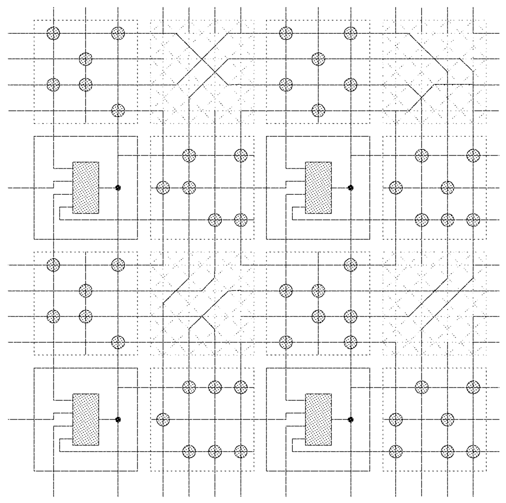

Figure 1. Architecture of an FPGA with four-input LUT cells (figure courtesy of M.P. Leong). The LUTs, shown as grey rectangles are connected to programmable routing resources (shown as wires, dots and diagonal switch boxes).

In a traditional FPGA design flow, a user enters a description of the desired circuit by using a hardware description language (HDL) such as VHDL or Verilog. A CAD tool is used to synthesize a netlist from the HDL description. Another CAD tool is then used to decompose the netlist to fit the logic resources of the FPGA, and then a place and route (P&R) tool is used. Following this, the logic and interconnection of the FPGA is defined, and the CAD tool outputs a bitstream which can be downloaded to the FPGA's configuration RAM.

Apart from the basic FPGA architecture just described, current trends are to incorporate commonly used features so that designers can integrate entire systems on a single FPGA device. Apart from cost and board area benefits, this also improves performance since higher transfer rates can be achieved if all components are on the same chip. A contemporary FPGA usually has the following features: carry chains to enable fast addition, wide decoders, tristate buffers etc; blocks of on-chip memory and multipliers; embedded microprocessors; programmable I/O standards in the IOC; delay locked loops (DLLs) and phase locked loops (PLL) for clock de-skewing, phase shifting and multiplication.

In addition to the architectural features described, intellectual property (IP) cores, implemented using the LC resources of the FPGA, are available from vendors which can be incorporated into a design. These include bus interfaces, networking components, memory interfaces, signal processing functions, microprocessors etc and can significantly reduce development time and effort.

2 FPGAs vs ASICs and Microprocessors

Microprocessors offer an easy to use, powerful and flexible implementation medium for digital systems. Their utility in embedded applications makes them an overwhelming first choice for robotics systems since they come in a wide variety of models which can cater to different cost and performance requirements. Moreeover, it is relatively easy to find software developers and they are widely supported by development systems, compilers, debuggers, libraries and operating systems. Unfortunately, their generality does not make them the best choice for a large class of applications which need to be optimised for performance, power or board space.

Application specific integrated circuits (ASICs) and FPGAs are able to arrange computations in a spatial rather than temporal fashion and greater levels of parallelism than a microprocessor can be achieved. Thus performance improvements of several orders of magnitude can be achieved. Also, the absence of caches and instruction decoding can result in the same amount of work being done with less chip area and lower power consumption [2].

Compared with ASICs, FPGAs offer very low non recurrent engineering (NRE) costs. This is often a more important factor than the fact that FPGAs have higher units costs and applications in robotics normally do not have the very high volumes required to make ASICs a cheaper proposition. As integrated circuit feature sizes continue to decrease, the NRE costs associated with ASICs continue to escalate, making the volume at which it becomes cheaper to use an ASIC much higher. FPGAs will be used in increasingly more applications, ASICs only being cost effective for the highest performance or highest volume applications.

Additional benefits of FPGAs are that its technology provides a shorter time to market than ASICs since the associated fabrication time is essentially zero, making many fabrication iterations within a single day possible. This allows more advanced algorithms to be deployed and makes possible problem-specific customisations of designs. A final consideration is that FPGA based designs are inherently less risky in terms of technical feasibility and cost since shorter design times and lower upfront costs are involved.

3 Applications

3.1 Logic Replacement

FPGAs were firstly used as logic replacement devices, having the benefit of being able to replace a number of small and medium scale integration devices. Another benefit comes from being able to reprogram the devices, so design changes and bug fixes can be made by changing the bitstream without modifying the printed circuit board. An example of this type of application might be to interface peripheral devices to a microprocessor system, the required address decoders, memory controllers, bus interfaces and motor controllers being implemented in a single FPGA. With modern devices, large amounts of logic can be incorporated on a single device, resulting in large savings in time to market, footprint and power consumption. Furthermore, board complexity is greatly reduced through tighter integration. FPGAs are also commonly used to prototype ASICs.

3.2 Reconfigurable Computing

Since FPGAs are general purpose logic devices, they can be used to develop high performance implementations of computational tasks. A direct implementation of an algorithm in hardware can achieve higher levels of parallelism than a microprocessor based design, and are often several orders of magnitude faster, use less area and have lower power consumption.

One other defining feature of FPGAs is the ability to reconfigure the device in the field or even at runtime. Field programmability allows hardware designs to be modified by downloading a bitstream, and the bitstream itself can be delivered via many different means including the internet or a telemetry system. Runtime reconfigurability is a feature which perhaps has not yet been exploited to its full potential and opens the way for customised hardware subsystems to be generated and downloaded to the FPGA only when needed. In the future, this may allow for systems which use the FPGA's logic in a manner similar to virtual memory and where portions of the design are downloaded to the device on a demand-based fashion. This would allow much smaller devices to be used, resulting in cost and power savings while at the same time freeing designers from logic limitations of the device.

4 FPGA Design Tools

One barrier that has affected the adoption of FPGAs has been the large amount of specialized knowledge required to use them. Designers needed expertise in digital logic design, floating point arithmetic was not available, devices were small so straightforward parallel implementations were not possible etc. Other issues that arise in hardware implementations include describing parallelism in a design and decisions regarding wordlength precision for fixed and floating point units. Furthermore, testing is usually more difficult since the FPGA normally forms a part of a larger system and interfaces may be difficult to observe and exercise.

Progress has been made in using C or C++ (e.g. [5]) as the input language for FPGA designs, an example of a commercial quality tool being Handel-C (www.celoxica.com). The use of traditional programming languages improves the productivity of experienced designers since low level details are handled by the compiler in a manner analogous to C versus assembly language for software development. Another difference with potentially large implication is that using these tools, software developers can also develop FPGA-based applications.

Many developers find that domain specific languages such as MATLAB/Simulink offer even greater improvements in productivity since it is interactive, includes a large library of primitive routines and toolkits and has good graphing capabilities. Indeed, many designs are first prototyped in MATLAB and then converted to other languages for implementation. Tools such as the MATCH compiler [3] and Xilinx System Generator can translate a subset of MATLAB/Simulink directly to an FPGA design.

The recent availability of Linux for microprocessors internal to an FPGA provides a familiar software development environment for programmers. This greatly facilitates program development through the availability of an enormous range of open source libraries as well as high quality development tools. Such tools can greatly speed up the development time and improve the quality of embedded systems.

To facilitate debugging, tools such as Xilinx's ChipScope offer on-FPGA logic and bus analyzer functionality. This enables a designer to view internal FPGA and processor signals at full operating speed.

5 Case Studies

FPGAs can be applied to almost any problem that can be solved using digital techniques. In this section, several examples in which FPGA technology was used successfully will be described.

5.1 Mars Lander Cameras

The Mars Pathfinder, Mars Surveyor '98 and Mars Surveyor '01 lander craft use several charge coupled device (CCD) based cameras for stereo and multispectral imaging. They are used to assist with navigation of surface rovers, confirm the manipulation of robotic arms and capture images for scientific investigations such as studies of surface morphology, minerology, atmospheric dust and water vapor. FPGAs are used extensively in these instruments for tasks including: decoding of commands from the spacecraft computer, control of image data storage and generating stepper motor drive pulses for a two axis gimbal which controls the cameras orientation in azimuth and elevation [6].

FPGAs have advantages for space applications since radiation qualified devices are available off-the-shelf, they have short design times and they can be reprogrammed after they are deployed. A major design challenge is to ensure correct operation in the presence of radiation which causes bit errors to occur with much greater frequency than within the Earth's atmosphere. In order to address this issue, periodic configuration bitstream readback and reconfiguration, cyclic redundancy checking (CRC) and logic redundancy are employed. All of these techniques are well matched with FPGA architectures.

5.2 Formula 1

Although not exactly a robot, a Formula 1 (F1) car has many similar characteristics. The 2003 BMW Williams car uses a vehicle control and monitoring (VCM) unit made from a Texas Instruments DSP chip and a Xilinx Virtex XCV600-E FPGA [1]. The VCM controls all aspects of the chassis except for the engine and its functions include:

- Control of the hydraulically actuated gear changes: including control of the clutch and gearbox actuators.
- Modelling of wheel slip to maximise traction and minimise tire wear.
- Control of the hydraulically actuated differential to optimise stability of the car when cornering.
- Launch control to maximise acceleration from a standing start.
- Logging and telemetry: approximately 220 channels of data are logged to a compact flash card for later analysis and a subset of this data is sent to the pit via a telemetry transmitter.

- Fault tolerance: the VCM uses redundant sensors so that it can continue to work correctly even in the presence of sensor failures. Decision logic is incorporated to override or ignore suspect inputs. The pit crew can also control whether or not a sensor is overridden via the telemetry link.

FPGA technology is beneficial for this application since the DSP is able to offload performance critical functions to the hardware of the FPGA; allows weight, space and power consumption to be saved through the reduction of parts; enables features to be easily added and removed as the rules of F1 are continually changing and finally, reduces the time required to incorporate new innovations, allowing the team to gain a competitive edge.

5.3 Autonomous Fuzzy Behaviour Control

Li et. al., describe a car-like robot which uses an FPGA to implement autonomous fuzzy behaviour control (AFBC), its control knowledge being derived from a small number of fuzzy rules [4]. The robot is capable of performing wall following, corner turning, garage parking and parallel parking manoeuvers. Fuzzy logic control and sensor-based behaviours are combined in the AFBC FPGA (an Altera EPF6024ACT144-3 chip which provides 24000 gates), its input being readings from six infrared sensors. The AFBC has two ouputs, a pulse width modulation (PWM) signal to a DC servomotor for control steering and another output for speed control.

As an example, for reverse parking, the driver passes the parking space by a short distance, turns the steering wheel, reverses the car into the space, the steering wheel being turned to straight as the car moves into the space, and the position is adjusted by going forward and backward until the car is in the middle of the space. This strategy is implemented directly in the fuzzy logic controller. The FPGA has submodules which convert the sensor data into fuzzy input status, a fuzzy inference module and a defuzzification module which performs a weighted average to determine a crisp outputs which are sent to the steering and speed motors. An autonomous robot of length 30 cm, width 24 cm and weight 5 kg was used to demonstrate the real-time ability of the AFBC to implement real-time driving and parking manoeuvers using a modest sized FPGA.

5.4 Autonomous Flying Object Navigated by Optical Flow

In an application which requires a high speed feedback control system, Yamada et. al. demonstrate a single FPGA system which can make a flying object (FO) hover in the air based on a high speed vision system [7]. Inputs are obtained via three CMOS horizontal optical sensors at 40 frames per second. Optical flow is computed in real-time in the FPGA and the attitude of the flying object is derived. An additional optical sensor mounted on top of the FO is used to determine a reference position based on two targets on the ceiling. Using these data, the roll, pitch, yaw and height of the object are controlled in an autonomous manner. The FO has a total weight of 400 g.

6 Conclusion

FPGAs are general purpose computing devices which can be used to reduce board size and accelerate applications in almost any domain. As they continue to improve in density, performance, cost and ease of usage, their applications in robotics can only increase.

Acknowledgement

The authors gratefully acknowledge support from a Chinese University of Hong Kong direct grant.

References

[1] L. Boland. Formula 1 racing: The Xilinx advantage. *Xcell*, 47(Fall):46–49, 2003[1].

[2] A. DeHon. The density advantage of configurable computing. *Computer*, 33(4):41–49, April 2000.

[3] M. Haldar, A. Nayak, A. Choudhary, and P. Banerjee. A system for synthesizing optimized FPGA hardware from MATLAB. In *ICCAD '01: Proceedings of the 2001 IEEE/ACM international conference on Computer-aided design*, pages 314–319, Piscataway, NJ, USA, 2001. IEEE Press.

[4] T.-H. S. Li, S.-J. Chang, and Y.-X. Chen. Implementation of human-like driving skills by autonomous fuzzy behavior control on an FPGA-based car-like mobile robot. *IEEE Transactions on Industrial Electronics*, 50(5):869–880, Oct 2003.

[5] I. Page. Constructing hardware-software systems from a single description. *Journal of VLSI Signal Processing*, 12(1):87–107, 1996.

[6] R. Reynolds, P. Smith, L. Bell, and H. Keller. The design of mars lander cameras for mars pathfinder, mars surveyor '98 and mars surveyor '01. *IEEE Transactions on Instrumentation and Measurement*, 50(1):63–71, Feb 2001.

[7] H. Yamada, T. Tominaga, and M. Ichikawa. An autonomous flying object navigated by real-time optical flow and visual target detection. In *Proceedings of the IEEE International Conference on Field Programmable Technology (FPT)*, pages 222–227, 2003.

[1] http://www.xilinx.com/publications/xcellonline/xcell 47/xc pdf/xc formula47.pdf