Supporting SoC on FPGAs

Professor Andy Downton and Dr Martin Fleury
Dept of Electronic Systems Engineering, University of Essex, UK

Background

In the next two decades, Moore's Law predicts continuing step increases in IC capacity at the expense of spiralling capital and fabrication outlay, which in turn will require increasingly massive markets to amortise the cost. Hitherto, 'a' if not 'the' major mass market for state-of-the-art ICs has been high-performance general-purpose microprocessors in PCs, but this market is mature and perhaps close to saturation. One way to match device capacity to applications is to turn from the ubiquitous microprocessor to a ubiquitous flexible embedded device, each device being a variant around a theme. The outcome is a heterogeneous SoC architecture such as that pioneered by ARM for smaller embedded systems, as well as recent multiprocessor chips released by Intel. A limitation of this approach is that it enforces a traditional co-processor model on the design tools used to build systems with such chips, which we would argue is not the only (or necessarily the best) way of exploiting the processing potential of future chips with a magnitude greater capacity over current devices. In fact, we advocate that the new ubiquitous embedded device could be a platform FPGA, such as the Xilinx Virtex family, provided one could arrive at a commonly agreed SoC architecture for such an FPGA.

FPGA trends

Like microprocessors, FPGAs have benefited from Moore's Law in terms of increasing device capacity and processing capability. Whereas they were originally intended as a form of configurable integrated glue logic, they are now being applied to network control and signal processing. However, unlike microprocessors, software development for FPGA-based applications is currently only at an evolutionary stage equivalent to a 1970s microprocessor development environment in terms of languages, operating systems, and peripheral device support. By development environment, we mean an environment that exists on the FPGA itself. Such an environment would mean that each application would no longer need to be developed from scratch and would reduce reliance on proprietary design tools. We would argue that potentially an FPGA-based SoCs can scale and adapt more readily than ASIC SoCs to future mass markets, but only if the current software tool culture is reversed, so that instead of viewing the microprocessor as the core of a SoC (with FPGA fabric treated as an uncommitted co-processor), the FPGA fabric itself becomes The Processor, with the potential to embed a variety of additional co-processing elements (including soft and hard microprocessor cores) on the same SoC. The vision below explores the implications of this inverted viewpoint.

SoC on FPGAs

An FPGA core can be viewed as a grid of processing elements (PEs) of arbitrary granularity, supported by a scalable interconnect. At its simplest level, the reconfigurable logic can emulate an SIMD machine or a systolic array. Unlike earlier SIMDs however, the PEs can be combined to form custom circuits, which in turn can be combined to form an embedded program of arbitrary structure, dimensions and complexity, removing most of the architectural restrictions that limited the use of SIMDs in the past. The program is embedded in the hardware structure, so there are no inherent inefficiencies associated with a fetch-execute cycle, and communication is not restricted to regular nearest neighbour communication. The latest platform FPGAs support multiple clock domains, potentially allowing a forest-like architecture combining multiple embedded programs, as well as offering the possibility of dynamic and partial reconfigurability. Gigabit serial transceivers are already available on-chip, as well as low-level network device drivers (*e.g.* for Ethernet and Bluetooth). Hence, FPGAs are also the natural architecture in which to explore and develop other foreseeable architectural refinements, such as network-on-a-chip (to overcome the communications bottleneck of current bus-based on-chip communication) and wireless chips (to reduce excessive pin-counts, one of the primary cost drivers in microelectronic fabrication). Finally, there is no fixed consensus on what elements SoC should include, with each manufacturer offering a variety of different versions targeted to different application domains, *e.g.* logic, signal processing, networking.

*So*ftware Environment for SoC on FPGAs

Unfortunately, development and exploitation of these capabilities is currently hindered by the limited and piecemeal nature of the co-design tools available for SoC on FPGA. To understand what is missing from the toolset, it is necessary (i) to abstract from a circuit-level view of the chip, as designing at this level will be increasingly unsustainable (and unnecessary) as complexity increases, and (ii) to propagate an FPGA-centric view of SoC (as outlined above) through all aspects of the toolset, by analogy with earlier generations of microprocessor development tools.

- Purpose-designed **programming languages** are the first pre-requisite for programming FPGAs, and several examples now exist (e.g. Handel-C, Impulse-C and Bach-C/Catapult-C). These abstract from the circuit description to the virtual application level, and hence are inherently more suited to co-design than previous hardware description languages, by directly mapping hardware aspects of the design such as parallelism, variable data widths, and hardware synchronisation to abstract software concepts (*e.g.* CSP in the case of Handel-C). Their common use of C as a semantic base however hides a number of underlying research issues concerned with the language timing model (synchronous, asynchronous or untimed) and how designers should reason about soft and hard real-time execution as they move from specification to implementation. These issues re-emerge as design moves from regular FPGA fabric alone to heterogeneous SoC on FPGA.
- **Device drivers** are currently provided as low-level function calls within the FPGA programming language, rather than being abstracted to a separate device-related Run-Time Environment (RTE). Memory access is idiosyncratic to the particular hardware configuration and largely remains the programmer's responsibility, rather than being handled by a generalised **memory manager** appropriate to the embedded applications for which such SoCs will typically be used. In future, **network support** may be required to other devices on-chip as well as off-chip. Currently, on-chip cores and transceivers are typically connected to a system bus or bus hierarchy, which presents problems in terms of scalability and in terms of interfacing multiple clock domains.
- Device drivers and memory managers are aspects of **Operating Systems (OS)**. There is as yet no OS purpose-designed for FPGAs, though there are, of course, conventional OSs that will run on soft or hard cores embedded within an FPGA. A true FPGA operating system would differ significantly from these. There might be no dynamic memory allocation or dynamic task scheduling, but dynamic reconfiguration would be needed; conventional time-shared task execution would not occur but task synchronisation and management are still required.
- **Debug** requirements extend all the way from co-simulation through parallel partitioning to on-chip performance tuning to meet soft and hard real-time performance constraints. Particular problems occur in debugging multiple heterogeneous co-processing elements on a single chip.

Many of the areas listed above were investigated in a different context in EPSRC's Portable Software Tools for Parallel Architectures programme a decade ago, but are more readily applicable to SoC on FPGA than to the discrete parallel processors of that time (or indeed to the current large-scale Grid), because FPGAs imposes no inherent granularity or physical structure on how parallelism is exploited.

Significance, Scale and Timeliness

The UK has an influential and innovatory international record in parallel computing, ranging from abstract conceptual work such as CSP through systems innovations such as the transputer to languages such as Handel-C. Mainstream architectures are dominated by international companies such as Intel and Xilinx, who control short term market trends, but opportunities exist to capitalise on the explosive growth of the FPGA applications market by developing the next generation of software tools for SoC on FPGA (as Celoxica have already done for basic FPGA fabrics). More significantly, software-defined FPGA-based systems are a much lower-cost route into embedded systems than ASICs, lending themselves to low and medium-volume applications, which are less likely to be worth exporting to third-world manufacturing, and hence provide some scope for sustaining the UK electronics industry in an area where it currently remains active. As chip capacities and design abstraction increases, it is foreseeable that more and more of the ASIC market will be absorbed by FPGAs (due to reduced design costs), providing increased competitive opportunities for the UK. To exploit these opportunities, academia needs to introduce its students to the next generation of SoC on FPGA design tools within degree programmes, emphasising the importance of maintaining close relationships between research, industry and undergraduate teaching.