

---

# Teaching digital controller design skills for embedded systems and mechatronics

Jonathan A. Dell

*Department of Electronics, University of York, Heslington, York, UK*

*E-mail: dell@ohm.york.ac.uk*

**Abstract** This paper describes the ECAD work undertaken by third-year MEng students in the Department of Electronics at York in support of embedded systems and mechatronics. It describes in detail the digital electronics computer aided design (ECAD) element of the course where the students use Xilinx Foundation CAD to develop and test a simplified microprocessor.

**Keywords** digital controller design; embedded systems; hierarchical design; mechatronics; Xilinx™ ECAD

MEng courses in the Department of Electronics at the University of York are of four years' duration. In the final year students undertake a six-month research project either in industry or within one of the department's research groups. In preparation for this all the MEng students undertake a number of electronics CAD exercises as the principal academic component during the final term of their third-year studies. During this period the MEng students are also completing a software engineering group project where groups of about eight students work together on a large software development task; this develops group working skills that will be essential in their later careers.

The location of the ECAD exercises at this point in their programme of study is selected because it consolidates the third-year academic study and prepares them for their individual project work in the later part of the fourth-year studies. The core material in the electronics CAD exercises comprises numerical methods for ECAD, analogue ECAD using PSPICE and digital ECAD using Xilinx<sup>1</sup> Foundation. Students additionally choose two from three optional ECAD exercises; these include control, communications and music, and are based around MATLAB and use appropriate SIMULINK toolboxes.

The main objective of the digital ECAD exercise is to gain first-hand experience of all stages in the digital design cycle and the principal task is to complete the design of a simplified microprocessor based around a 4-bit architecture. The work includes design entry by schematic capture and hardware description language (VHDL), functional and timing simulation. Also testing and fitting the design into a commercial field programmable gate array (FPGA) is required. Further objectives are:

- to gain an understanding of the hierarchical design approach which assists in the production of a complex system from the most primitive components;
- to obtain an insight into the details of processor operation; and
- to understand the development of effective test vectors in a hierarchical design.

On completion of the design the students can load the processor design into prototype hardware containing a Xilinx 4000 series FPGA. They can then observe the execution a few instructions in a simple program that activates the hardware in real time.

### **Digital ECAD laboratory work**

A teaching laboratory containing about 30 PC workstations is utilised for all the ECAD practical sessions. Students work in pairs, as this is found to be the most effective learning environment. The digital ECAD laboratory work is undertaken during two-hour sessions each week over a seven-week period. This comparatively short period means that the laboratory work has to be carefully planned. Additionally, students are expected to spend at least two hours of their own time between sessions preparing for the work they will be doing in the following session. A rough breakdown of the activities undertaken in each of the sessions and a guideline of the essential intervening work is shown in Table 1. The students are provided with extensive notes covering the design task and reference material on the use of the hardware description language (VHDL) and the Xilinx development package. The student's work is carefully monitored during the first few sessions by inspecting their logbooks. If students do not make sufficient progress in the early stages of the module developments they can be issued with model worked solutions for the dual-port register unit and the arithmetic unit thus enabling them to continue unimpeded with the rest of the work.

### **The simplified microprocessor design**

The overall design is carefully chosen to include the main features of a typical microprocessor architecture without introducing unnecessary complexity. The chosen 4-bit microprocessor design consists of four functional units, the controller, the data register, the arithmetic unit and the instruction memory. These units are interconnected by a top-level schematic that is provided to all the students as a starting point, this defines the architecture of the system and the detailed interconnection between the modules. The only unit for which the design has already been completed is the instruction memory and this is set up with a small program of instructions to furnish a rudimentary system test which allows the processor operation to be observed. A block diagram of the top-level design is shown in Fig. 1.

#### **The register unit**

The register unit consists of three 4-bit D-type register blocks, two of these form general purpose data registers and the third acts as the programme counter (PC) providing the instruction memory address. The students have to start by developing a generic master/slave latch design from basic gates as the latch in the Xilinx component library has inappropriate characteristics. A single level of hierarchy is used to combine four of these latches to form a 4-bit register block. Three of these blocks are further combined by another level of hierarchy to form the dual-port register

TABLE 1 *Session work breakdown*

Session	Main task
1	Overview of the exercise and familiarisation with the Xilinx ECAD package. A simple counter design is provided so that the students can familiarise themselves with all stages of the design process. This design is described in a circuit schematic and after compilation can be downloaded onto a prototype hardware system containing an FPGA where it can be observed to operate in real time. Prior to the following session detailed descriptions for the dual port register and arithmetic units are studied and initial designs are prepared.
2	Implementation of the dual port register and arithmetic units is undertaken during sessions two and three. This involves initially establishing the basic functional elements, such as the full adder and master/slave flip/flop, using simple gates and the schematic capture facilities provided in the Xilinx package. These are then combined into the required functional blocks through a hierarchical design process that is supported by the package. The Xilinx interactive simulation tools are used to provide functional simulation and assist with debug of the designed elements. Prior to session four the effective testing of the register and arithmetic parts of the processor structure is analysed and a suite of test vectors has to be designed using a form provided with the notes.
3	
4	Further development of the functional units, in particular the aspects of their control and timing needed to satisfy the overall design requirements, is undertaken during sessions three and four. Also implementation of the test vectors is accomplished through the interactive simulation facilities provided. Integration of the completed dual-port register and arithmetic units into the top-level design is completed. Prior to session six a detailed description of the control block is studied and the use of VHDL to establish a behavioral description is revised. A finite state machine (FSM) implementation for the controller is designed making use of VHDL for its description.
5	
6	Implementation of the control unit VHDL design is undertaken during sessions six and seven. The completed design is used to form a new schematic element and this forms the final element of the top-level design. Initially students are encouraged to implement a single processor instruction to facilitate debugging of the completed processor. As further instructions are implemented more thorough testing of the complete processor design can be performed. Finally on completing the design processes, which includes device fitting, the processor can be downloaded into the prototype system containing an FPGA where its operation in real-time can be observed.
7	

unit. For simplicity the register control signals are not encoded so that all register outputs can be routed to either of the output buses A and B through appropriate arrangements of simple gates. Register input data is provided from a single input bus so further control inputs are required to determine which register is updated. The latch updating clock must be arranged so that data processed through the arithmetic unit can be written back to the same source register later in the same processor cycle. An asynchronous clear function is also required to initialise the system.

Once the design of this block is complete the Xilinx interactive simulation facilities can be employed to exercise the registers, observe their functional behaviour

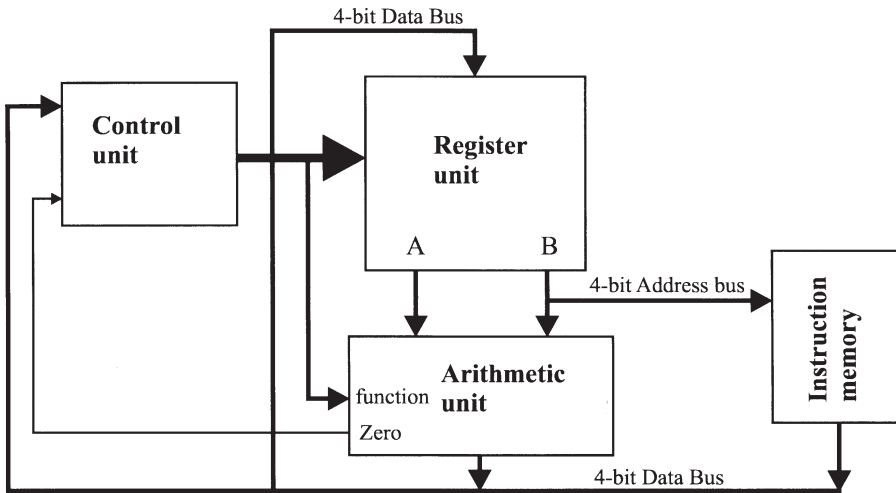


Fig. 1 Simple microprocessor design.

TABLE 2 Arithmetic unit functions (\*represents don't care)

$f_2$	$f_1$	$f_0$	Function
0	0	0	pass through ( $A \rightarrow Aout$ )
0	0	1	increment ( $A + 1 \rightarrow Aout$ )
0	1	0	decrement ( $A - 1 \rightarrow Aout$ )
0	1	1	*
1	0	0	add ( $A + B \rightarrow Aout$ )
1	0	1	*
1	1	0	*
1	1	1	subtract ( $A - B \rightarrow Aout$ )

and examine their timing. For the correct operation of the register block in the overall system all aspects of its design must be correct. Students can make a hard copy of the simulated timing diagrams for inclusion in their logbooks.

### The arithmetic unit

The arithmetic unit has two 4-bit inputs, a 4-bit output and a single status output indicating when the arithmetic result is zero. It is required to perform a simple family of operations according to a 3-bit function code as shown in Table 2. The function control bits have been arranged to minimise the gating required in the final implementation.

Students are initially required to develop a basic half-adder from simple gate elements and thus through a single level of hierarchy a full adder. A further level of hierarchy is used to combine these full adders to form a 4-bit unit. This unit must

then be combined with appropriate data manipulation and routing controls to establish the required range of operations. These can all be implemented through simple gate structures. A latched zero flag, which reflects the result of the previous arithmetic operation, is also needed to implement one of the required processor instructions.

Once the design of this block is complete the Xilinx interactive simulation facilities can again be employed to exercise the arithmetic functions, observe their functional behaviour and examine their timing. For the correct operation of the arithmetic block in the overall system all aspects of its design must be correct. Students can make a hard copy of the simulated timing diagrams for inclusion in their logbooks.

### The controller

The primary job of the control unit is to manage the instruction fetch-execute cycle of the processor and to direct the operation of the data processing units for each of the various phases associated with the different instructions. The outputs of the control unit are connected directly to the controlling inputs of the arithmetic and dual port register units so their correct sequence is vitally important. Controller inputs come principally from the instruction memory but the arithmetic zero flag forms a conditional input which is used by one instruction. Students are provided with a form to help with the planning of the instruction sequences and the setting of the control bits.

A repertoire of just four instructions is required by the microprocessor design; these are detailed in Table 3. These have been carefully chosen to provide useful functionality, but could easily be extended if required. In detail, the control unit has to fetch instructions from the instruction memory, decode the operation code presented, implement a sequence of operations in the arithmetic and dual port register units as required to perform the particular instruction. Students are encouraged to work on one simple instruction such as **inc a** to start with as this allows the instruction code to be ignored until later. Control for the other instructions follows similar lines and students generally find their design work quite straightforward.

Students are required to develop the control unit using a VHDL behavioural description where the simple state machine structure is typically implemented using a case<sup>2</sup> statement. It is then straightforward to add additional instructions as new

TABLE 3 *Microprocessor instructions*

Instruction mnemonic	Machine code	Function
Br	2	Branch unconditionally to the address given in the next instruction word.
Bnz	1	Branch to the address given in the next instruction word if the result of the last arithmetic operation is NOT zero.
inc a	8	Add 1 to the current contents of the A register.
dec a	9	Subtract 1 from the current contents of the A register.

cases, when the details of their design become apparent. They are also encouraged to make the state variable bits available as additional controller outputs so that the current state of the controller can be observed in the simulation. These extra outputs assist greatly with debugging the controller and can be used later when the hardware implementation is tested.

Once the design of the controller block is complete the Xilinx interactive simulation facilities can again be employed to exercise it, observe its functional behaviour and examine its timing. For the correct operation of the controller in the overall system all aspects of its design must be correct. During initial testing the inputs to the controller, i.e. the 4-bit instruction code and the zero flag bit, are disconnected from the other parts of the system. When the controller's correct operation is confirmed these connections can be restored.

### The instruction memory

The instruction memory is pre-programmed with a simple double loop task suitable for observing the processor's operation and this is shown in Table 4. The instruction code is gated onto the internal data bus by an additional output from the controller, this is to avoid conflict with the output of the arithmetic unit. The two units are simply arranged to be mutually exclusive. The instruction memory address is provided from the register B output bus so the program counter (PC) must be routed to this port during the instruction fetch phase of the operating cycle.

The registers are all reset to zero so that the first instruction **inc a** is obtained from address zero. The first loop repeats until the contents of register A returns to zero after fifteen loops have been performed. The second loop uses the **dec a** instruction but otherwise operates in a similar manner. The final instruction ensures that the loops continue forever.

### Testing the designs

Test design is one of the most challenging aspects of the digital design process. As with any system design using VLSI technology it is not possible to construct exhaus-

TABLE 4 *The simple program instructions*

Address	Machine code	Mnemonic
0	8	loop1: inc A
1	1	Bnz
2	0	Loop1 (branch addr.)
3	9	loop2: dec A
4	1	Bnz
5	3	Loop2 (branch addr.)
6	2	Br
7	0	Loop1 (branch addr.)
8	f	Unused
...	...	...
f	f	Unused

tive tests for every component in the system and all nodes in the circuit. In designing tests for the arithmetic and register architecture within this processing structure students are expected to develop a compromise strategy where every data path is exercised with both zero and one to ensure that no stuck-at faults exist. They are encouraged to follow the guidelines given in their laboratory notes and these are shown below:

- ensure that every bit on every data path, including the paths in the arithmetic unit and the registers, is driven by at least one '1' and one '0' in the test data set.
- ensure that all registers are written and read through every available port with alternating data patterns.
- ensure that alternating data is transferred between all registers in the data processing structure.
- ensure that all functions of the arithmetic unit are tested with these patterns.

Also, to minimise test time for the data processing structure, the students are encouraged to design their test data set so that the vectors will require less than 20 clock cycles to complete. To assist in the design of test data students are given a pro-forma which helps to remind them of the control input and output signals in question.

For simplicity of testing, the controller is considered as a separate exercise. Here it is most convenient to monitor the state changes as they occur. This made possible by the state variable bits having been made visible. The progress of instruction execution can then be observed as well as the controller outputs that will eventually affect the data processing structure. Again students are given a pro-forma to direct the establishment of these tests and remind them of the signals involved.

### The prototype hardware

The teaching laboratory is equipped with a prototype system designed at York and this actually contains a Texas Instruments DSP component as well as a Xilinx FPGA component. The DSP system is used in some of the other ECAD exercises but for the purposes of this design exercise the DSP system is disabled and the FPGA is in full control of the prototype system. The prototype hardware also has a front panel with 16 digital inputs and 16 latched digital outputs with associated LED displays. This enables the observation in real-time of signals, such as the state variable bits or the internal data bus, associated with the system design. The hardware contains an XC4005E FPGA into which the design is loaded through a proprietary download cable.

The hardware configuration determines that specific pins on the FPGA are connected to the inputs and latched digital outputs. In order to link the internal data bus or other parts of the system to these pins the circuit compiler is forced by locking the pins within the design. A further pin is linked to the output latch enable and this must be driven by a signal from the controller so that the output latch is updated only when the required register content or other data is available on the bus. Finally if the processor is connected to a suitable clock the changes in content of register A can be observed as the program is executed.

## Assessment

The MEng ECAD exercises are all marked and contribute to the overall assessment of the third year. The main vehicle for assessment of the Digital ECAD exercise is the logbook which students keep individually as the work proceeds. This will typically contain initial designs on paper, completed pro-formae for data processing and controller tests, simulation results such as waveform diagrams and brief comments. These logbooks are handed in for marking at the end of the exercise but as noted previously they are monitored closely by the lab staff during the early stages of the exercise.

Students are also required to submit specific items at several stages during the exercise, mainly to provide an indication of their progress. At the end of session three they are required to submit the designs for the dual-port register unit and the arithmetic unit. If it appears that they have made insufficient progress to proceed further without considerable delay worked solutions can be provided.

The test vector design for the data processing functions is submitted prior to session four to determine if they are following the correct approach, additional guidance can be given at this stage if required. Simulation results, which are derived from their practical work, forms an important part of the logbook evidence and must be submitted with the logbook after the exercise is completed.

## Conclusion

The Digital ECAD exercise has been run successfully over the last four years with only minor modifications to accommodate later versions of the Xilinx package. Students find the work challenging and fulfilling and course feedback suggests that beneficial experiences have been achieved.

The Xilinx CAD software has proved reliable and moderately straightforward for the students once they have climbed the steep learning curve at the start. The lab demonstrators quickly solve most of the difficulties encountered. Experience has shown that the use of these CAD tools is extremely beneficial as many students go on to use Xilinx in their Industrial placements and research projects. Students going on to use similar ECAD packages find they are able to make a flying start on their projects.

The exercise is seen to meet the principle objective of exposing the students to the full design cycle and underpin their understanding of digital systems. It also reinforces their understanding of the VHDL language and the concepts of design hierarchy. The student's knowledge and appreciation of microprocessor architecture and operation is also enhanced.

Although the processor design is quite simple it provides the basis for the controller employed in a Mechatronic or embedded system as its capability can be easily extended and tailored to the actual requirements.

## References

- 1 See the site [www.xilinx.com](http://www.xilinx.com)
- 2 Charles H. Roth Jr., *Digital Systems Using VHDL* (PWS Publishing, Boston, 1997).