Part A – SRAM & SDRAM

The Nios II hardware development tutorial synthesises a NIOS II processor with, 20 KB of on-chip memory, a timer, a JTAG UART, 8 parallel I/O pins and a system ID Component. Part A of assignment 4 requires you to interface the 512 KB SRAM and 8 MB SDRAM on the Altera DE2 Board to this design.
You should then test that the memory is functioning correctly by running the Memory Test programs available within the NIOS II IDE. You should modify the Memory Test Programme so that your Name and ID number are shown in the terminal window each time the memory is tested.
Hints
1. Initially use the altera_up_avalon_sram Controller (SRAM/SSRAM) for the SRAM interface
2. Use the SDRAM controller for the SDRAM Interface
3. Add a PLL to advance the clock for the SDRAM by 3ns compared with the system clock.

Part B – Custom Instruction

Part B requires you to develop a Custom Instruction to count the number of leading 1s (or 0s – see Table 1) in the 32 bit number passed to the instruction. You should write a program to test your Custom Instruction. You should also develop a test routine in C or assembler that performs the same function as the Custom Instruction and compare the speed of the Custom Instruction against your software implementation.

Part C – PWM module
Pulse Width Modulation (PWM) has many uses, but it main application is to allow the control of the power to electrical devices. For Part C, you are to use PWM to control the intensity of an LED (See Table 1) on the DE2 Board. You should write software so that the intensity gradually varies from off to fully on to fully off and then repeats.

To implement a PWM generator typically three components are required a) A PWM register which the programme writes to, that stores the required PWM value, b) A counter, of the same width as the PWM register, that counts up to the maximum value, goes back to zero and then repeats, c) A comparator that compares the value of the counter with that of the register and then generates the single bit PWM output.

There are many ways that you can implement the PWM module, the simplest way is to use a QSYS GPIO module as the register that the programme can write to and then outside QSYS generate the counter and comparator. A more elegant solution is to generate a full QSYS block that has the register and comparator and single bit output.