LAB Assignment #2 for Hardware/Software Codesign with FPGAs

Assigned: Mon., Oct 3, 2013
Due: Mon., Oct 7, 2013

Description: Implement a slave peripherial and incorporate more advanced LFSR.

Lab 2 is similar to the last lab but this time with a more interesting sequence. Please download and incorporate the LFSR code from the webpage ‘Lab #2 VHDL code’ link. You need to create a slave peripherial with 6 registers.

slv_reg0:

Will remain read-write by the processor with bit(0) tied to the ’_start’, bit(1) to ’_continue’ and bit(2) to ’_exhaustive_or_subset’.

You should change the read-write behavior for bit(1) so that it is cleared (set to 0) one clock cycle AFTER it is set to a ’1’ by a write from your C code.

The remaining slave registers will be updated ONLY by the vhdl code that I’ve supplied.

slv_reg1:
Row1 (low order 6-bits)
slv_reg2:
Col1 (low order 6_bits)
slv_reg3:
Row2 (low order 6-bits)
slv_reg4:
Col2 (low order 6_bits)
slv_reg5:
bit(0) is ’_ready’ and bit(1) is ’_OverRunErr’

Your C code should write the ’_exhaustive_or_subset’ bit with a ’0’ (default) if doing exhaustive (which generates more than 8 million combinations) or a ’1’ for subset (which generates more than 30K combinations). You must ensure that this bit remain fixed at 0 or 1 during the entire execution.

Your C code should write the ’_start’ bit with a ’1’ and then in the next stmt with a ’0’. You should then add a while loop that checks the status of bit(1) in slv_reg5, when it becomes ’1’, the loop should terminate.

Inside the while loop, wait for the ’_ready’ bit to be asserted, and when that happens, read and print the contents of slv_reg1 through slve_reg4. You should then write a ’1’ to the ’_continue’ bit to inform the VHDL code to generate the next pairing.

Study the output sequence and report any patterns in the behavior that you notice. Prepare to demo this in class.