Notes on the LCD controller design

* The AHB bus used does not support splits.
* The System runs off of HCLK, The external clock is not used…
* HPROT is not used
* The maximum HREADY cycles is 1 for the slave interface
* The maximum HREADY cycles is 32 for the master interface
* All memory reads should be in incremental bursts whenever possible.
  + Ensure the bursts do not cross a 1kB boundry (10 bits of addressing)
  + No cache line bursting needed or supported
  + The max burst size is programmable in a control register
* The pixel clock is defined by 0xE01F C1B8 LCD Configuration register
* Assume the CCLK is HCLK for the problem
* The line clock is the rising edge of HSYNC
* The internal FIFOs are changed to 32x32 each. These are on the memory interface.
  + This is not the best for low power, but easier to design
  + Make sure you have room for a burst when you start one
  + There are two fifo memories.
    - These may be used as a single 64x32 fifo, or two fifos depending on single or dual panel display modes
* HSIZE is always 32 bits (codes as 010) The bus size of this system.
* HBURST is either 0 or 1, No cache line burst are in the system
* The controller only supports little endian byte and pixel ordering
  + The control bits for big endian are inactive
    - No BEBO or BEPO supported. (These bits ignored)
    - Only little endian modes supported.
* The LCD gray scaling mechanism for STN panels is a frame number comparison. You will have to add a frame counter. The frame counter counts 1 to 15 using a CRC ox X3+X+1.
* This can be generated as frame0\_d = (frame0<<1)^((frame0[3])?3:0); when the vsync occurs.
* The resulting pixel is a 1 if the color code is >= the frame counter. It is zero otherwise. Code 0 is no pixels on, and code 15 is a pixel on every frame. This is reset to 1 at the start of simulation. The frame number is sent out of the design. The frame number is modified for comparison. To prevent visual artifacts, each pixel has a modified frame modification selected by a position value. This value is calculated concatenating the line number and column number into a 20 bit value. The 4 bit xor value is selects a set of bits from the counter according to the selection table which follows.
* Row column XOR to 4 bits
  + Bit 3 -- xor of the following position address bits
    - 1
    - 5
    - 6
    - 7
    - 8
    - 10
    - 12
    - 13
    - 16
  + Bit 2
    - 2
    - 6
    - 7
    - 8
    - 9
    - 11
    - 13
    - 14
    - 17
  + Bit 1
    - 0
    - 3
    - 7
    - 8
    - 9
    - 10
    - 12
    - 14
    - 15
    - 18
  + Bit 0
    - 0
    - 4
    - 5
    - 6
    - 7
    - 9
    - 11
    - 12
    - 15
    - 19
* Selection table (Taken from actual verilog code). Frame0 is the frame counter, xv1 is the 4 bit value from the position XOR.
  + case(xv1)
  + 0: frame0a=frame0;
  + 1: frame0a={ frame0[3],frame0[1],frame0[2],frame0[0]};
  + 2: frame0a={ frame0[0],frame0[1],frame0[2],frame0[3]};
  + 3: frame0a={ frame0[1],frame0[0],frame0[2],frame0[3]};
  + 4: frame0a={ frame0[2],frame0[0],frame0[1],frame0[3]};
  + 5: frame0a={ frame0[2],frame0[0],frame0[3],frame0[1]};
  + 6: frame0a={ frame0[0],frame0[3],frame0[2],frame0[1]};
  + 7: frame0a={ frame0[0],frame0[3],frame0[1],frame0[2]};
  + 8: frame0a={ frame0[2],frame0[1],frame0[0],frame0[3]};
  + 9: frame0a={ frame0[0],frame0[2],frame0[3],frame0[1]};
  + 10: frame0a={ frame0[2],frame0[1],frame0[3],frame0[0]};
  + 11: frame0a={ frame0[1],frame0[0],frame0[3],frame0[2]};
  + 12: frame0a={ frame0[1],frame0[2],frame0[0],frame0[3]};
  + 13: frame0a={ frame0[1],frame0[2],frame0[3],frame0[0]};
  + 14: frame0a={ frame0[3],frame0[0],frame0[1],frame0[2]};
  + 15: frame0a={ frame0[3],frame0[0],frame0[2],frame0[1]};