LCD controller IP

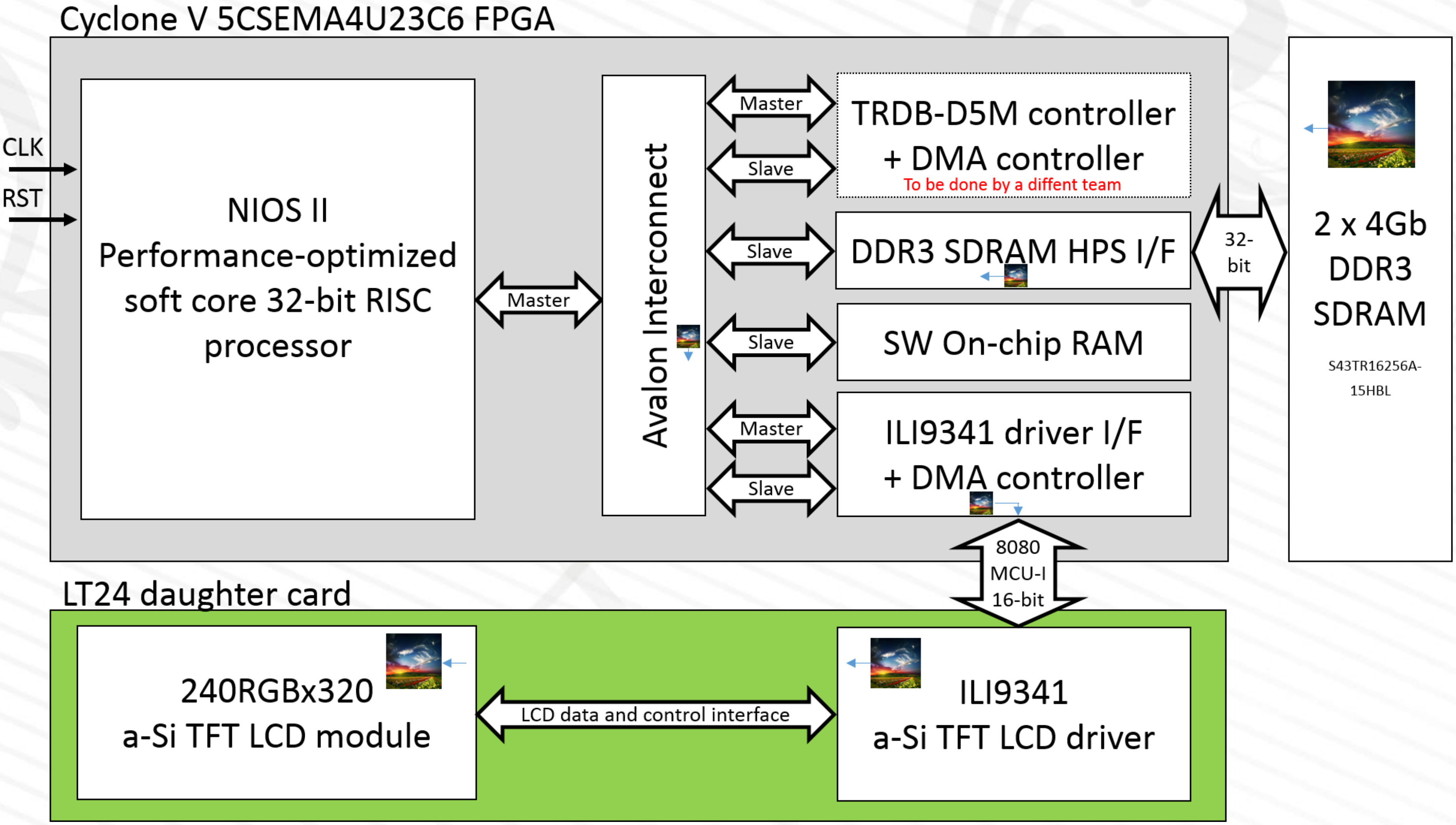
Marcos Vinicius Silva Oliveira

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

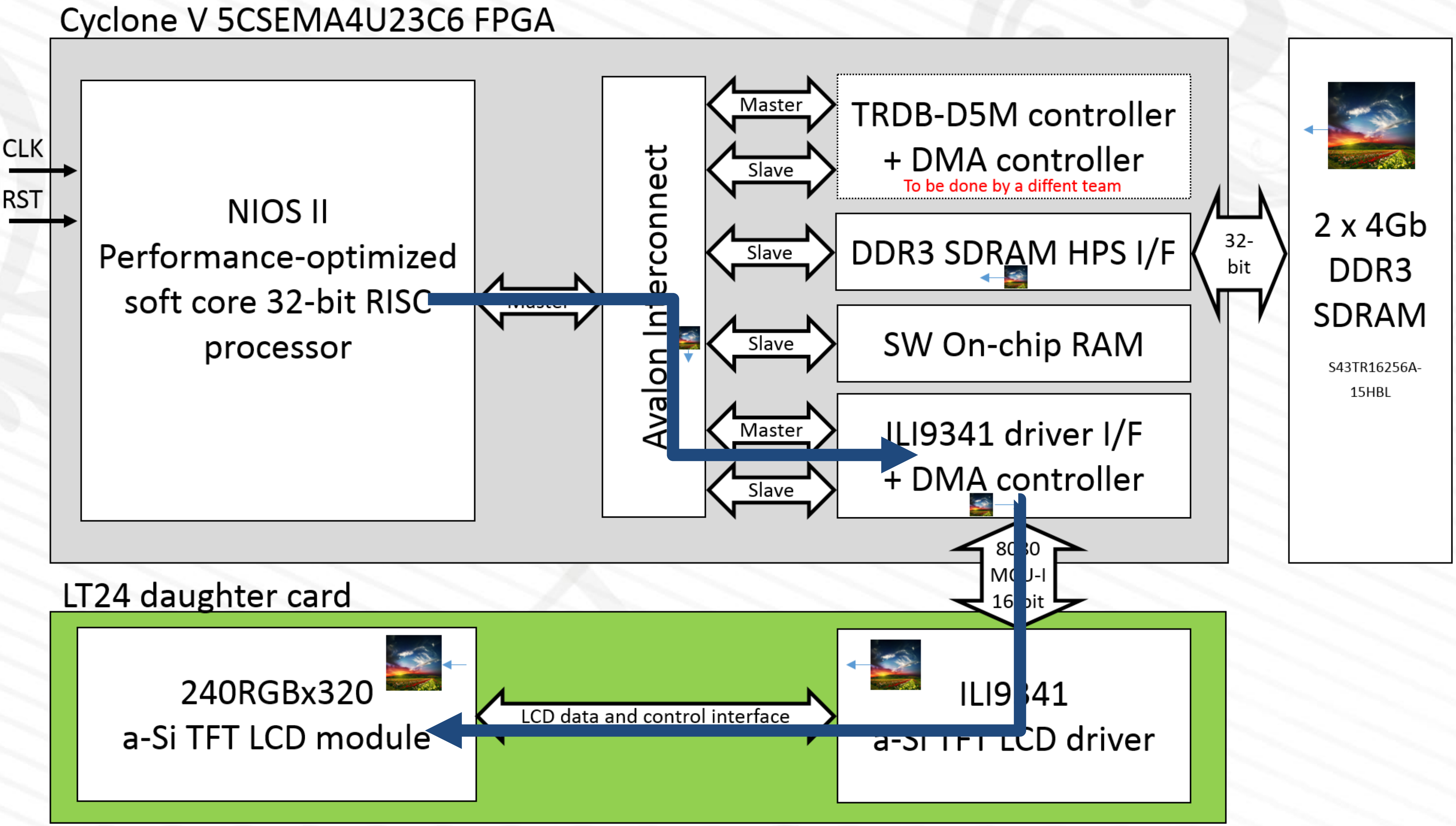
Summary

- Lab3 block diagram
- Modes of operation
- Constraints
- LCD controlleer IP
    - Freatures
    - Register Map
    - Top level connections
    - ILI 4391 connections
    - Block diagam
    - Simulation Results
    - Logic Analyser results
    - Demo

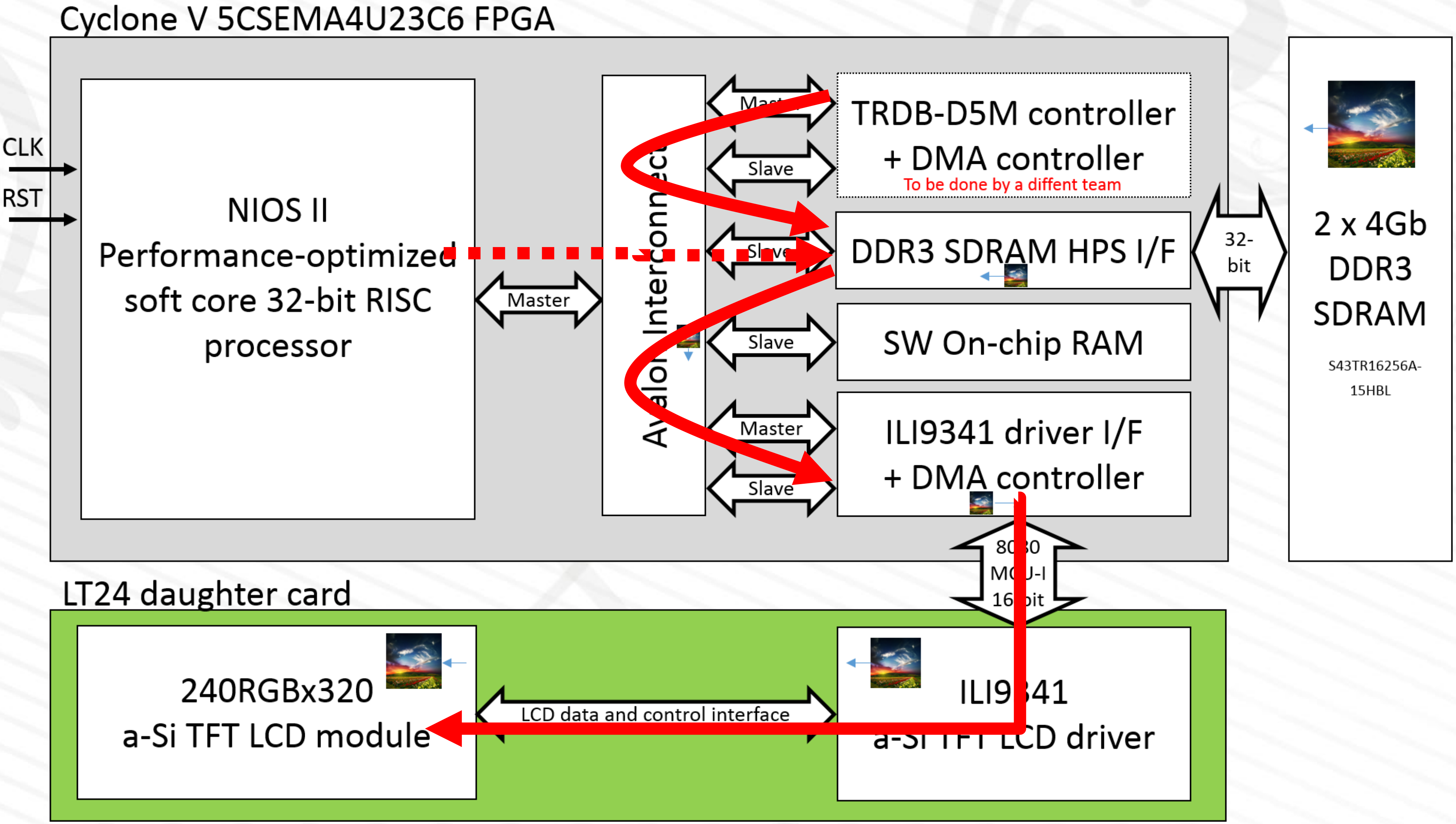
Lab3 block diagram



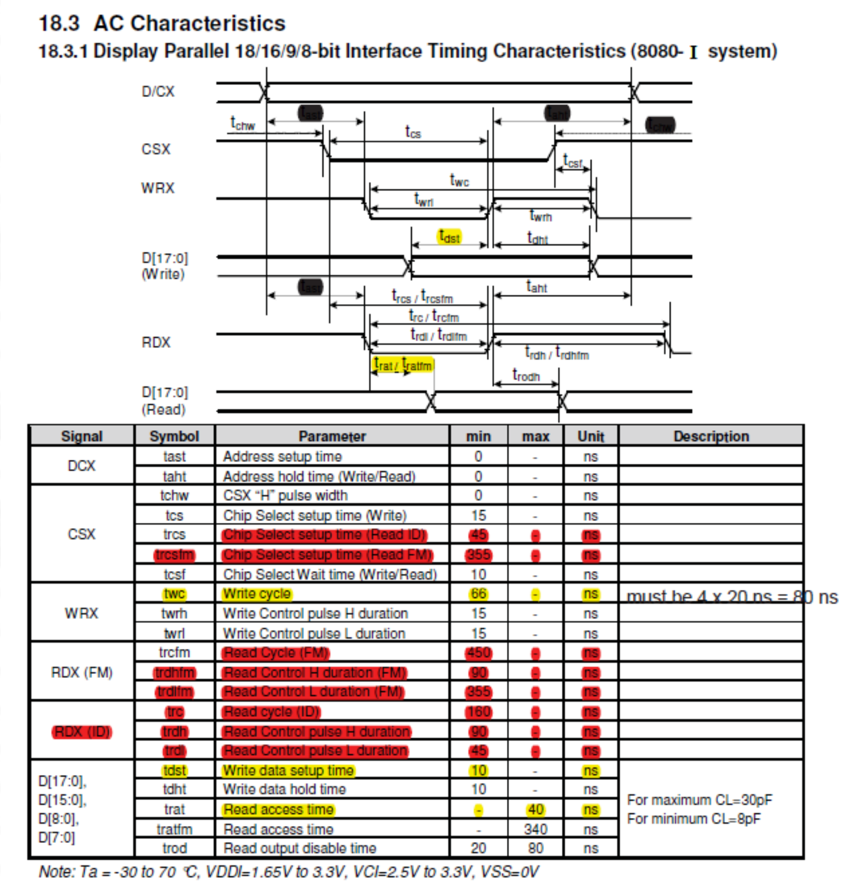
Mode of operation A



Mode of operation B



Constraints



18.3 AC Characteristics

18.3.1 Display Parallel 18/16/9/8-bit Interface Timing Characteristics (8080- I system)

Signal	Symbol	Parameter	min	max	Unit	Description
DCX	tast	Address setup time	0	-	ns	
	taht	Address hold time (Write/Read)	0	-	ns	
CSX	tchw	CSX "H" pulse width	0	-	ns	
	tcs	Chip Select setup time (Write)	15	-	ns	
	trcs	Chip Select setup time (Read ID)	45	-	ns	
	trcsfm	Chip Select setup time (Read FM)	355	-	ns	
	tcsf	Chip Select Wait time (Write/Read)	10	-	ns	
WRX	twc	Write cycle	66	-	ns	must be 4 x 20 ns = 80 ns
	twrh	Write Control pulse H duration	15	-	ns	
	twrl	Write Control pulse L duration	15	-	ns	
RDX (FM)	trcfm	Read Cycle (FM)	450	-	ns	
	trdhfm	Read Control H duration (FM)	90	-	ns	
	trdlfm	Read Control L duration (FM)	355	-	ns	
RDX (ID)	trc	Read cycle (ID)	160	-	ns	
	trdh	Read Control pulse H duration	90	-	ns	
	trdl	Read Control pulse L duration	45	-	ns	
D[17:0], D[15:0], D[8:0], D[7:0]	tdst	Write data setup time	10	-	ns	For maximum CL=30pF For minimum CL=8pF
	tdht	Write data hold time	10	-	ns	
	trat	Read access time	-	40	ns	
	tratfm	Read access time	-	340	ns	
	trod	Read output disable time	20	80	ns	

Note: Ta = -30 to 70 ℃, VDDI=1.65V to 3.3V, VCI=2.5V to 3.3V, VSS=0V

- Control signals synchronous with Avalon clock -> 50 MHz
- Writing cycles should last 4 clock cycles fo meet the min twc of 66 ns.
- Read cycles should last at least **9** cycles to reach the trcfm of 450 ns.
- Dummy data in read cycle
  - Handled in SW as performance not important for read accesses

Maximum frame refresh rate:

$$\frac{50\ \text{MHz}}{4\times240\times320} = 162.76\ \text{Hz}.$$

Freatures

LCD controller IP features	
Modes of operation	A: Processor writes data to ILI9341 B: DMA engine writes data to ILI9341
ILI9341 write access	Yes
ILI9341 read access	Yes
Multiple buffering	Implemented in SW
Number of buffers	Not limited by the IP (limited by off-chip memory)
Max Frame rate	162.76 Hz
Minimal interval between frames	configurable by PUSHBUTTONS / SW
Frame receiveid Ack	Accessible by the Arduino interface D0 pin
FIFO size	256 words (1 M4K)
StartAdress	configurable by SW
EndAddress	configurable by SW
DMA Request Mode	New request only after current is finished
DMA burst length	configurable by SW (2-256) (could be 1024 if FIFO up)
FIFO threshold	configurable by SW
IRQ support	Yes
IRQ masking	configurable by SW
DMA sync rst	set by SW
Debug LCD FSM	Yes
Debug DMA FSM	Yes
Debug DMA Addr	Yes
Debug DMA RX DATA	Yes
Debug Pushbuttons	Yes

Register Map

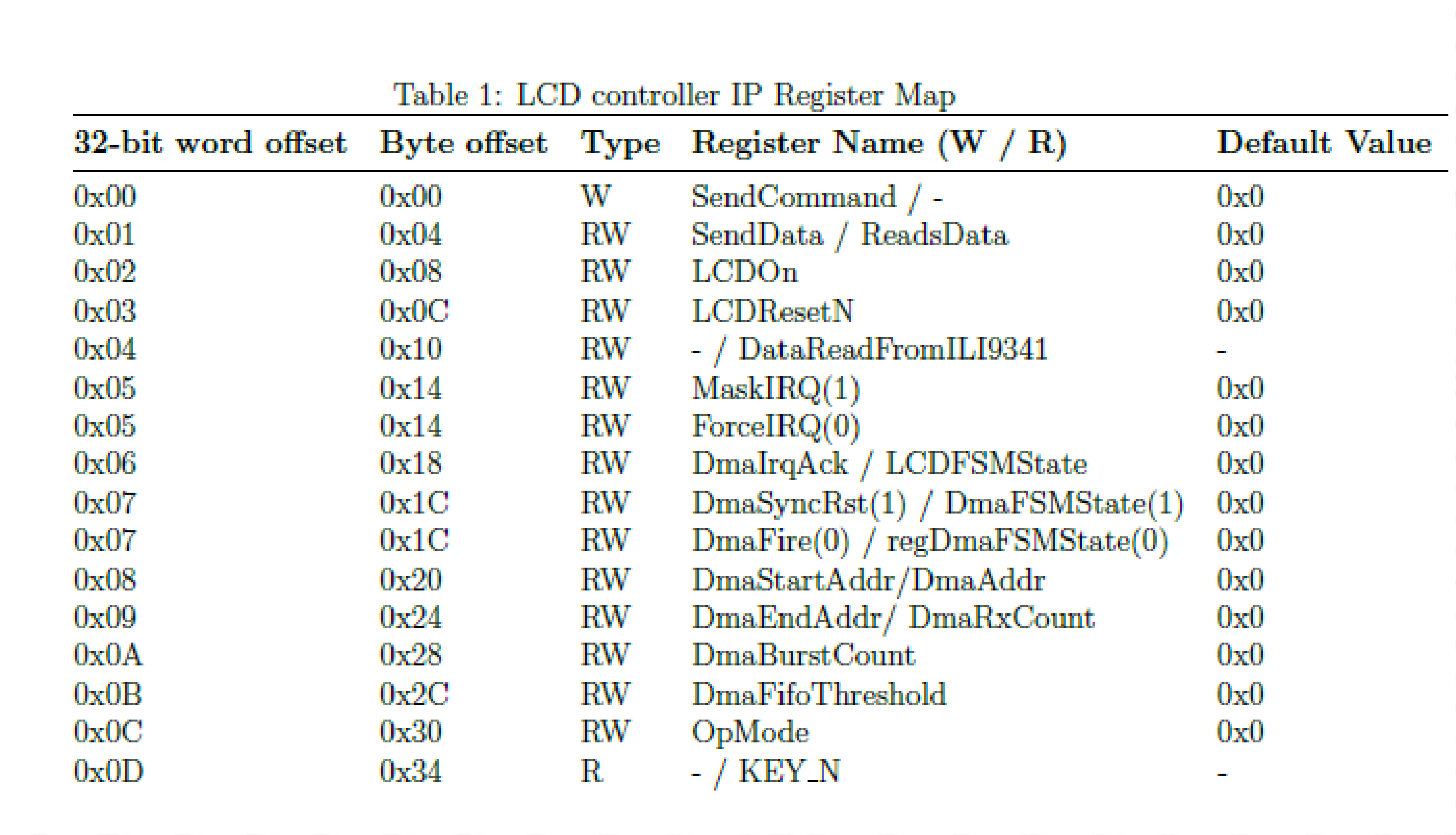
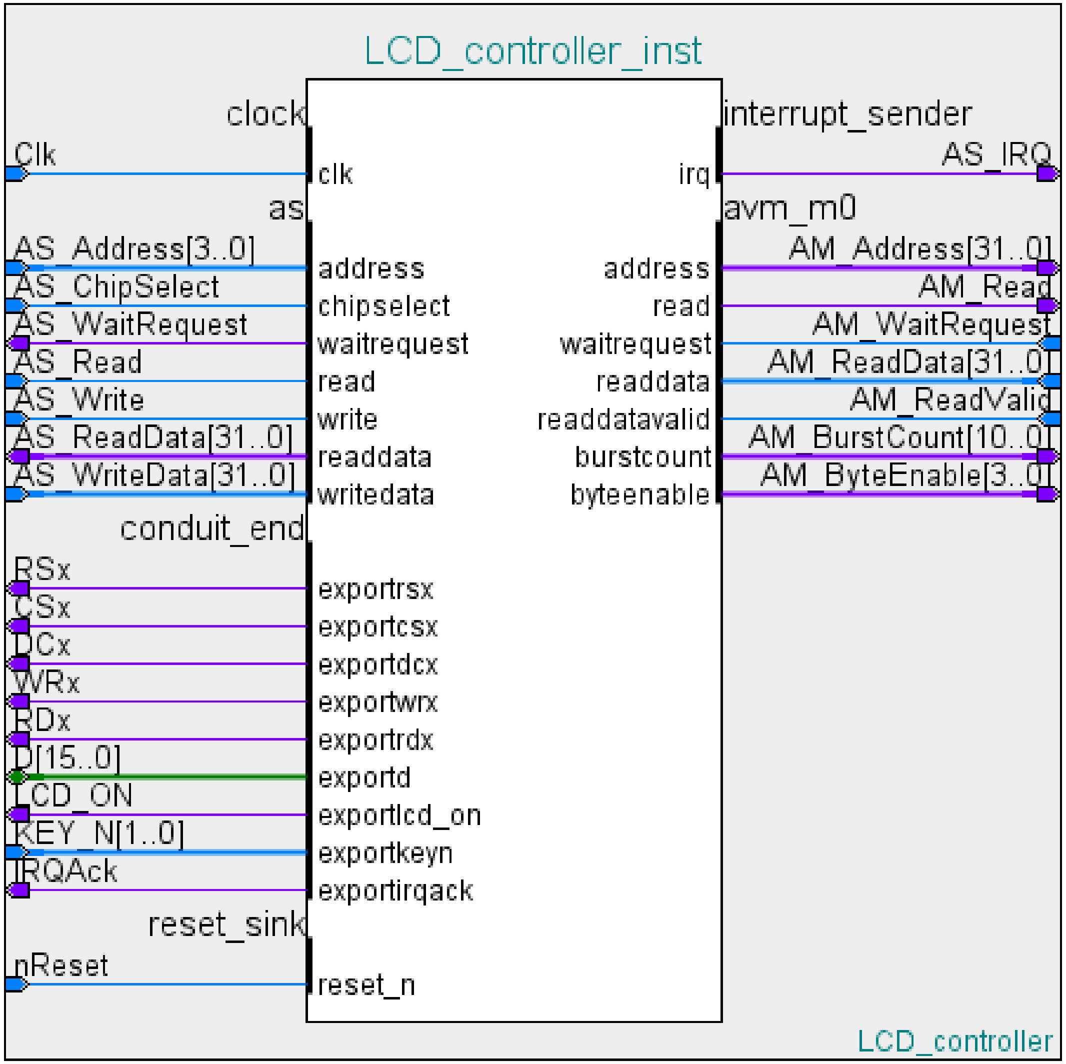


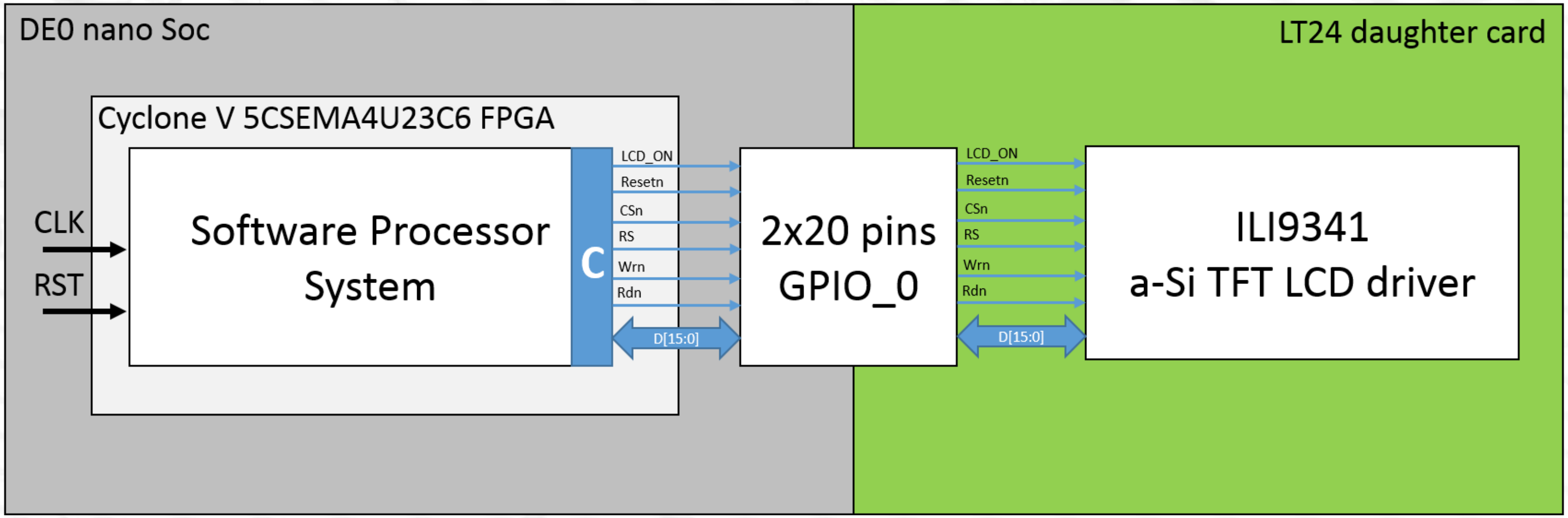
Table 1: LCD controller IP Register Map

32-bit word offset	Byte offset	Type	Register Name (W / R)	Default Value
0x00	0x00	W	SendCommand / -	0x0
0x01	0x04	RW	SendData / ReadsData	0x0
0x02	0x08	RW	LCDOn	0x0
0x03	0x0C	RW	LCDResetN	0x0
0x04	0x10	RW	- / DataReadFromILI9341	-
0x05	0x14	RW	MaskIRQ(1)	0x0
0x05	0x14	RW	ForceIRQ(0)	0x0
0x06	0x18	RW	DmaIrqAck / LCDFSMState	0x0
0x07	0x1C	RW	DmaSyncRst(1) / DmaFSMState(1)	0x0
0x07	0x1C	RW	DmaFire(0) / regDmaFSMState(0)	0x0
0x08	0x20	RW	DmaStartAddr/DmaAddr	0x0
0x09	0x24	RW	DmaEndAddr/ DmaRxCount	0x0
0x0A	0x28	RW	DmaBurstCount	0x0
0x0B	0x2C	RW	DmaFifoThreshold	0x0
0x0C	0x30	RW	OpMode	0x0
0x0D	0x34	R	- / KEY_N	-

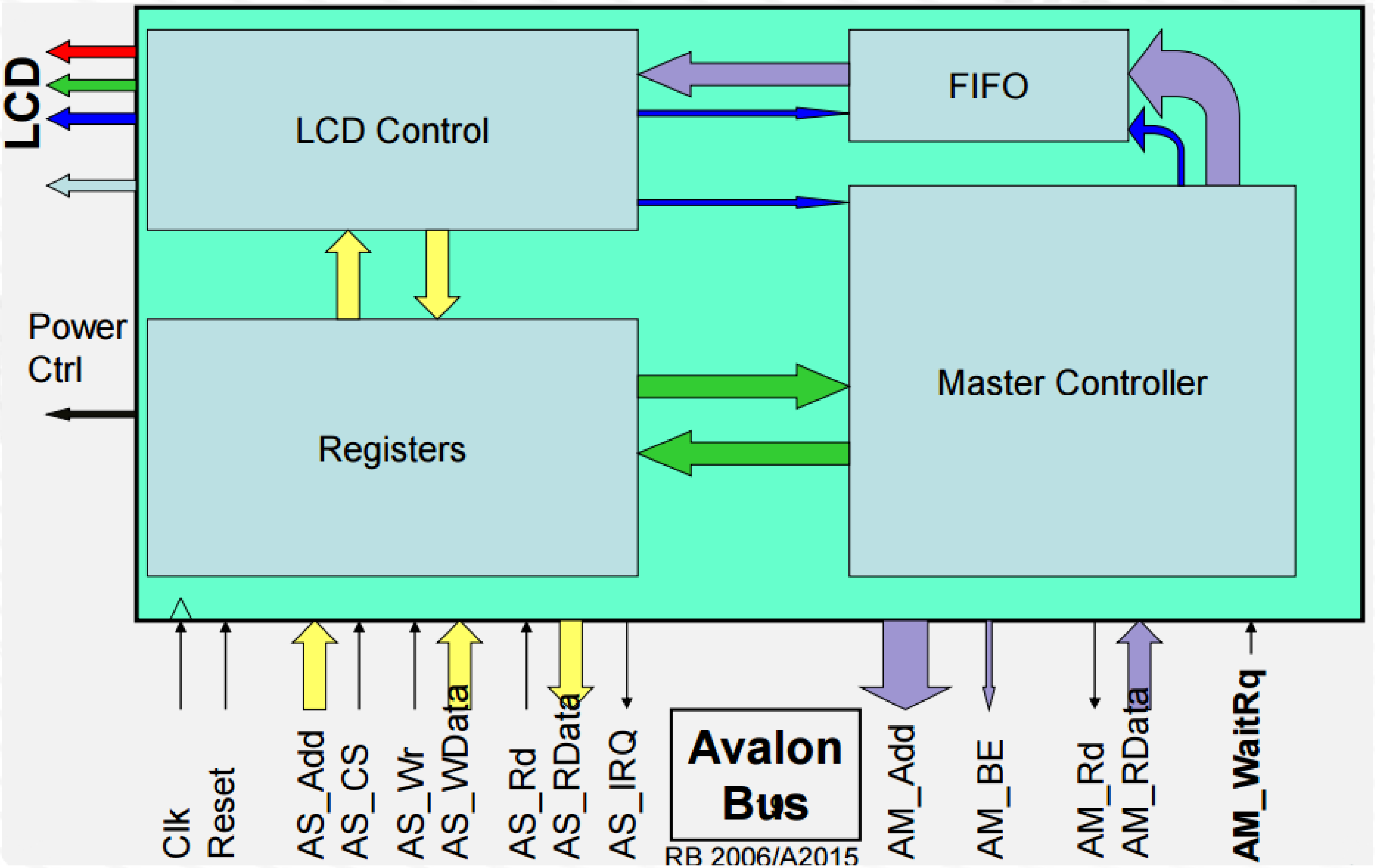
LCD controller IP



ILI9341 <-> LCD controller IP

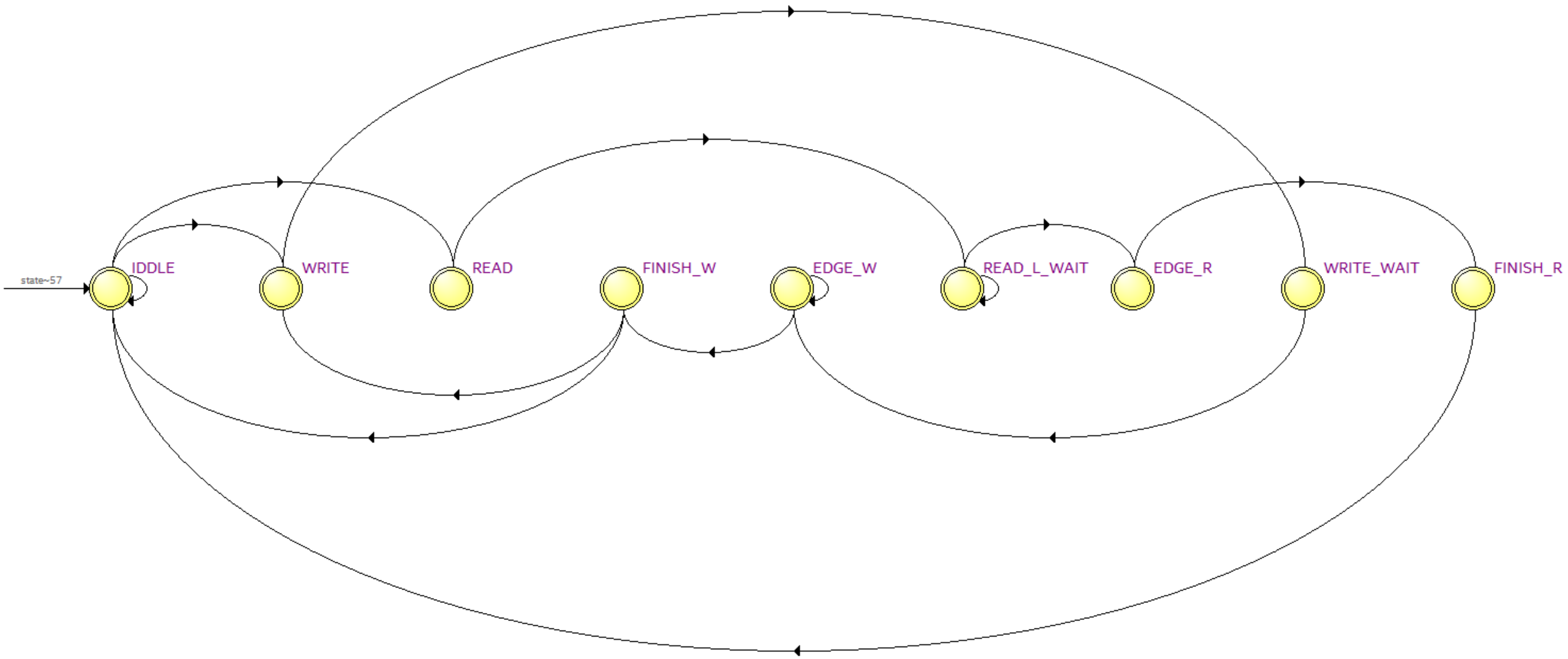


LCD controller IP block diagram

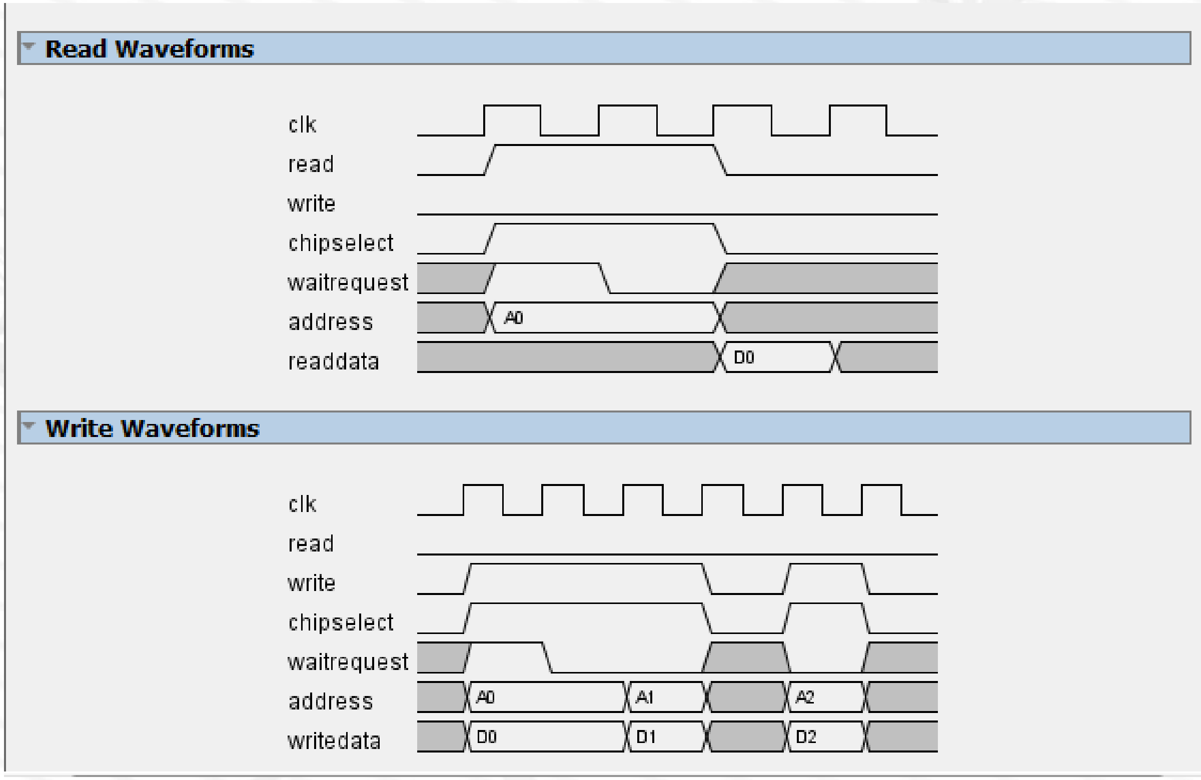


PS: KEY_N and IRQAck are not shown here, and some of the pin names maybe slightly different

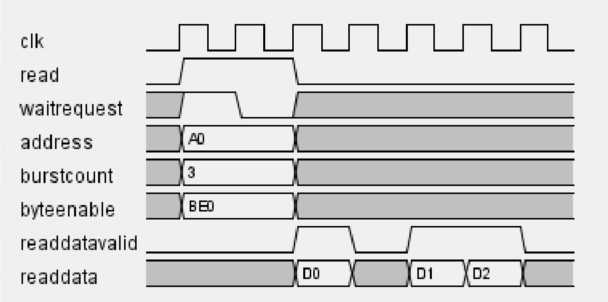
LCD control

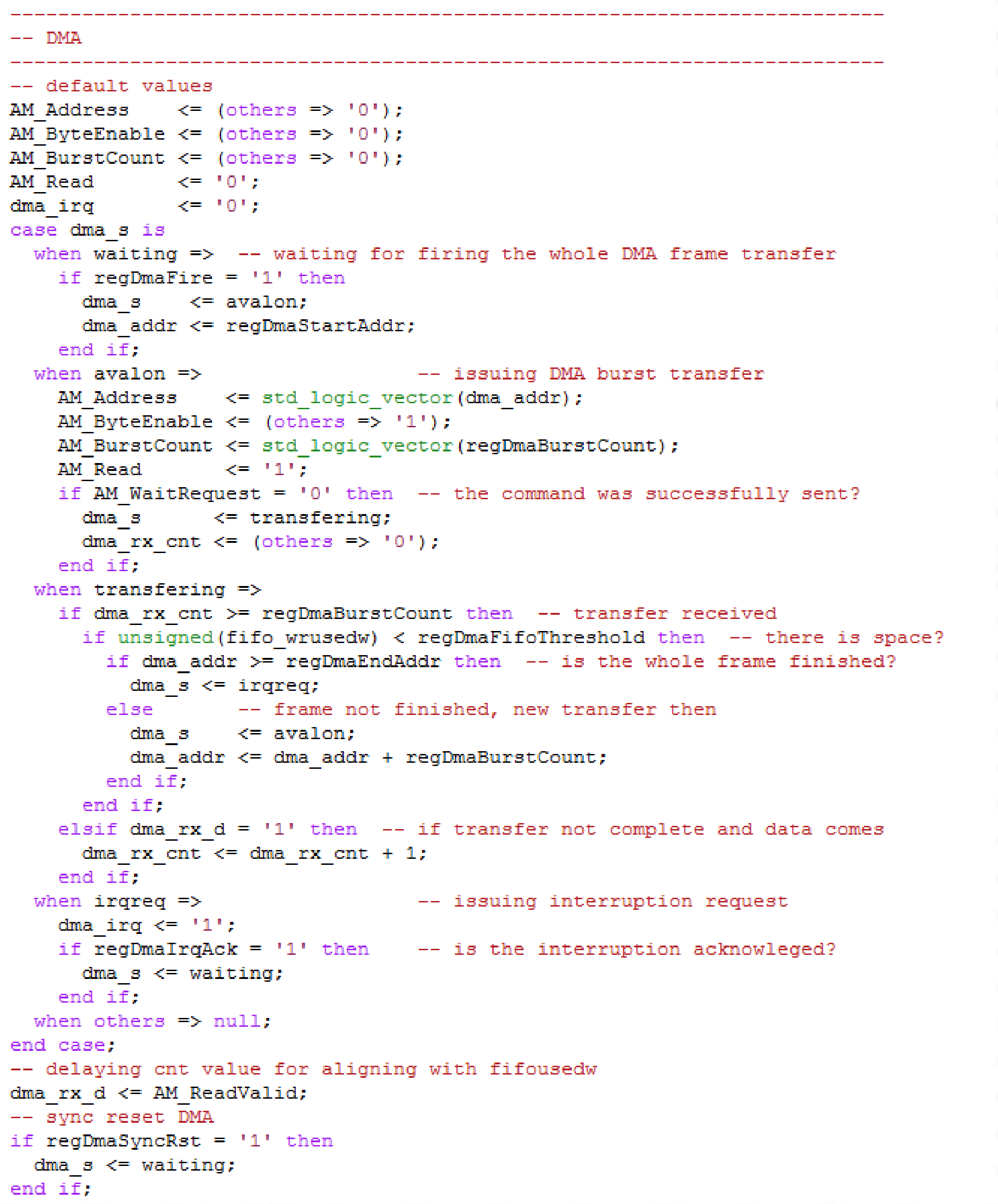

state~57
IDDLE
WRITE
READ
FINISH_W
EDGE_W
READ_L_WAIT
EDGE_R
WRITE_WAIT
FINISH_R

Avalon Slave Signals



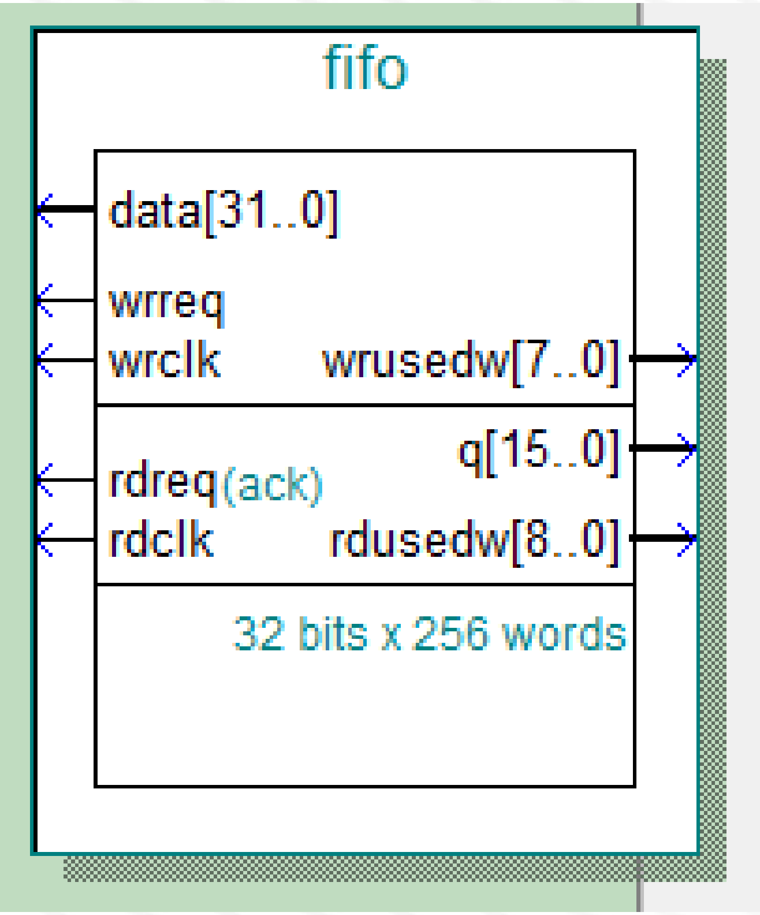
DMA engine





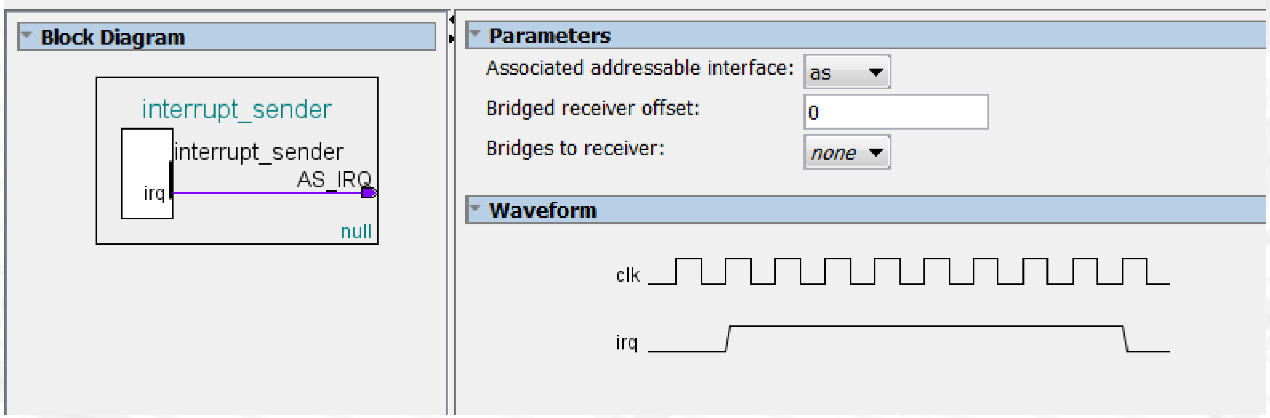
```
--------------------------------------------------------------------------
-- DMA
--------------------------------------------------------------------------
-- default values
AM_Address    <= (others => '0');
AM_ByteEnable <= (others => '0');
AM_BurstCount <= (others => '0');
AM_Read       <= '0';
dma_irq       <= '0';
case dma_s is
  when waiting =>  -- waiting for firing the whole DMA frame transfer
    if regDmaFire = '1' then
      dma_s    <= avalon;
      dma_addr <= regDmaStartAddr;
    end if;
  when avalon =>                  -- issuing DMA burst transfer
    AM_Address    <= std_logic_vector(dma_addr);
    AM_ByteEnable <= (others => '1');
    AM_BurstCount <= std_logic_vector(regDmaBurstCount);
    AM_Read       <= '1';
    if AM_WaitRequest = '0' then  -- the command was successfully sent?
      dma_s      <= transfering;
      dma_rx_cnt <= (others => '0');
    end if;
  when transfering =>
    if dma_rx_cnt >= regDmaBurstCount then  -- transfer received
      if unsigned(fifo_wrusedw) < regDmaFifoThreshold then  -- there is space?
        if dma_addr >= regDmaEndAddr then  -- is the whole frame finished?
          dma_s <= irqreq;
        else       -- frame not finished, new transfer then
          dma_s    <= avalon;
          dma_addr <= dma_addr + regDmaBurstCount;
        end if;
      end if;
    elsif dma_rx_d = '1' then  -- if transfer not complete and data comes
      dma_rx_cnt <= dma_rx_cnt + 1;
    end if;
  when irqreq =>                  -- issuing interruption request
    dma_irq <= '1';
    if regDmaIrqAck = '1' then    -- is the interruption acknowleged?
      dma_s <= waiting;
    end if;
  when others => null;
end case;
-- delaying cnt value for aligning with fifousedw
dma_rx_d <= AM_ReadValid;
-- sync reset DMA
if regDmaSyncRst = '1' then
  dma_s <= waiting;
end if;
```

Asymetric FIFO



- Quartus IP
- Asymetric FIFO
  - WIDTH IN = 32 bits
  - WIDTH OUT = 16 bits
- Show-ahead synchronous
- WR used word flag
- RD used word flag

IRQ interface



Block Diagram

Parameters

Associated addressable interface: as

Bridged receiver offset: 0

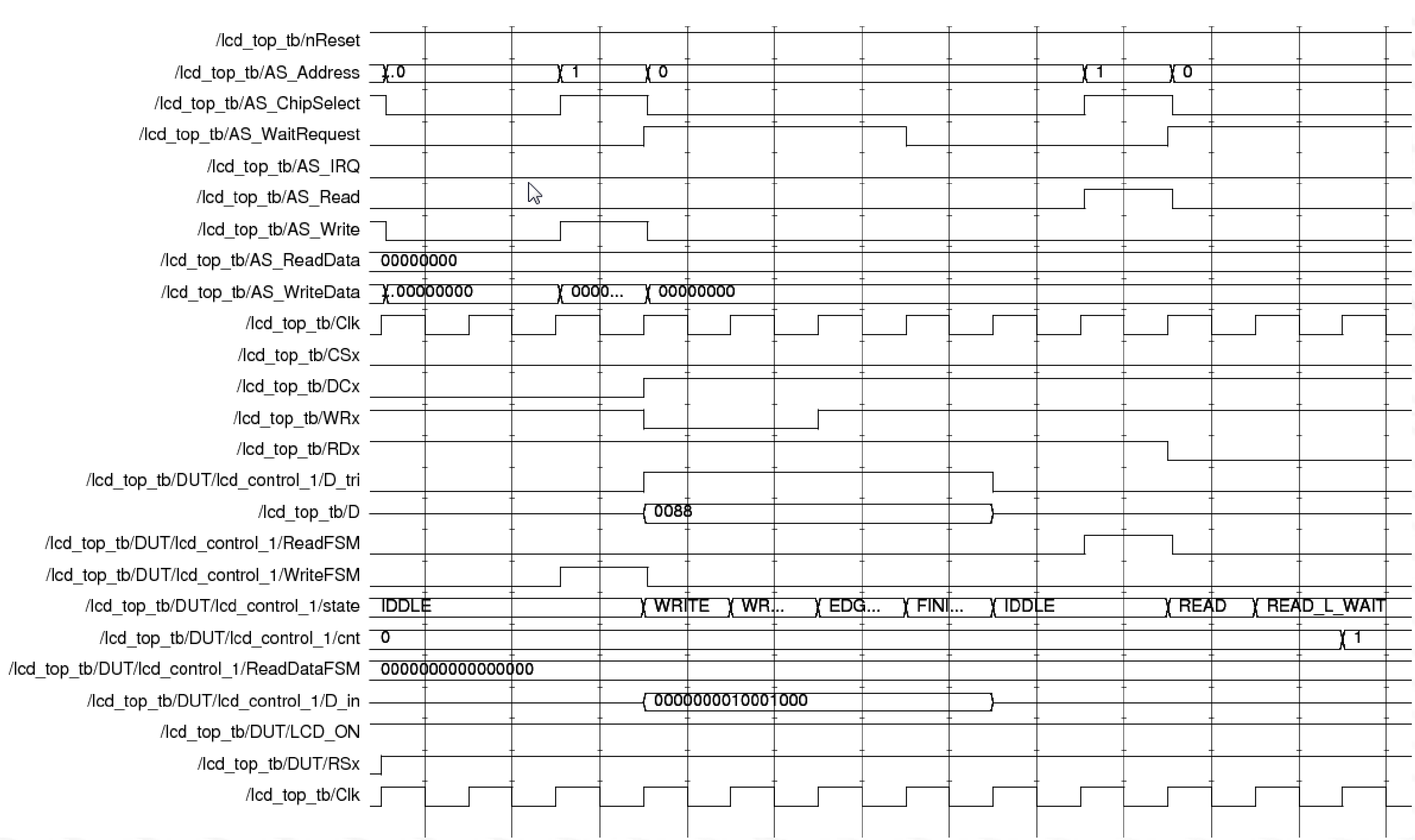
Bridges to receiver: *none*

Waveform

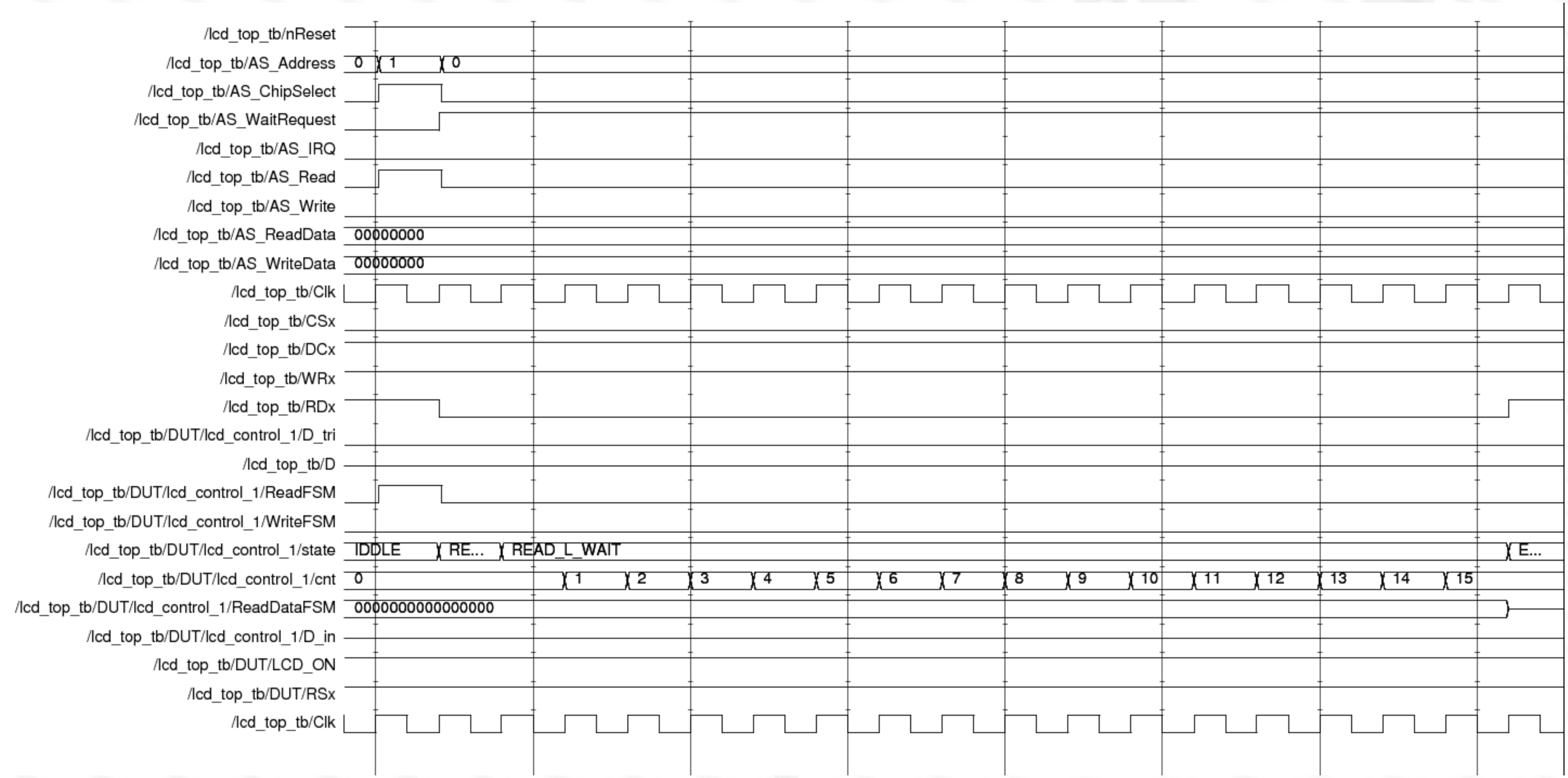
PS: According to Avalon Interface Specification, IRQ Acknowledge has to be implementation specific, that is implemented by the user.

0x06	0x18	RW	DmaIrqAck

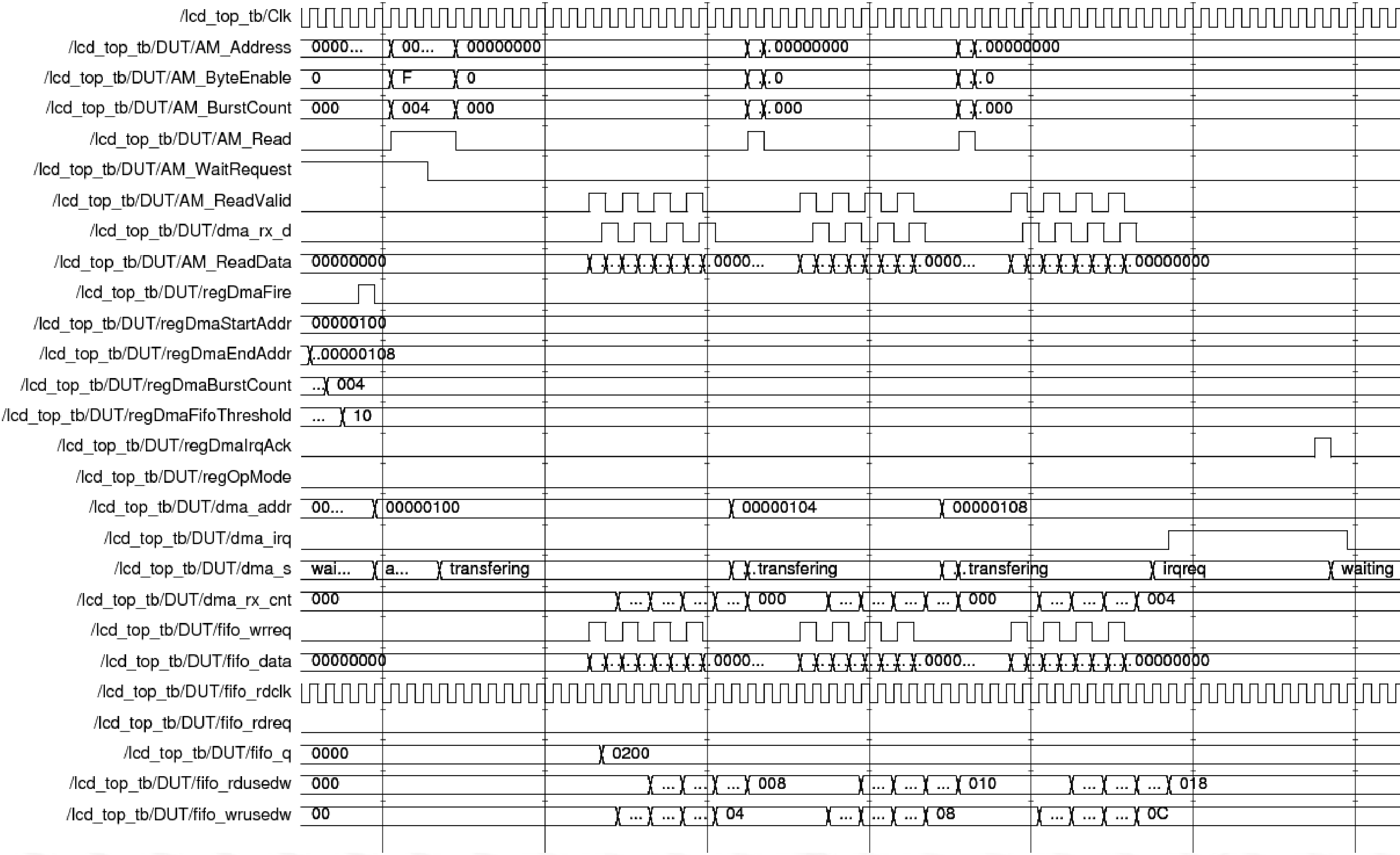
Modelsim Simulation Write Cycle



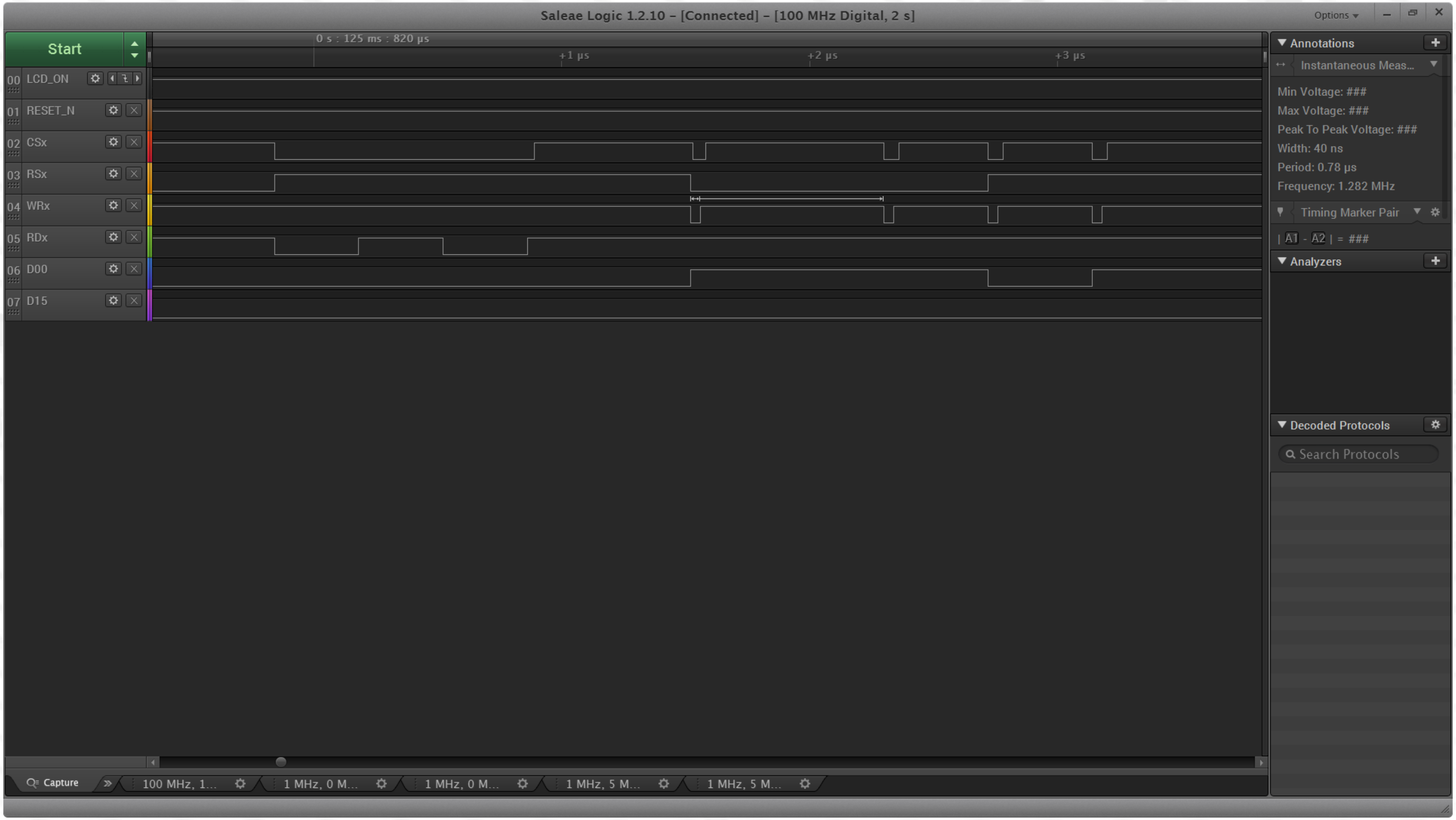
Modelsim Simulation Read Cycle



Modelsim Simulation DMA Engine


/lcd_top_tb/Clk
/lcd_top_tb/DUT/AM_Address 0000... 00... 00000000 00000000 00000000
/lcd_top_tb/DUT/AM_ByteEnable 0 F 0 0 0
/lcd_top_tb/DUT/AM_BurstCount 000 004 000 000 000
/lcd_top_tb/DUT/AM_Read
/lcd_top_tb/DUT/AM_WaitRequest
/lcd_top_tb/DUT/AM_ReadValid
/lcd_top_tb/DUT/dma_rx_d
/lcd_top_tb/DUT/AM_ReadData 00000000 0000... 0000... 00000000
/lcd_top_tb/DUT/regDmaFire
/lcd_top_tb/DUT/regDmaStartAddr 00000100
/lcd_top_tb/DUT/regDmaEndAddr 00000108
/lcd_top_tb/DUT/regDmaBurstCount ... 004
/lcd_top_tb/DUT/regDmaFifoThreshold ... 10
/lcd_top_tb/DUT/regDmaIrqAck
/lcd_top_tb/DUT/regOpMode
/lcd_top_tb/DUT/dma_addr 00... 00000100 00000104 00000108
/lcd_top_tb/DUT/dma_irq
/lcd_top_tb/DUT/dma_s wai... a... transfering transfering transfering irqreq waiting
/lcd_top_tb/DUT/dma_rx_cnt 000 ... 000 ... 000 ... 004
/lcd_top_tb/DUT/fifo_wrreq
/lcd_top_tb/DUT/fifo_data 00000000 0000... 0000... 00000000
/lcd_top_tb/DUT/fifo_rdclk
/lcd_top_tb/DUT/fifo_rdreq
/lcd_top_tb/DUT/fifo_q 0000 0200
/lcd_top_tb/DUT/fifo_rdusedw 000 ... 008 ... 010 ... 018
/lcd_top_tb/DUT/fifo_wrusedw 00 ... 04 ... 08 ... 0C

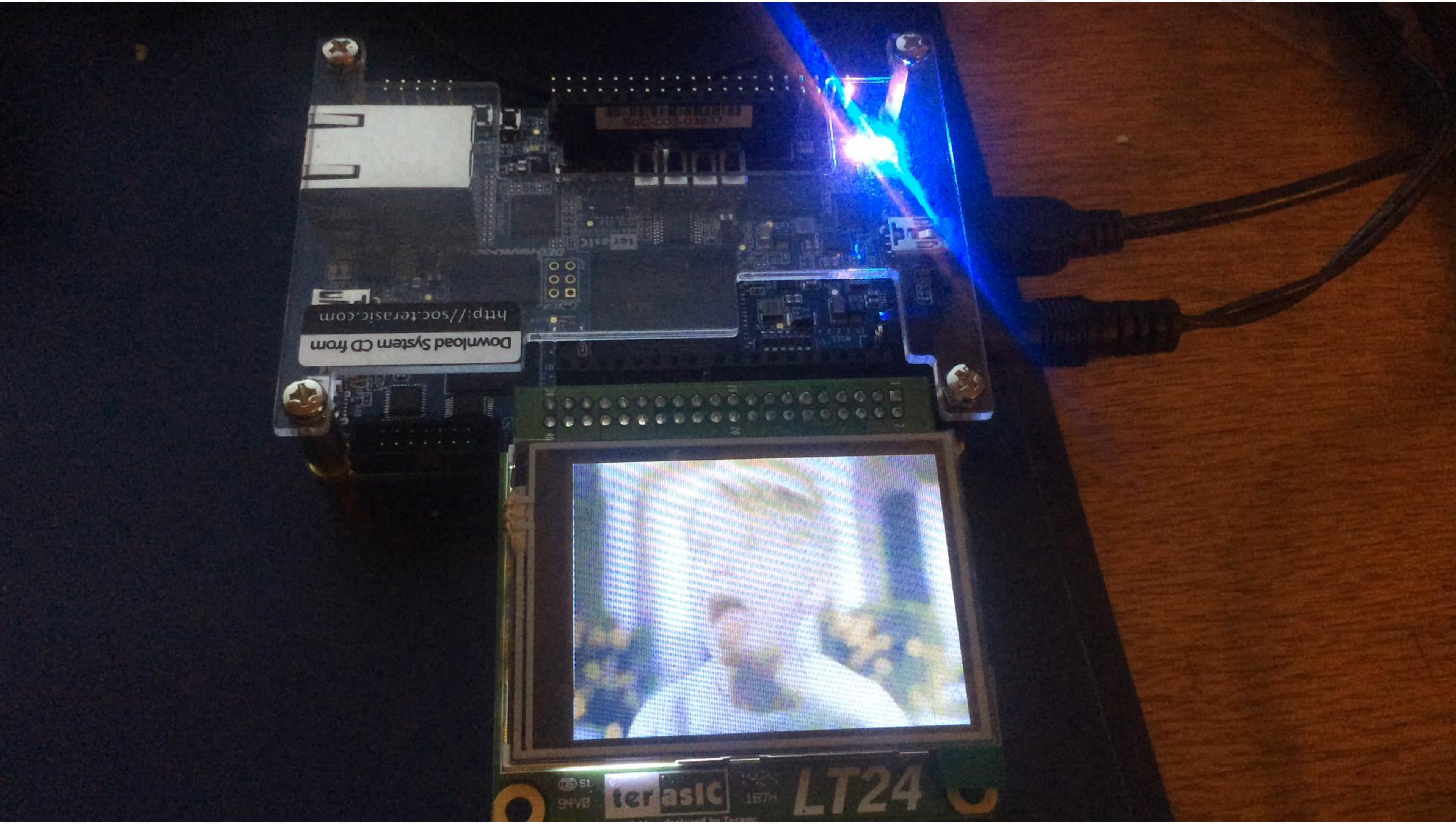
Logic Analyser



Min Voltage: ###
Max Voltage: ###
Peak To Peak Voltage: ###
Width: 40 ns
Period: 0.78 µs
Frequency: 1.282 MHz

| A1 - A2 | = ###

Demo



Demo