PLL CLOCKS

FREE RUNNING:

\[
\begin{align*}
\text{OSCILLATOR: } & 72.44 \text{ MHz} \\
\text{PCLK: } & 36.22 \text{ MHz} \\
\text{ECL: } & 18.11 \text{ MHz} \\
\text{EPCLK/2: } & 9.055 \text{ MHz} \\
\text{HWCLK: } & 4.53 \text{ MHz} \\
\text{WCLK: } & 2.26 \text{ MHz} \\
\end{align*}
\]

27.6 ns
55.2 ns
220.8 ns
441.6 ns

(TTL: PCLK/2)
(TTL: THWCLK)

(High max CLK freq: 2.5 MHz)
WAK
DISPM6
BWL
BLK
BLK
BLC
HVC
ER3H
ER2H
ER1H
U328-3

$E$ CHARGE DELAY TO REGEN REFRESH
$E$ CHARGE DELAY TO REGEN REFRESH
$E$ CHARGE DELAY TO REGEN REFRESH

4 x SCALE ENLARGEMENT

HWC

U328-3
U333-4

43 PIXEL DELAY

1 DELAY OFF
3 DELAYS ON
NLT 2 DELAYS

EALKING - 71 NM.
Left Edge Timing

Right Edge Timing

Extra delay to compensate for 1/4 pixel delay in XING SHIFT (FPA by pixel)
**Data Out (Processor Write)**

1. **REQ A**
2. **New Data Rdy**
3. **Data Out Pins**

**Notes:**
1. User asserts "REQA" telling CPU it can send more data.
2. As processor begins to write new data out, user clears the "REQA" signal.
3. On the falling edge of "New Data Rdy" the output data is valid. Data stays valid until the next CPU write or an init occurs.

**Data In (Processor Read)**

1. **Req B**
2. **Data Trans**
3. **Data In Pins**

**Notes:**
1. User asserts "Req B" telling CPU it has data ready.
2. User sets up data before "Data Trans" occurs.
3. On rising edge of "Data Trans" the user clears the data ready request.
4. The CPU has finished reading the data on the falling edge of "Data Trans". The input data pins may now change.

*Signal from user to DRV11*
DATA OUT - HOST WRITE

DATA OUT BUS

READY

BUSY

CYCLE REQ

F1

DATA VALID

DATA

STATE (Ready for DATA)

1. HOST sets F1 bit to 1, signalling a "DATA OUT" DMA operation
2. HOST sets "GO" bit in CSR, this says we can start DMA cycles
3. After READY is set, if the system is ready to accept data (STATE = 1) CYCLE REQ will be set.
4. When the HOST starts its DMA cycle, it sets BUSY. This causes CYCLE REQ to be cleared. Note that STATE is also cleared.
5. On the rising edge of BUSY, a word of data is latched into the interface.
6. When the Omega is ready for another word, STATE is set, which causes another CYCLE REQ to occur.

DATA IN - HOST READS

DATA IN BUS

READY

BUSY

CYCLE REQ

F1

STATE - DATA AVAIL

DATA VALID

DATA VALID

1. HOST sets F1 to a 1, we'll do data → host
2. HOST sets "GO" bit of CSR indicating that we can start DMA requests
3. When a word of data is available, STATE is set which causes CYCLE REQ to be set.
4. When BUSY is set, CYCLE REQ is cleared as is STATE
5. DMA cycle done, data is no longer needed, Omega can load next word
6. When the next word is available for the host to read, STATE and CYCLE REQ are set and another cycle can begin.

NOTE THAT CYCLE REQ is always held off for about a micro second after READY is set.
Red Controller Data from CPU/memroy Timing

Notes:
1) When computer interface has data for display controller, it latches the data into the cache high and initiates the transfer by a rising edge on IBATSTBH.

2) This causes "IBATACCH" to go low until the data is accepted by the interface. If multiple bytes are unpaired from a word without skipping "CLIDRHYH", this line won't go high until "CLIDRHYH" makes a positive transition.

3) "IBATRHYH" goes high when the display controller synchronizes the request internally.

4) When the micro-code wants to read the data onto the bus, "IBANREN" goes low to enable the micro-port buffer onto the data bus.

5) If this is the last byte available to be transferred, assert "CLIDRHYH" to terminate the transfer sequence; else another cycle sequence will start.

6) When "CLIDRHYH" goes high, the "IBATRHYH" signal from the display will be released as will be "IBATACCH".

* Signal to display controller
The output cycle starts with the falling edge of "OBAIDLEDENL".

Latch the data from the display when "OBAIDLEDENL" is low and a rising edge of "PROCCLK" occurs. A 74LS377 fits this need.

If the interface can only accept this byte now, assert "SODEBYH". This tells the display not to send any more data for now. If "SODEBYH" is held low, the display is free to send more data at its leisure.

If "SODEBYH" was asserted in step 3, the rising edge of "ODATSTBH" clears "ODATRBYH" informing the display that it may send another data byte to the interface.

* Signal to display controller