example project to interface to M6800-style peripheral

- OLED display control project:
  - https://code.google.com/p/idlab-general/source/browse/universal_eval/OLED_display/

- clock_enable_generator:
example interface: OLED display (M6800)

- page 5 of datasheet: shows four interface options
- page 10: shows signal descriptions

---

MPU Interface Pin Assignment Summary

<table>
<thead>
<tr>
<th>Bus Interface</th>
<th>Data/Command Interface</th>
<th>Control Signals</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>D7 D6 D5 D4 D3 D2 D1 D0</td>
<td>E R/W /CS D/C /RES</td>
</tr>
<tr>
<td>8-bit SPI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3-wire SPI</td>
<td>Tie LOW</td>
<td>Tie LOW /CS Tie LOW /RES</td>
</tr>
<tr>
<td>4-wire SPI</td>
<td>Tie LOW</td>
<td>Tie LOW /CS Tie LOW /RES</td>
</tr>
</tbody>
</table>

Wiring Diagrams

6800-MPU Parallel Interface
The parallel interface consists of 8 bi-directional data pins, R/W, D/C, E, and /CS. A LOW on R/W indicates write operation, and HIGH on R/W indicates read operation. A LOW on D/C indicates “Command” read or write, and HIGH on D/C indicates “Data” read or write. The E input serves as data latch signal, while /CS is LOW. Data is latched at the falling edge of E signal.

<table>
<thead>
<tr>
<th>Function</th>
<th>E</th>
<th>R/W</th>
<th>/CS</th>
<th>D/C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write Command</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Read Status</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Write Data</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Read Data</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
OLED display timing

- page 51 of display controller's datasheet
OLED timing diagram (original)

Figure 13-1: 6800-series MCU parallel interface characteristics
OLED timing diagram (ignore irrelevant parts)

Figure 13-1: 6800-series MCU parallel interface characteristics
OLED timing diagram (annotate with min/max constraints)

Figure 13-1: 6800-series MCU parallel interface characteristics

- **D/C#**
  - $t_{ss_{min}} = 10$ ns

- **R/W#(WR#)#**
  - $t_{sm_{min}} = 0$ ns

- **E(RD#)**

- **CS#**
  - $t_{tr_{max}} = 15$ ns
  - $t_{ds_{min}} = 40$ ns
  - $t_{sw_{min}} = 300$ ns
  - $t_{pw_{CSH_{min}}} = 60$ ns

- **D[7:0] (WRITE)**
  - Valid Data

- **D[7:0] (READ)**
  - Valid Data
OLED timing diagram
(make adjustments to simplify)

Figure 13-1: 6800-series MCU parallel interface characteristics

- D/C#: t_{AS min}=10
- R/W#: (WR#): t_{fr max}=15
- E(RD#)
- CS#: t_{fr max}=15
- D[7:0] (WRITE)
- D[7:0] (READ)

- PW_{CSL min}=60
- PW_{CSH min}=60
- t_{DIW min}=7
- t_{tACC}
- t_{DIIR}
- t_{OH}

2014-05-09 firmware bootcamp M. Andrew
OLED display control loop

if (transaction_required = '1') then
  if (individual_transaction_counter < 10) then
    transaction_in_progress <= '1';
    if (individual_transaction_counter < 1) then
      internal_sync <= not internal_sync;
      internal_enable <= '1';
    elseif (individual_transaction_counter < 4) then
      internal_chip_select <= '1';
      -- itc = 0, 1, 2, 3 (for 75 ns)
    else
      internal_chip_select <= '0';
      internal_enable <= '0';
    end if;
  else
    individual_transaction_counter <= individual_transaction_counter + 1;
  end if;
else
  transaction_required <= '0';
  transaction_in_progress <= '0';
  individual_transaction_counter <= (others => '0');
end if;

-- total 300 ns cycle time
-- (takes a cycle to get here from elsewhere)
-- itc = 0 (for 25 ns)
-- itc = 4, 5, 6, 7, 8, 9 (for 150 ns)
-- itc = 10 (for 25 ns)
testbench / simulation
scope traces
OLED timing diagram
(verify met all requirements)

Figure 13-1: 6800-series MCU parallel interface characteristics (ns)

- **D/C#:**
  - $t_{AS} = 25\, \text{ns}$
  - $t_{min} = 10\, \text{ns}$

- **R/W# (WR#):**

- **E(RD#):**
  - $t_{PD} = 300\, \text{ns}$
  - $t_{min} = 300\, \text{ns}$
  - $PW_{csh} = 75\, \text{ns}$
  - $PW_{csh} \geq 300\, \text{ns}$
  - $PW_{csh} \geq 60\, \text{ns}$

- **CS#:**
  - $t_{PD} = 60\, \text{ns}$

- **D[7:0] (WRITE):**
  - $t_{DIW} \geq 7\, \text{ns}$
  - $t_{DIW} \geq 40\, \text{ns}$

- **D[7:0] (READ):**
  - $t_{ACC}$
  - $t_{DHR}$
  - $t_{OH}$

Valid Data
advice

• use constants where possible:
  – to reduce instances of “magic numbers:”
    • constant mywidth : integer := integer(ceil(log2(1300.0)));
  – for future expandability:
    • signal mycounter : unsigned(mywidth-1 downto 0);

• better yet, use generics!

• don't be intimidated by the flurry of messages that go by as it compiles
  – pay attention to the warnings that are your fault; these are clues as to what you are doing wrong
warning messages

- Net Skew is the difference between the minimum and maximum routing delays for the net. Note this is different from Clock Skew which is reported in TRCE timing report. Clock Skew is the difference between the minimum and maximum path delays which includes logic delays.
- The fanout is the number of component pins not the individual I/O loads.

Timing Score: 394773 (Setup: 394773, Hold: 0, Component Switching Limit: 0)

WARNING: Par: 465 - Your design did not meet timing. The following are some suggestions to assist you to meet timing in your design.

- Review the timing report using Timing Analyzer (In ISE select “Post-Place & Route Static Timing Report”). Go to the failing constraint(s) and evaluate the failing paths for each constraint.
- Try the Design Goal and Strategies for Timing Performance (In ISE select Project -> Design Goals & Strategies) to ensure the best options are set in the tools for timing closure.
- Use the Xilinx “SmartPlace” script to try special combinations of options known to produce very good results.
- Visit the Xilinx technical support web at http://support.xilinx.com and go to either “Troubleshoot-Tech Tips->Timing & Constraints” or “TechXticles->Timing Closure” for tips and suggestions for meeting timing in your design.

Asterisk (*) preceding a constraint indicates it was not met.
- This may be due to a setup or hold violation.

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Check</th>
<th>Worst Case</th>
<th>Best Case</th>
<th>Timing</th>
<th>Timing</th>
</tr>
</thead>
<tbody>
<tr>
<td>* ts_clock_150 = PERIODIC TIMEGRP “clock_150”</td>
<td>SETUP</td>
<td>-7.486ns</td>
<td>14.155ns</td>
<td>108</td>
<td>584773</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.667 ns HIGH 50%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Check</th>
<th>Worst Case</th>
<th>Best Case</th>
<th>Timing</th>
<th>Timing</th>
</tr>
</thead>
<tbody>
<tr>
<td>* ts_clock_480 = PERIODIC TIMEGRP “clock_480”</td>
<td>SETUP</td>
<td>6.394ns</td>
<td>16.696ns</td>
<td>287</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 ns HIGH 50%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

All constraints were met.

Generating Pad Report.
- All signals are completely routed.

Total REAL time to PAR completion: 15 secs

WARNING: Par: 282 - There are 1 loadless signals in this design. This design will cause Bitgen to issue DRC warnings.

2014-05-09 firmware bootcamp M. Andrew
more advice

- don't use up all the global clocks; use a single clock and distribute clock enables with the appropriate duty cycles
- read through coding recommendations from Xilinx (and follow them!):
- if possible, use someone else's *working* code
- build things one at a time and **test them**!
parting words...

• A complex system that works is invariably found to have evolved from a simple system that worked. A complex system designed from scratch never works and cannot be patched up to make it work. You have to start over with a working simple system. – John Gall (1975)