MAX10 Simple Power Sequencer

John Dillon and Mark Frost, Intel PSG, March 2016

Overview
Sequencing Arria 10 on power down requires that $\text{Vccio} - \text{Vcckt} > 1.92\text{ volts}$. This condition is difficult to achieve unless active discharge of the regulators outputs is implemented due to capacitance on the outputs of the regulators. There are also timing restraints on how fast Arria 10 powers up and down as shown in the figure below.

A relatively simple scheme for implementing Arria 10 power up/down sequencing has been developed using a low-cost MAX 10 FPGA. The figure below shows an overview of the scheme.
**Operation Power Up**

- On power up measure the input supply and ensure its within tolerance
- Enable the input current limiter (ES1010)
- Enable the first regulator (Core) and wait for Power Good on the first regulator
- When Power Good is valid for the first regulator Enable the second regulator (Transceivers)
- Repeat for the required number of regulators continuously monitor the input supply for under/over voltage conditions

**Operation Power Down Sudden Input Power Disconnection (Worse Case Condition)**

- Measure the input power supply if the supply isn’t within regulation power down
- Disable the Enable on the required power supply
- Wait for Power Good to be not valid
- Enable the active discharge FET
- Disable the next regulator and repeat the sequence
Testing
To test the operation a MAX10 demonstration board (10M08) was used on an Enpirion power supply board and wired up as per the above diagram (Figure 1)

The board was tested for power up and power down sequencing.
Power UP

4 supplies powered to 1.2 volts over 60ms; the 20ms delay is due to the soft start on each of the regulators.
Power Down

All 4 power supplies shut down in under 400us; the supplies don’t have to be shut down this quickly but if you do shut down quickly it saves on input supply capacitors (hold up capacitors) in case of a sudden power removal.

The 4 power supplies have no load (so worse case).
Power Down Under Voltage Lock Out

In this case the input 12 volt supply was removed (the blue trace) and the scope was triggered from one of the power supply outputs, none of the power supplies are under load but if you observe the 12 volt supply because we power down so quickly there’s little drop over time on the 12 volt rail.

The 12 volts would drop more quickly if full load was on the all of the outputs but because we actively discharge the outputs we control the discharge time and save on hold up capacitors.
Capacitor Discharge Circuit Considerations for FPGAs
To discharge the capacitor bank, a N-channel Power MOSFET is chosen to have an on-resistance (RDS(on)) suitable to discharge the largest capacitance bank, within 10ms to ensure full shutdown sequence of discharging 10 channels occurs in less than 100ms. An auxiliary power supply must be provided to drive the shutdown circuit for at least 100ms after power down has been activated. This would be sourced from the input power supply and would probably be the MAX10 power supply.

Time to discharge calculation
Using a 3 X RC time constant where R is the combined resistance of the ESR of the capacitor bank (assumed to be 10mΩ), the parasitic trace resistance (assumed to be 50mΩ) and the Power MOSFET RDS(on). Also included is the temperature dependency of the Power MOSFET on-resistance that can be as high as 1.5 X the maximum RDS(on) at Tamb = 25˚C, which is assuming the junction has reached the absolute maximum temperature rating (typical TJ(max) = 150°C).

If for instance the core had an output capacitance of 15mF and the discharge time needed was 10ms a 3 times RC constant is required = 3 * ((60m ohms +(1.5 * RDS (on)) * 15MF = 8 m/seconds

Therefore the power mosfet needs an RDS (on) of less than 80 m/ohms at Vgs = 4.5V at an ambient temperature of 25C

Safe Operating Area and Transient Thermal Stress
The Power MOSFET is chosen to have a power rating capable of handling the transient power dissipation of the discharge current. Peak power is calculated via simulation and the value checked against the transient power capability graphs of the Power MOSFET datasheet. As the Power MOSFET will be dissipating the capacitor’s energy as a function of both current and voltage over time then the Safe Operating Area (SOA) curve in the datasheet needs to be reviewed. This will give the maximum single pulse that the Power MOSFET can safely handle whilst ensuring the junction temperature does not exceed the absolute maximum rating, typical TJ(max) = 150°C. An SOA should be based on the application’s ambient operating temperature with the required MOSFET gate drive. In the case of discharging the 0.9V charged capacitor bank, then review the SOA curve for single pulse peak current capability at 1V between 1ms and 10ms pulse width curves. The SOA should be for a typical application ambient temperature, which is assumed to 60°C.

The peak surge current from the capacitor bank needs to be measured in the actual circuit to ensure that sufficient parasitic resistance is slowing down the response to avoid sharp rising current peaks that could cause EMI issues and also transient thermal stress on both the Power MOSFET and capacitor bank. It is assumed that the parasitic trace resistance is about 50mΩ. If significantly less, then a series resistor needs adding to the drain of the power mosfet.

A suitable manufacture for the power mosfet is the Diodes Inc DMN3027LFG

http://www.diodes.com/catalog/N_Channel_30V_23/DMN3027LFG_11364
This mosfet is in a 3mm * 3mm Powerdi package and is 100% Avalanche rated

**Demonstration Board Wiring Diagram**

The Max10 board schematics used for the demonstration is called the
Altera_10M08_E144_schmatic_REV_1.pdf

The sequencing I/O is wired from connector J8 on the MAX10 board the silk screen is used as the wiring reference rather than the schematics

<table>
<thead>
<tr>
<th>Silk Screen J8</th>
<th>Power Supplies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pin 41 wire to</td>
<td>Enable power supply 1</td>
</tr>
<tr>
<td>Pin 43 wire to</td>
<td>POK power supply 1</td>
</tr>
<tr>
<td>Pin 38 wire to</td>
<td>Discharge FET power supply 1</td>
</tr>
<tr>
<td>Pin 46 wire to</td>
<td>Enable power supply 2</td>
</tr>
<tr>
<td>Pin 47 wire to</td>
<td>POK power supply 2</td>
</tr>
<tr>
<td>Pin 44 wire to</td>
<td>Discharge FET power supply 2</td>
</tr>
<tr>
<td>Pin 55 wire to</td>
<td>Enable power supply 3</td>
</tr>
<tr>
<td>Pin 56 wire to</td>
<td>POK power supply 3</td>
</tr>
<tr>
<td>Pin 59 wire to</td>
<td>Enable power supply 4</td>
</tr>
<tr>
<td>Pin 60 wire to</td>
<td>POK power supply 4</td>
</tr>
<tr>
<td>Pin 57 wire to</td>
<td>Discharge power supply 4</td>
</tr>
</tbody>
</table>

For the ADC channel s Connector J4 is used (note only one ADC is currently used to measure the input supply)

J4 pin 1 wire to Input supply use a resister divider to ensure the measure voltage is below 3 volts

J4 pin 2 wire to Output of Power supply 1

J4 pin 3 wire to Output of Power Supply 2

J4 pin 4 wire to Output of Power Supply 3

J4 pin 5 wire to Output of Power Supply 4

When measuring on the ADC channel s ensure the measured voltages are below the maximum ADC inputs
Design Considerations

The design is meant as a starting point for sequencing power supplies and the only verification performed is on the demonstration board. Customers should ensure that the sequencing performed is correct for their design and conforms to the sequencing in the Arria 10 pin connection guidelines.

The Max10 power supply should be designed to provide power for the time taken to discharge all of the Arria 10 power supplies this may require additional hold up capacitors on the output of the power supply that provides power to the MAX10.

The Arria 10 power rails aren’t measured on the demonstration board with the ADC channels but they could be used to prevent pre bais on start up. The discharge fets could be used to prevent pre bais on start up.

The core and transceiver rails are usually the supplies with the most current capability and will put most stress on the discharge fets, because we software control the sequencing these rails when shut down could be allowed to discharge gracefully (therefore preventing the stress on the discharge fets) if required on start-up we can measure the rails to prevent pre bais and discharge if required.
**MAX10 The Future**

Max 10 can be used for much more than sequencing a few ideas are presented below

- **Remote sense**

  A PWM signal supplying a low pass filter will produce a DC output because we can measure the output of the power supplies and we can produce a DC output we can implement remote sense of the power supply by summing the DC into the feedback path of the relevant power supply

- **SMB Bus**

  The Max 10 could be used as a PM bus master to control SM bus regulators and for smart VID

- **Voltage/Current and Temperature Measurement**

  With the ADC’s we can measure voltage/current and temperature, under voltage over voltage, over temperature, over current and window comparators can easily be implemented

- **Sequencing**

  Max 10 is well suited to sequencing operations

- **Telematics**

  A web server could be implemented on the Max10 for Telematics

- **Programming**

  Max10 could be used to program the main system FPGA

**FPGA Design**

The FPGA design was targeted at the MAX10 10M08 mainly because of the wide availability of the low cost MAX10 Evaluation Board ([https://www.altera.com/products/boards_and_kits/dev-kits/altera/kit-max-10-evaluation.html](https://www.altera.com/products/boards_and_kits/dev-kits/altera/kit-max-10-evaluation.html)). However the ultimate aim was to fit this comfortably within a 10M04 device for low cost application in a broad range of power sequencing designs.

The design is primarily based around the NIOS II softcore processor to allow for rapid prototyping and changes to the design. The three figures below show a block diagram of the design and flowcharts describing the code flow; the design itself is simple and self-explanatory.
ADC
NIOSII
Parallel IO
JTAG UART
On-chip RAM
On Board Flash memory
Parallel IO
Analogue inputs
IO from/to power electronics
Host PC
Analogue inputs
IO from/to power electronics
ADC
NIOSII
Parallel IO
JTAG UART
On-chip RAM
On Board Flash memory
Parallel IO
Analogue inputs
IO from/to power electronics
Host PC
POWER UP SEQUENCE

1. Reset
2. Wait for PowerUP button
3. Enable Power Group #1
   - POK?
   - No
4. Enable Power Group #2
   - POK?
   - No
5. Enable Power Group #3
   - POK?
   - No
6. Enable Power Group #4
   - POK?
   - No
7. VIN OK still? POKs?
   - Yes
   - No
8. User power down?
   - Yes
   - No

Move to Power down sequence
POWER DOWN SEQUENCE

Wait until commanded to enter powerdown

Send Power Off command to Power Group #4
Set Power Group #4
Discharge line
Wait

Send Power Off command to Power Group #3
Set Power Group #3
Discharge line
Wait

Send Power Off command to Power Group #2
Set Power Group #2
Discharge line
Wait

Send Power Off command to Power Group #1
Set Power Group #1
Discharge line
Wait

Jump back to Power Up Sequence

How to modify and rebuild the design

The attached design files were built using Quartus Prime Standard Edition, v15.1. Tools required:

- Quartus Prime Lite or Standard Edition (any version that supports MAX10)


The recommended approach for rebuilding the example design to create a .POF for storage in the MAX10 CFM (Configuration Flash Memory) is detailed below:
1) Open Quartus and load up the project file (PowerSeq1.qpf). Open the top level hierarchy and PowerSeq1.vhd will open. This is the top-level of the design, and really only contains two components, plus the ports and mappings to the outside world:

a. **POWER1**: QSys generated component which contains the NIOS II sub system

b. **LED_FLASH**: A simple LED flashing component that drives LED D1 on the evaluation kit

2) Fire up QSys and open Power1.qsys, you should see the following system:

   - **ALTPLL_0**: Takes the 50MHz onboard clock and generates an 80MHz system clock
   - **CUSTOM_RESET_SYNC_0**: Provides a system wide synchronous reset using the PLL locked output to gate when to take the rest of the system out of reset
   - **NIOS_RAM**: 16KB of on-chip RAM. The NIOSII processor boots and executes solely from this on-chip RAM. The RAM is initialised with the file power1_NIOS_RAM.hex which
contains the application code for the NIOS. This .hex file is generated by the Eclipse tools (see the software section later on)

d. **JTAG_UART**: UART debug component for providing host PC access to the debug messages generated by the NIOSII processor

e. **TIMER**: Required by the NIOSII processor

f. **NIOS**: Economy model of the NIOSII processor with standard settings

g. **NIOS_IO**: Parallel IO interface from the NIOSII processor, which is exported to the FPGA fabric as test_pio_output. This provides all of the IO accesses the NIOSII application code requires.

h. **ADC**: Modular ADC IP core set in Standard sequencer with Avalon-mm sample storage mode. The sequencer is set to use five ADC inputs and the onboard temperature sensing diode.

3) You can regenerate the QSys component from the Generate menu:

![Image of QSys generation interface]
This will create the required components for inclusion into the top-level VHDL design (power1/power1_inst.vhd) plus the .SOPCINFO file (power1.sopcinfo) which the Eclipse tools require to understand the device tree and memory map of the NIOSII processor.

4) From within QSys, open up the Eclipse build tools (Tools, NIOS II Software Build Tools for Eclipse) and set the workspace to point to the software directory in the example design (the first time you do this you may need to import the application and the BSP into Eclipse)

5) There are two main files associated with the NIOS application, main.c (main code loop) and power_inc.h (include file with various constants). Firstly rebuild the BSP by right-clicking on power1_bsp and selection NIOS/Generate BSP. Then rebuild the application by selecting the power1 application and Build Project. This will generate the NIOS executable, power1.elf.

6) Finally we need a new .hex file to allow the on-chip RAM to be initialised with the NIOS application code, and this can also be generated from within Eclipse. Select the power1 application and select Make Targets, Build:

![Make Targets](image)

Select mem_init_generate and hit Build. This will create the file power1_NIOS_RAM.hex that Quartus will incorporate into the on-chip RAM.

7) The final stage of the build process is to do a complete compile from within Quartus. The compile should complete in a few minutes and Quartus will generate the required .sof and .pof files. The resource summary should look like the diagram below (and you can see that this design will easily fit into the smaller MAX10 devices)
8) Use the Quartus programmer to load the .POF into the CFM on the MAX10 device.
**UART display**

The NIOS II processor generates a stream of messages to show what is happening. These can be accessed via the JTAG UART and USB Blaster connected to a host PC.

To display the UART data you can use with the Eclipse tools provided with the NIOS EDS (Embedded Design Suite, or for simplicity use the NIOS2-TERMINAL command line application: launch the NIOS2 command shell:

And run the terminal application with **nios2-terminal**:

```
$ nios2-terminal
nios2-terminal: connected to hardware target using JTAG UART on cable
nios2-terminal: "USB-Blaster [USB-0]", device 1, instance 0
nios2-terminal: (Use the IDE stop button or Ctrl-C to terminate)

MAX10 Power Sequencing example design
****** Wait for user input to begin power-up sequence
```
Future Work & Ideas

- Create a System Console GUI to allow more user interaction (e.g. change the order of the power sequencing, vary the delays etc)

- Add in I2C/PMbus to configure and control the power components in greater detail

- Add in access to the user flash memory (UFM) to store user parameters

- Add watchdog and other board monitor features

- Remove the NIOS soft core processor and recode the state machines in RTL for an optimum size and performance design