Here's the file structure and initial VHDL code implementation for the battery system design and signal control system, based on the specified requirements. This setup supports efficient management and configuration within an FPGA development environment.

## 1. File Structure

1. `battery_control.vhd`: Contains the VHDL code for monitoring and managing battery functions.
2. `signal_decoder.vhd`: VHDL code for a multiplexer or decoder to handle input and output signals based on input conditions.
3. `constraints.xdc`: Constraints file for the Artix-7 FPGA, mapping VHDL code signals to specific physical pins.

## 2. VHDL Code Implementation

---

### `battery_control.vhd`

This file includes the VHDL code for the battery control module, covering current monitoring, temperature sensing, and voltage regulation.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity battery_control is
    Port (
        clk           : in  STD_LOGIC;           -- Clock signal
        reset         : in  STD_LOGIC;           -- Reset signal
        current_level : in  STD_LOGIC_VECTOR(7 downto 0); -- Input
for current level monitoring
        temp_sensor   : in  STD_LOGIC_VECTOR(7 downto 0); -- Input
for temperature sensor
        voltage_level : in  STD_LOGIC_VECTOR(7 downto 0); -- Input
for voltage level monitoring
        warning       : out STD_LOGIC;           -- Output warning
signal if levels exceed limits
        control_sig   : out STD_LOGIC            -- Control signal
for battery management
    );
end battery_control;
```

```
architecture Behavioral of battery_control is
    signal temp_warning : STD_LOGIC := '0';
    signal current_warning : STD_LOGIC := '0';
    signal voltage_warning : STD_LOGIC := '0';
begin
    process(clk, reset)
    begin
        if reset = '1' then
            warning <= '0';
            control_sig <= '0';
        elsif rising_edge(clk) then
            -- Temperature check
            if temp_sensor > "10010010" then  -- Example threshold
for temp
                temp_warning <= '1';
            else
                temp_warning <= '0';
            end if;

            -- Current check
            if current_level > "01100100" then  -- Example
threshold for current
                current_warning <= '1';
            else
                current_warning <= '0';
            end if;

            -- Voltage check
            if voltage_level < "00101010" then  -- Example
threshold for low voltage
                voltage_warning <= '1';
            else
                voltage_warning <= '0';
            end if;

            -- Generate warnings and control signal
            warning <= temp_warning or current_warning or
voltage_warning;
            if warning = '1' then
                control_sig <= '1';
            else
                control_sig <= '0';
            end if;
        end if;
    end process;
end Behavioral;
```

### signal_decoder.vhd

This file includes the VHDL code for a decoder or multiplexer that manages signal routing within the spacecraft system.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity signal_decoder is
    Port (
        sel      : in  STD_LOGIC_VECTOR(1 downto 0); -- Selection
input for decoding
        input_0  : in  STD_LOGIC;                    -- Input
signal 0
        input_1  : in  STD_LOGIC;                    -- Input
signal 1
        input_2  : in  STD_LOGIC;                    -- Input
signal 2
        input_3  : in  STD_LOGIC;                    -- Input
signal 3
        output   : out STD_LOGIC                     -- Routed
output based on selection
    );
end signal_decoder;

architecture Behavioral of signal_decoder is
begin
    process(sel, input_0, input_1, input_2, input_3)
    begin
        case sel is
            when "00" =>
                output <= input_0;
            when "01" =>
                output <= input_1;
            when "10" =>
                output <= input_2;
            when "11" =>
                output <= input_3;
            when others =>
                output <= '0';
        end case;
    end process;
end Behavioral;
```

### constraints.xdc

This .xdc file is for mapping VHDL signals to the physical pins of the Artix-7 FPGA. Adjust the pin assignments according to the specific FPGA board being used.

```
# Clock signal
set_property PACKAGE_PIN W5 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports clk]

# Reset signal
set_property PACKAGE_PIN U16 [get_ports reset]
set_property IOSTANDARD LVCMOS33 [get_ports reset]

# Battery control signals
set_property PACKAGE_PIN V4 [get_ports current_level[0]]
# Continue defining pins for the current_level vector
# Repeat for temp_sensor and voltage_level pins...

# Warning and control signal outputs
set_property PACKAGE_PIN Y18 [get_ports warning]
set_property PACKAGE_PIN W17 [get_ports control_sig]

# Signal decoder/multiplexer pins
set_property PACKAGE_PIN J15 [get_ports sel[0]]
set_property PACKAGE_PIN K15 [get_ports sel[1]]
set_property PACKAGE_PIN L16 [get_ports input_0]
```

**Example VHDL File (top_level.vhd)**

This example creates a simple entity that you can modify according to design requirements. Here, we use only a few pins to illustrate.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity top_level is
    Port (
        clk   : in  std_logic;    -- Clock input
        rst   : in  std_logic;    -- Reset input
        led   : out std_logic     -- LED output
    );
end top_level;

architecture Behavioral of top_level is
    signal counter : integer := 0;
begin
    process(clk, rst)
    begin
```

```
        if rst = '1' then
            counter <= 0;
            led <= '0';
        elsif rising_edge(clk) then
            counter <= counter + 1;
            if counter = 50000000 then
                led <= not led;
                counter <= 0;
            end if;
        end if;
    end process;
end Behavioral;
```

**Constraints File (top_level.xdc)**

Here, some of the pins are assigned according to the names and configurations provided. Configuration pins (e.g., TCK, TMS) do not need to be assigned in the constraints file unless the design explicitly requires them. This file is adapted for general-purpose input/output (GPIO) pins:

```
## XDC file configuration for xc7a100tcsg324 device

# Clock pin assignment
set_property PACKAGE_PIN K17 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports clk]

# Reset pin assignment
set_property PACKAGE_PIN K18 [get_ports rst]
set_property IOSTANDARD LVCMOS33 [get_ports rst]

# LED pin assignment
set_property PACKAGE_PIN L14 [get_ports led]
set_property IOSTANDARD LVCMOS33 [get_ports led]
```

**Instructions for Uploading to GitHub**

1. **Repository Structure**: Create a folder structure for the project on GitHub as follows:

```
/vhdl_project
├── src
│   └── top_level.vhd
└── constraints
    └── top_level.xdc
```

**Push to GitHub**: Push the project to your GitHub account.

```
git remote add origin
https://github.com/your_username/vhdl_project.git
git push -u origin master
```