Lattice USB Type-C Solution Design Document

October 2016

Reference Design RD1210

Introduction

This design document describes the implementation details of the following Lattice FPGA based USB Type-C solutions:

- CD/PD for Hosts/Devices – Dual Role Port (DRP)
- CD/PD for Charger – Source Only

Detailed block diagrams of both solutions and RTL hierarchy information are provided in this document. Specific information on RTL customization in terms of supported Power Profiles and the VBUS voltage control unit are also provided. The goal is to guide users in modifying the RTL to support desired Power Profiles and PMIC interface.

The CD/PD for Hosts/Devices - Dual Role Port (DRP) solution is the superset solution and the CD/PD for Charger - Source Only solution is the subset of the DRP solution. Implementation details are focused on the superset DRP solution.

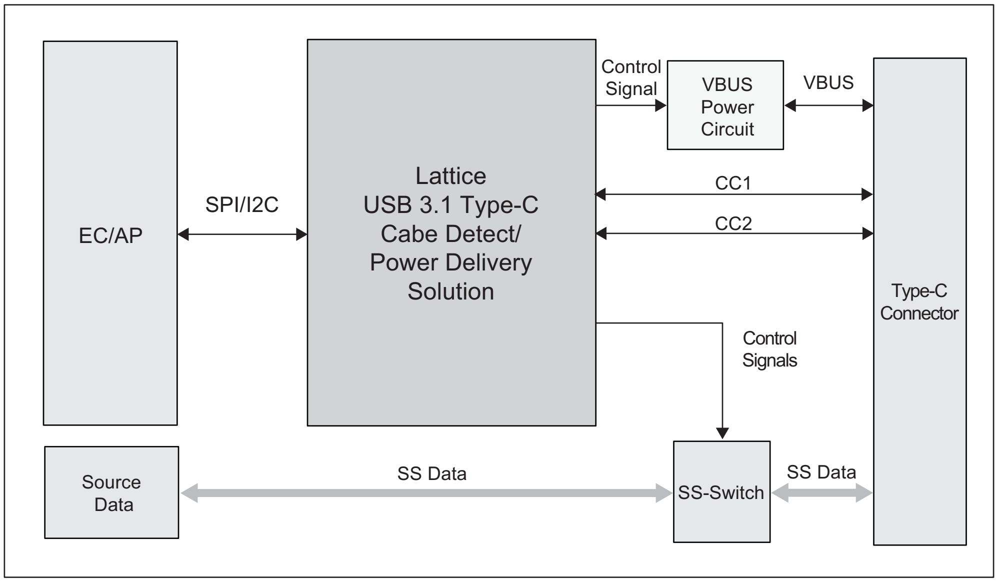
The solutions are implemented based on the Universal Serial Bus Power Delivery Specification revision 2.0, version 1.1 and the Universal Serial Bus Type-C Cable and Connector Specification revision 1.1.

For more information on schematics and compliance results for the solutions, refer to DS1052, Lattice USB Type-C Solution Data Sheet and RD1209, Lattice USB Type-C Solution.

System Block Diagram

Figure 1 shows the system level block diagram of the CD/PD for Hosts/Devices - Dual Role Port (DRP) solution. The Application Processor (AP) or Embedded Controller communicates with the solution over I2C serial link. The Lattice USB Type-C solutions are designed to provide bidirectional configuration channel signals (CC1/CC2) to the Type-C connector. The solution also provides VBUS Control Unit signal and Super- Speed (SS) switch control signals.

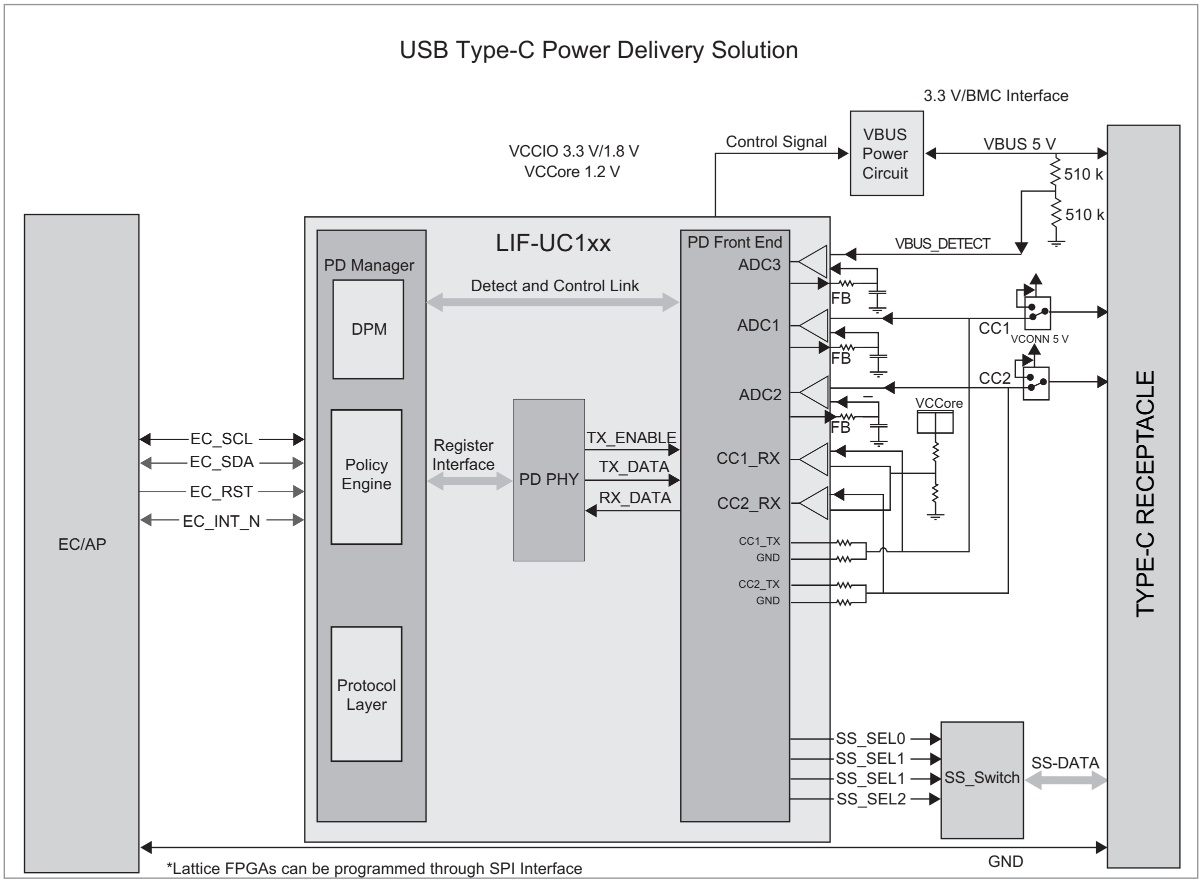
Figure 1. System Level Block Diagram



System Level Functional Block Diagram

Figure 2 shows the system level functional block diagram. It consists of the PD manager, PD PHY and PD Front End.

Figure 2. System Level Functional Block Diagram



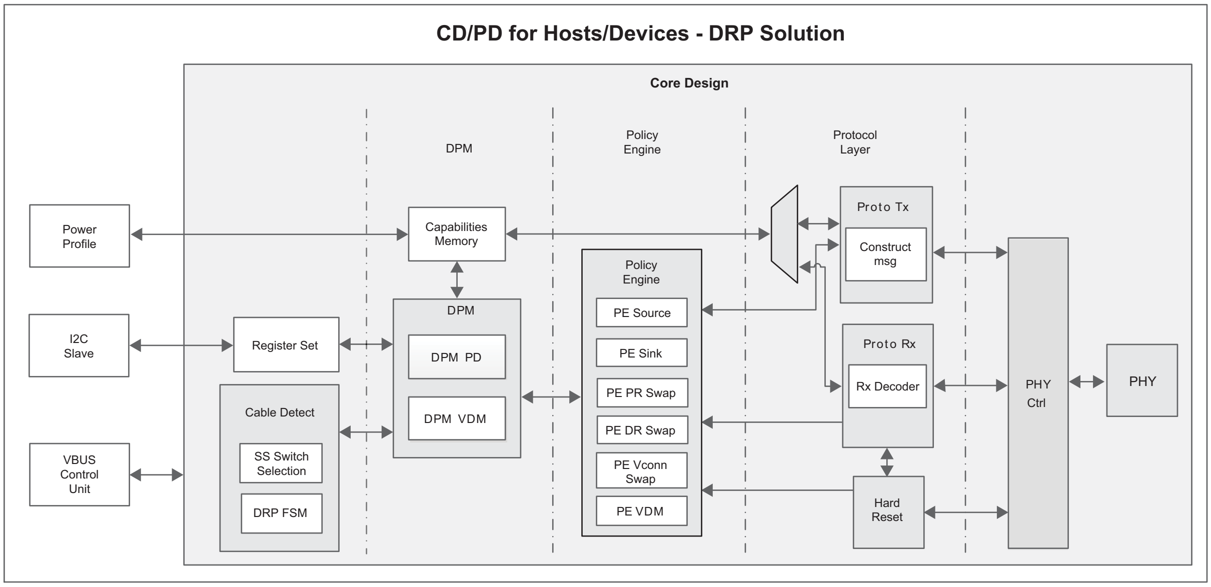
Note: This block diagram is only indicative. Refer to DS1052, Lattice USB Type-C Solution Data Sheet for the exact implementation.

Implementation: CD/PD for Hosts/Devices - DRP Solution

Figure 3 shows the detailed module level diagram of the DRP solution. It consists of four modules:

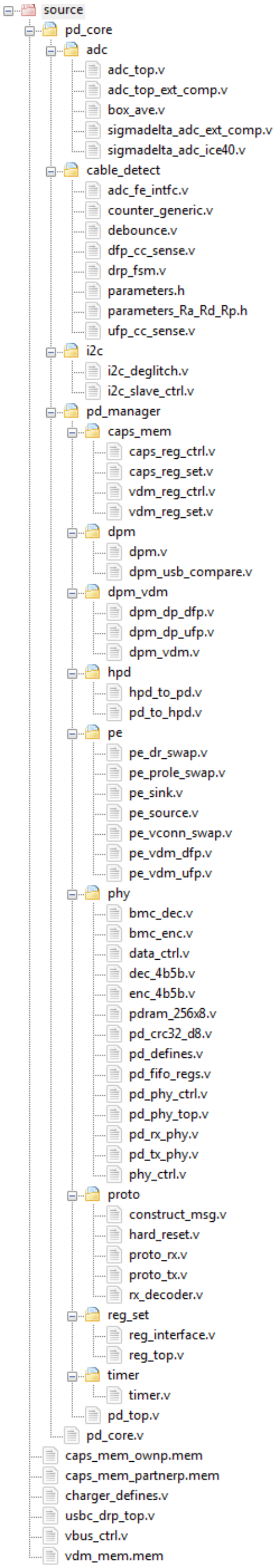
- Core Design
- I2C Slave Controller
- VBUS Control Unit
- Power Profile

Figure 3. DRP Solution Module Level Diagram



RTL Design Folder Structure

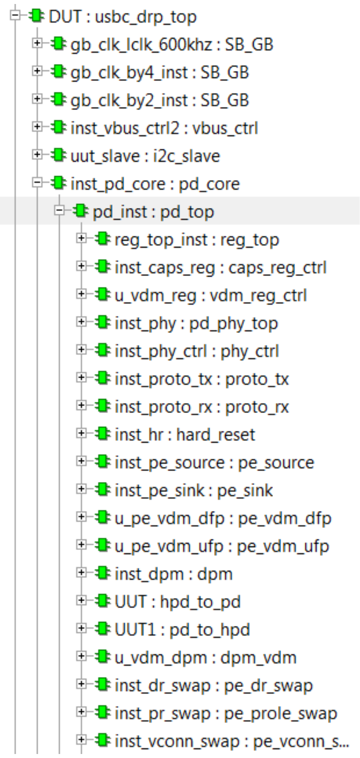
Figure 4. RTL Folder Structure



```
source
    pd_core
        adc
            adc_top.v
            adc_top_ext_comp.v
            box_ave.v
            sigmadelta_adc_ext_comp.v
            sigmadelta_adc_ice40.v
        cable_detect
            adc_fe_intfc.v
            counter_generic.v
            debounce.v
            dfp_cc_sense.v
            drp_fsm.v
            parameters.h
            parameters_Ra_Rd_Rp.h
            ufp_cc_sense.v
        i2c
            i2c_deglitch.v
            i2c_slave_ctrl.v
        pd_manager
            caps_mem
                caps_reg_ctrl.v
                caps_reg_set.v
                vdm_reg_ctrl.v
                vdm_reg_set.v
            dpm
                dpm.v
                dpm_usb_compare.v
            dpm_vdm
                dpm_dp_dfp.v
                dpm_dp_ufp.v
                dpm_vdm.v
            hpd
                hpd_to_pd.v
                pd_to_hpd.v
            pe
                pe_dr_swap.v
                pe_prole_swap.v
                pe_sink.v
                pe_source.v
                pe_vconn_swap.v
                pe_vdm_dfp.v
                pe_vdm_ufp.v
            phy
                bmc_dec.v
                bmc_enc.v
                data_ctrl.v
                dec_4b5b.v
                enc_4b5b.v
                pdram_256x8.v
                pd_crc32_d8.v
                pd_defines.v
                pd_fifo_regs.v
                pd_phy_ctrl.v
                pd_phy_top.v
                pd_rx_phy.v
                pd_tx_phy.v
                phy_ctrl.v
            proto
                construct_msg.v
                hard_reset.v
                proto_rx.v
                proto_tx.v
                rx_decoder.v
            reg_set
                reg_interface.v
                reg_top.v
            timer
                timer.v
            pd_top.v
        pd_core.v
    caps_mem_ownp.mem
    caps_mem_partnerp.mem
    charger_defines.v
    usbc_drp_top.v
    vbus_ctrl.v
    vdm_mem.mem
```

RTL Hierarchy

Figure 5. RTL Folder Structure



```
DUT : usbc_drp_top
  gb_clk_lclk_600khz : SB_GB
  gb_clk_by4_inst : SB_GB
  gb_clk_by2_inst : SB_GB
  inst_vbus_ctrl2 : vbus_ctrl
  uut_slave : i2c_slave
  inst_pd_core : pd_core
    pd_inst : pd_top
      reg_top_inst : reg_top
      inst_caps_reg : caps_reg_ctrl
      u_vdm_reg : vdm_reg_ctrl
      inst_phy : pd_phy_top
      inst_phy_ctrl : phy_ctrl
      inst_proto_tx : proto_tx
      inst_proto_rx : proto_rx
      inst_hr : hard_reset
      inst_pe_source : pe_source
      inst_pe_sink : pe_sink
      u_pe_vdm_dfp : pe_vdm_dfp
      u_pe_vdm_ufp : pe_vdm_ufp
      inst_dpm : dpm
      UUT : hpd_to_pd
      UUT1 : pd_to_hpd
      u_vdm_dpm : dpm_vdm
      inst_dr_swap : pe_dr_swap
      inst_pr_swap : pe_prole_swap
      inst_vconn_swap : pe_vconn_s...
```

Core Design

The core design for DRP solution consists of the following modules:

- Device Policy Manager (DPM)
- Policy Engine (PE)
- Protocol Layer
- PHY
- Cable Detect
- Register Set

Device Policy Manager

The Device Policy Manager consists of two modules:

- DPM for Power Delivery (PD)
- DPM for Vendor Defined Messages (VDM)
- Capabilities Memory

Device Policy Manager for Power Delivery

The Device Policy Manager applies local policies based on the fixed Register Set or inputs from the AP to the Power Delivery negotiation process.

Role of DPM in sink operation:

- Compare the received PDOs from the port partner and make a valid request.
- Handle Power Role SWAP, Data Role SWAP and VConn SWAP requests.
- Inform the AP about different power transitions.
- Decide on which SOP* to use for the current communication.

Role of DPM in source operation:

- Evaluate the request received from the port partner and make a decision on whether to send an Accept message or a Reject message.
- Handle Power Role SWAP, Data Role SWAP and VConn SWAP requests.
- Inform the AP about different power transitions.
- Decide on which SOP* to use for the current communication.

Device Policy Manager for Vendor Defined Message

Device Policy Manager for Vendor Defined Messages implements the VDM message flow as defined by the Display Port Alternate Mode. The message sequence identifies the attached UFP device. If the attached device supports Display Port Alternate Mode, the necessary actions are performed to enter Display Mode and the SS switch is configured accordingly. Otherwise, the AP receives information to enter Billboard Mode.

For VDM Message Flows for Display Port, see section 5.3 of the VESA Proposed Display Port Alternate Mode on USB Type-C Standard specification.

Capabilities Memory

This is a pre-initialized memory which has the default Port Source and Sink Power capabilities (PDO). In addition, it has space to hold the received port partners Source and Sink Power capabilities (PDO) for the AP to access.

Policy Engine

The Policy Engine interprets the Device Policy Manager's input in order to implement Policy for a given Port and directs the Protocol Layer to send appropriate messages. The Policy Engine consists of six modules,

- PE Source
- PE Sink
- PE PR Swap
- PE DR Swap
- PE VConn Swap
- PE VDM

PE Source

In a Dual Role Port, if the negotiated role by the Cable Detection module is Source, then this module is active, otherwise, it is disabled.

Functions:

- To handle the flow of different Power Delivery messages that has to be transmitted and received during the negotiation process.
- To handle the Soft Reset and Hard Reset messages to reset the protocol layer and power levels respectively.

For implementation details, refer to Figure 8-39 and Figure 8-41 of the USB PD specification.

PE Sink

In a Dual Role Port, if the negotiated role by the Cable Detection module is Sink, then this module is active, otherwise, it is disabled.

Functions:

To handle the flow of different Power Delivery messages that has to be transmitted and received during the negotiation process.

To handle the Soft Reset and Hard Reset messages to reset the protocol layer and power levels respectively.

For implementation details, refer to Figure 8-40 and Figure 8-42 of the USB PD specification

PE Power Role (PR) Swap

In a Dual Role Port, this module handles Port Role Swap requests and maintains the new and original Port Roles.

For implementation details, refer to Figure 8-51 and Figure 8-52 of the USB PD specification.

PE Data Role (DR) Swap

In a Dual Role Port, this module handles Data Role Swap requests and maintains the new and original Data Roles.

For implementation details, refer to Figure 8-51 and Figure 8-50 of the USB PD specification.

PE VConn Swap

VConn Swap is not supported. A VConn Swap request received will be rejected.

PE VDM

Policy Engine for VDM handles VDM message transfers without interrupting PD messages. It transfers initiator (mostly DFP) VDM messages as instructed by DPM VDM and informs the response from the responder (UFP mostly) back to DPM VDM.

For implementation details, refer to Figure 8-61 and Figure 8-68 of the USB PD specification.

Protocol Layer

The Protocol Layer is divided into three modules:

- Proto Tx
- Proto Rx
- Hard reset

Proto Tx

Proto Tx module transmits the Power Delivery packet. It implements the necessary retry counters and message ID handlers as shown in Figure 6-19 of USB PD specification.

The Proto Tx module includes a construct message module which forms the necessary Power Delivery packet based on the input from Policy Engine.

Proto Rx

Proto Rx module receives a Power Delivery Packet. Based on the stored message ID, the Proto Rx module decides on whether to pass the received packet or to ignore it as shown in Figure 6-20 of USB PD specification.

The Proto Rx module includes a received packet decoder module that decodes the received packet on a successful reception (if the packet need not be ignored), and passes the received Power Delivery message type to the Policy Engine.

Hard Reset

Hard Reset module handles the hard reset condition if received by the port partner or if initiated by own port. On a hard reset condition, this module makes sure that all power levels and protocol layer is reset to default before any further PD communication.

For implementation details, refer to Figure 6-21 of the USB PD specification.

PHY

This module has four internal modules:

- Tx and Rx FIFO
- PD PHY Ctrl
- PD PHY Tx
- PD PHY Rx

The PD raw data size is 300 Kb. After BMC encoding, data size becomes 600 Kb. By referring to the CC-line eye diagram in the PD specification, use 4.8 MHz clock to get 16 samples per nominal bit-length and 8 samples per nominal half bit length.

PHY module generates Good CRC message automatically based on the received message ID and the stored message ID from Protocol layer.

Tx and Rx FIFO

The Protocol Layer is responsible to initiate the PD message transmission by writing the associated FIFO and asserting corresponding PHY signals. The PD PHY performs the PD packet transfer based on this. PHY then interrupts the Protocol Layer with the transfer results. Also it detects and receives the packet from the PD front end. Then it interrupts the Protocol Layer with the received results. If the packet is error-free, the Protocol Layer reads data from the Rx FIFO.

PD PHY Ctrl

The PD PHY control module provides Control path signals to the PD PHY Tx and PD PHY Rx.

PD PHY Tx

This module consists of BMC encoder and transmitter. The PD raw data is of size as 300K. After BMC encoding data size is 600K. By referring to the CC-line eye diagram in the PD specification, it has been used 4.8 MHz clock to get 16 samples per nominal bit-length and 8 samples per nominal half bit-length.

PD PHY Rx

This module consists of BMC decoder and receiver.

Cable Detect

Cable detect is an Electrical/Analog frontend which is used to:

- Detect which CC line is connected to the port partner.
- Control the SS switch select signals.

The CD logic can be further divided into three parts:

- DFP/Source side CC vRd detection for Attach and vOpen detection for Detach
- vRa detection for VConn supply
- UFP/Sink side presence of VBUS for Attach and absence for Detach from VBUS control Unit

Table 1. DFP and UFP Behavior by State

State	DFP Behavior	UFP Behavior
Nothing attached	– Sense CC pins for attach – Do not apply V_{BUS} or V_{CONN}	– Sense V_{BUS} for attach
UFP attached	– Sense CC for orientation – Sense CC for detach – Apply V_{BUS} or V_{CONN}	– Sense CC pins for orientation – Sense loss of V_{BUS} for detach
Powered cable/No UFP attached	– Sense CC for attach – Do not apply V_{BUS} or V_{CONN}	– Sense V_{BUS} for attach
Powered cable/UFP attached	– Sense CC for orientation – Sense CC for detach – Apply VBUS or VCONN	– Sense CC for orientation – Sense loss of VBUS for detach
Debug Accessory Mode attached	– Sense CC for detach – Reconfigure for debug	– N/A
Audio Adapter Accessory Mode attached	– Sense CC for detach – Reconfigure for analog audio	– N/A

Table 2. USB Type-C DFP Connection States

CC1	CC2	State		
Open	Open	vOpen	Nothing attached	N/A
Rd	Open	vRd	UFP attached	①
Open	Rd			②
Open	Ra	vRa	Powered cable / No UFP attached	①
Ra	Open			②
Rd	Ra	vRd + vRa	Powered cable / UFP attached	①
Ra	Rd			②
Rd	Rd	vRd CC1 and CC2	Debug accessory mode attached	N/A
Ra	Ra	vRa CC1 and CC2	Audio Adapter Accessory Mode attached	N/A

The USB Type-C specification has a wide range of voltage detection requirements based on three different types of current advertisements from the Source/DFP. The voltage detection on DFP/Source side and the UFP/Sink side are defined in the specification. For quick reference, the tables below show the values for the current advertisement of 3 A by the DFP.

Table 3. CC Voltages on DFP Side - 3.0 A at 5 V

	Minimum Voltage	Maximum Voltage	Threshold
Powered cable / adapter (vRa)	0.00 V	0.75 V	0.80 V
UFP (vRd)	0.85 V	2.45 V	2.60 V
No connection (vOpen)	2.75 V		

Table 4. Voltage on UFP CC Pins (Multiple DFP Current Advertisements)

Detection	Minimum Voltage	Maximum Voltage	Threshold
vRa	–0.25 V	0.15 V	0.2 V
vRd-Connect	0.25 V	2.04 V	
vRd-USB	0.25 V	0.61 V	0.66 V
vRd-1.5	0.70 V	1.16 V	1.23 V
vRd-3.0	1.31 V	2.04 V	

The two major roles of the Cable Detect module are:

- SS switch selection
- DRP FSM

SS Switch Selection

One feature of the USB Type-C is the reversible nature of the plug. The Configuration Channel (CC1 and CC2) vRd detection identifies the orientation of the plug. Once the orientation is detected, the device should be able to re-arrange its Data Channels to proper alignment in order to send the data across to the Port-Partner. This is done by means of the external SS Switch for the SSTX and the SSRX data lanes of USB 3.1.

This design provides flexible number of IOs that can be used to perform the SS switch lane selection by controlling the select lines available on the switches. The design can provide up to four IOs for the select lines.

Notes:

- *Not all USB Type-C devices need the SS switches. This requirement is only for USB 3.1 data communication capable devices.*
- *Devices such as USB chargers and those with Captive Cables may not need cross bar switches since the orientation is either fixed due to captive nature or not required due to the absence of USB data communication.*

DRP FSM

This design is implemented to achieve the full functionality defined in the USB Type-C specification. There are three types of ports based on this specification:

- UFP (Device type)
- DFP (Host type)
- DRP (can be both UFP and DFP)

The Lattice CD/PD Controller DRP is implemented in such a way that only additional features need to be removed to make the device either a UFP or a DFP. It makes the design a superset of the Type-C implementation, allowing easy customization for any subset implementation. For example, a charger is just a DFP without any SS switch selection. It can be easily created from this solution by removing the Sink/UFP related modules from both the CD and the PD logic.

For details on the selected DRP implementation, refer to the USB Type-C 1.1 specification.

Register Set

Register set is the register interface between the Application Processor (AP) and the Lattice USB 3.1 Type-C Power Delivery (PD) solution. Power Delivery is an interrupt driven process. Any major event in the Power Delivery process generates an interrupt to the AP and the AP responds with the necessary actions. The AP can access the following set of registers defined to control and identify the status of the Power Delivery process.

- Control Registers
- Status Registers
- Interrupt Registers
- Capabilities Memory Interface

Refer to RD1209, Lattice USB Type-C Solution for complete details on the Register Interface using I2C Slave interface.

I2C Slave Controller

This module is the I2C serial interface between the Register Set module and the AP I2C Slave Controller.

VBUS Control Unit

The primary functions of the VBUS Control Unit are:

- Detecting the availability of VBUS when Sink and informing the Cable detect module
- Enabling VBUS output when Source, as directed by the Cable detect module

Multiple voltage support and transitions are expected to be controlled by EC over I2C

When the VBUS_CTRL_MODULE_EN option is enabled, the following functions apply:

- Enabling a particular Voltage Level during Power Transition, as directed by DPM
- Monitoring VBUS line to check if a particular voltage has been reached
- Handling VBUS discharge

Power Profile

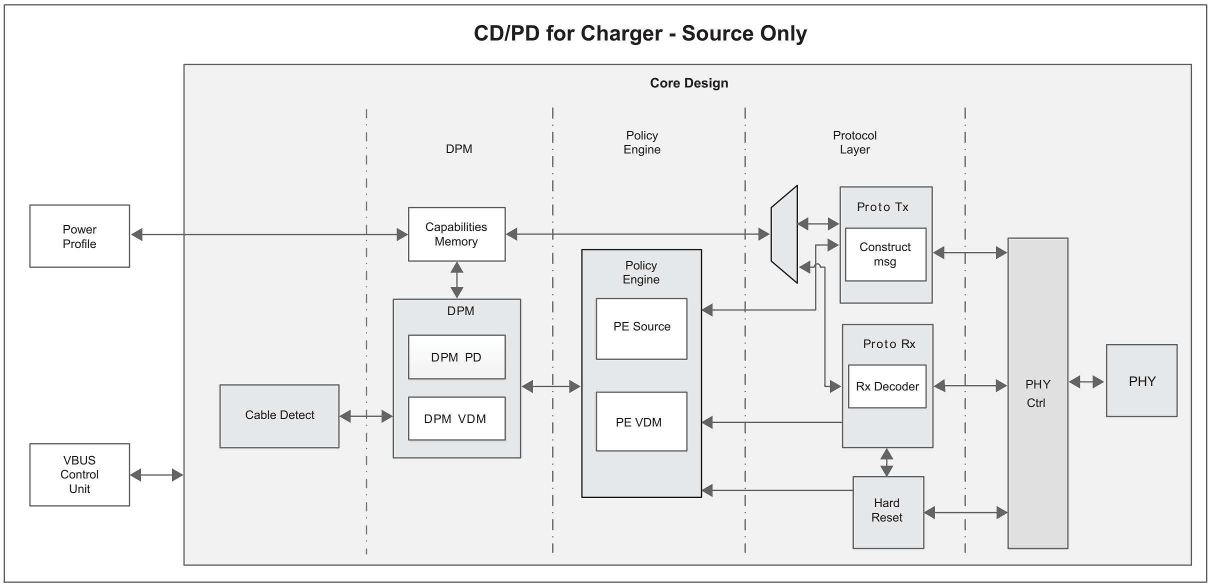
This module consists of the *caps_mem.mem* memory file that initializes the Capabilities Memory. By default, a single Power Delivery Object (PDO) is loaded. To modify the supported default Power Profiles or to increase the number of supported PDOs, this initialization file should be updated.

Implementation: CD/PD for Chargers – Source Only

Figure 6 shows the detailed module level diagram of DRP solution. It consists of four modules:

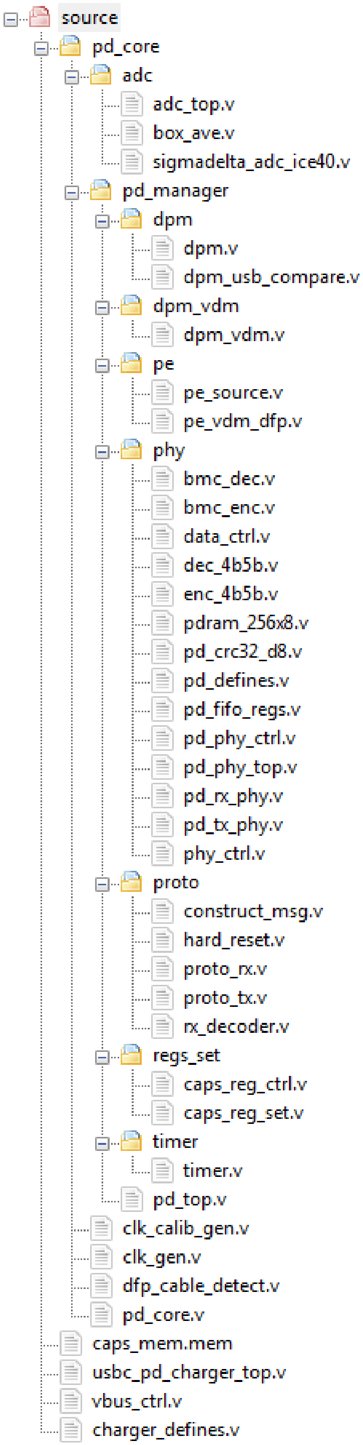
- Core Design
- VBUS Control Unit
- Power Profile

Figure 6. CD/PD for Charger - Source Only



RTL Design Folder Structure

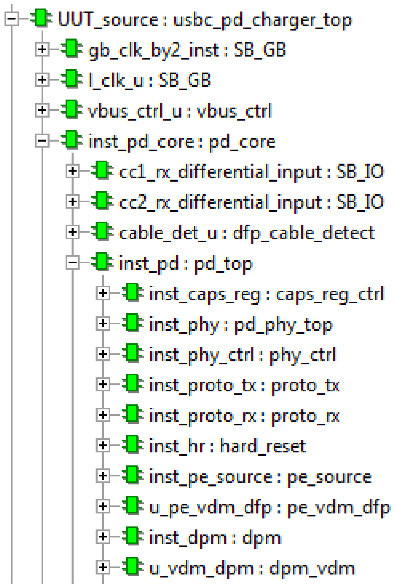
Figure 7. RTL Folder Structure



```
source
    pd_core
        adc
            adc_top.v
            box_ave.v
            sigmadelta_adc_ice40.v
        pd_manager
            dpm
                dpm.v
                dpm_usb_compare.v
            dpm_vdm
                dpm_vdm.v
            pe
                pe_source.v
                pe_vdm_dfp.v
            phy
                bmc_dec.v
                bmc_enc.v
                data_ctrl.v
                dec_4b5b.v
                enc_4b5b.v
                pdram_256x8.v
                pd_crc32_d8.v
                pd_defines.v
                pd_fifo_regs.v
                pd_phy_ctrl.v
                pd_phy_top.v
                pd_rx_phy.v
                pd_tx_phy.v
                phy_ctrl.v
            proto
                construct_msg.v
                hard_reset.v
                proto_rx.v
                proto_tx.v
                rx_decoder.v
            regs_set
                caps_reg_ctrl.v
                caps_reg_set.v
            timer
                timer.v
            pd_top.v
        clk_calib_gen.v
        clk_gen.v
        dfp_cable_detect.v
        pd_core.v
    caps_mem.mem
    usbc_pd_charger_top.v
    vbus_ctrl.v
    charger_defines.v
```

RTL Design Hierarchy

Figure 8. RTL Hierarchy



```
UUT_source : usbc_pd_charger_top
    gb_clk_by2_inst : SB_GB
    l_clk_u : SB_GB
    vbus_ctrl_u : vbus_ctrl
    inst_pd_core : pd_core
        cc1_rx_differential_input : SB_IO
        cc2_rx_differential_input : SB_IO
        cable_det_u : dfp_cable_detect
        inst_pd : pd_top
            inst_caps_reg : caps_reg_ctrl
            inst_phy : pd_phy_top
            inst_phy_ctrl : phy_ctrl
            inst_proto_tx : proto_tx
            inst_proto_rx : proto_rx
            inst_hr : hard_reset
            inst_pe_source : pe_source
            u_pe_vdm_dfp : pe_vdm_dfp
            inst_dpm : dpm
            u_vdm_dpm : dpm_vdm
```

Core Design

The core design for DRP solution consists of the following modules:

- Device Policy Manager (DPM)
- Policy Engine (PE)
- Protocol Layer
- PHY
- Cable Detect

Device Policy Manager

The Device Policy Manager consists of two modules:

- DPM for Power Delivery (PD)
- DPM for Vendor Defined Messages (VDM)
- Capabilities Memory

Device Policy Manager for Power Delivery

The Device Policy Manager applies local policies to the Power Delivery negotiation process.

Role of DPM:

- Evaluate the request received from the port partner and make a decision on whether to send an Accept message or a Reject message.
- Choose which SOP* to use for the current communication.

Device Policy Manager for Vendor Defined Message

Device Policy Manager for Vendor Defined Messages handles the SOP' communications to identify the capabilities of the attached cable plug.

Capabilities Memory

This is a pre-initialized memory which has the default Port Source and Sink Power capabilities (PDO). In addition, it has space to hold the received port partner's Sink Power capabilities (PDO).

Policy Engine

The Policy Engine interprets the Device Policy Manager's input in order to implement Policy for a given Port and directs the Protocol Layer to send appropriate messages. The Policy Engine consists of two modules,

- PE Source
- PE VDM

PE Source

This module implements the necessary statemachine for a power negotiation process.

Functions:

- To handle the flow of different Power Delivery messages that has to be transmitted and received during the negotiation process.
- To handle the Soft Reset and Hard Reset messages to reset the protocol layer and power levels respectively.

For implementation details, refer to Figure 8-39 and Figure 8-41 of the USB PD specification.

PE VDM

Policy Engine for VDM handles VDM message transfers without interrupting PD messages. It transfers initiator (mostly DFP) VDM messages as instructed by DPM VDM and informs the response from the responder (UFP mostly) back to DPM VDM.

For implementation details, refer to Figure 8-61 and Figure 8-68 of the USB PD specification.

Protocol Layer

The Protocol Layer implementation is the same as described in Protocol Layer subsection under Implementation: CD/PD for Hosts/Devices - DRP Solution..

PHY

The PHY implementation is the same as described in the PHY subsection under Implementation: CD/PD for Hosts/Devices - DRP Solution.

Cable Detect

Cable detect is an Electrical/Analog frontend which is used to:

- DFP/Source side CC vRd detection for Attach and vOpen detection for Detach
- vRa detection for VConn supply

VBUS Control Unit

The primary functions of the VBUS Control Unit are:

- Enabling VBUS output, as directed by the Cable Detect module
- Enabling a particular Voltage Level during Power Transition, as directed by DPM
- Monitoring VBUS line to check if a particular voltage has been reached
- Handling VBUS discharge

Power Profile

This module basically consists of the *caps_mem.mem* memory file that initializes the Capabilities Memory. By default, five Power Delivery Objects (PDO) are loaded. To modify the default supported Power Profiles or to increase the number of supported PDOs, this initialization file should be updated.

Customizing the Solutions

There are 4 modifiable sections in the solutions,

- Core Design
- VBUS Control Unit
- Power Profile
- I2C Slave Controller

Core Design

The Core Design is a compliant design. This section of the solutions is not expected to be customized or updated.

VBUS Control Unit

CD/PD for Host/Devices

‘VBUS_CTRL_MODULE_EN’ should be defined in the *charger_defines_drp.v* file if the voltage control unit of the solution needs to be enabled. Else, the voltage control and multiple voltage transitions are expected to be controlled by the EC. The solution interrupts the EC upon a voltage transition request.

If the ‘VBUS_CTRL_MODULE_EN’ is defined, the ADC threshold values in `charger_defines_drp.v` file, as shown below, must be updated according to the supported voltage. The values are characterized for the indicated voltage range. Any different voltage range used must be characterized.

```
`define VBUS_0P5V      83 //0.5V, unit = 25.16129mV
`define VBUS_4P75V    231//4.75V, unit = 25.16129mV
`define VBUS_5P5V     258//5.5V, unit = 25.16129mV
`define VBUS_8P55V    364//8.55V, unit = 25.16129mV
`define VBUS_9P7V     402//9.7V, unit = 25.16129mV
`define VBUS_11P4V    470//11.4V, unit = 25.16129mV
`define VBUS_12P85V   508//12.85V, unit = 25.16129mV
`define VBUS_18P05V   740//18.05V, unit = 25.16129mV
```

The parameters below are updated according to the different Power Profiles.

```
`define V_0V 3'd0
`define V_5V 3'd1
`define V_9V 3'd2
`define V_12V 3'd3
`define V_19V 3'd4

`define VSEL_0V 4'b0000
`define VSEL_5V 4'b0001
`define VSEL_9V 4'b0010


`define VSEL_12V 4'b0100
`define VSEL_19V 4'b1000
```

Also, the lines of code in vbus_ctrl.v file shown below must be modified according to the number of supported Power Profiles and their values.

```
assign vbus_low_th   = req_vbus_status == `V_5V  ? VBUS_4P75V  :
      req_vbus_status == `V_9V  ? VBUS_8P55V  :
      req_vbus_status == `V_12V ? VBUS_11P4V  :
      req_vbus_status == `V_19V ? VBUS_18P05V :
      0;

assign vbus_high_th  = req_vbus_status == `V_0V  ? VBUS_0P5V :
      req_vbus_status == `V_5V  ? VBUS_5P5V :
      req_vbus_status == `V_9V  ? VBUS_9P7V :
      req_vbus_status == `V_12V ? VBUS_12P85V :
      0;
```

CD/PD for Charger

The ADC threshold values in the *charger_defines.v* file must be updated according to the supported voltage. The values are characterized for the indicated voltage range. Any different voltage range used must be characterized.

```
      `define VBUS_0P5V      25 //0.5V, unit = 25.16129mV
      `define VBUS_4P75V    170//4.75V, unit = 25.16129mV
      `define VBUS_5P5V     215//5.5V, unit = 25.16129mV
      `define VBUS_8P55V    320//8.55V, unit = 25.16129mV
      `define VBUS_9P7V     375//9.7V, unit = 25.16129mV
      `define VBUS_11P4V    433//11.4V, unit = 25.16129mV
      `define VBUS_12P85V   492//12.85V, unit = 25.16129mV
      `define VBUS_18P05V   716//18.05V, unit = 25.16129mV
```

The parameters shown below are updated according to the different Power Profiles.

```
      `define V_0V  3'd0
      `define V_5V  3'd1
      `define V_9V  3'd2
      `define V_12V 3'd3
      `define V_19V 3'd4

      `define VSEL_0V  4'b0000
      `define VSEL_5V  4'b0001
      `define VSEL_9V  4'b0010
      `define VSEL_12V 4'b0100
      `define VSEL_19V 4'b1000
```

Also, the lines of code in the vbus_ctrl.v file shown below must be modified according to the number of supported Power Profiles and their values.

```
assign vbus_low_th   = req_vbus_status == `V_5V  ? VBUS_4P75V  :
      req_vbus_status == `V_9V  ? VBUS_8P55V  :
      req_vbus_status == `V_12V ? VBUS_11P4V  :
      req_vbus_status == `V_19V ? VBUS_18P05V :
      0;

assign vbus_high_th  = req_vbus_status == `V_0V  ? VBUS_0P5V :
      req_vbus_status == `V_5V  ? VBUS_5P5V :
      req_vbus_status == `V_9V  ? VBUS_9P7V :
      req_vbus_status == `V_12V ? VBUS_12P85V :
      0;
```

Power Profile

The *caps_mem.mem* file initializes capabilities memory on power on with the desired Power Profiles. The following is a sample 32-bit data content of this file starting from address 8’h10.

```
2A01905A
00000000
00000000
00000000
00000000
00000000
00000000
2A01905A
00000000
00000000
00000000
00000000
00000000
A1000000
00000000
22000021
```

Table 5 and Table 6 show the decoding of Source and Sink PDOs. Table 7 shows the PDO count. The number of supported Source and Sink PDOs should be updated in the *caps_mem.mem* file. Source PDOs start from address 10h, sink PDOs starts from address 17h and PDO Count value is present at address 19h. A maximum of six PDOs are supported in this design.

Table 5. Source PDO example

(Source PDO) x2A 01 90 5A – Start Address x10									
[31:30]	[29]	[28]	[27]	[26]	[25]	[24:22]	[21:20]	[19:10]	[9:0]
Fixed supply	Dual-Role Power	USB Suspend Supported	Externally Powered	USB Comm Capable	DR Swap	Reserved	Peak Current	Voltage in 50 mV units	Maximum Current in 10 mA
00	0	0	1	0	0	000	00	10'b00_0110_0100	10'b00_0010_1000
								(that is 100 * 50 mV = 5 V)	(that is 90 * 10 mA = 900 mA)

Table 6. Sink PDO Example

(Sink PDO) x2A 01 90 5A – Start Address x17									
[31:30]	[29]	[28]	[27]	[26]	[25]	[24:22]	[21:20]	[19:10]	[9:0]
Fixed supply	Dual-Role Power	USB Suspend Supported	Externally Powered	USB Comm Capable	DR Swap	Reserved	Peak Current	Voltage in 50 mV units	Maximum Current in 10 mA
00	0	0	1	0	0	000	00	10'b00_01 10_0100	10'b00_00 10_1000
								(that is 100 * 50 mV = 5 V)	(that is 90 * 10 mA = 900 mA)

Table 7. PDO Count Example

PDO Count – Address x1F				
Bit	**Name**	**Default**	**Access**	**Description**
[31:29]	Source PDO count[1]	001	R/W	Holds the value of number of valid source PDOs in the source PDO register set that support current less than or equal to 3 A
28	Reserved	0	R/W	
[27:25]	Sink PDO count	001	R/W	Holds the value of number of valid sink PDOs in the sink PDO register set
[24:8]	Reserved	0	R/W	
[7:5]	Source PDO count 5 A[2]	001	R/W	Holds the value of total number of valid source PDOs in the source PDO register set that support all current ranges (0 A – 5 A).
3	PDO UPDATE	0	R/W	Set this bit when the new source PDO count value or new sink PDO number is updated.
[2:0]	SINK PDO NUM	001	R/W	When current Power Role is Source: N/A When current Power Role is Sink: Holds the own port Sink PDO number for which a PD contract should be established

[1] Used when the attached cable plug is only 3A capable
[2] Used when the attached cable plug is 5A capable

Example:

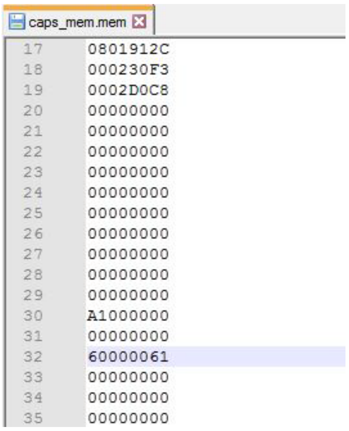
In the CD/PD for Charger – Source Only solution, the requirement is to support three source PDOs:

- 5 V at 3 A
- 7 V at 2.43 A
- 9 V at 2.0 A

The following steps should be taken to customize the design:

1. Open the *caps_mem.mem* file in source folder as shown in Figure 9.
2. Source capabilities starts from 10h address, that is from line number 17.
3. Change the values from line number 17 to line number 22 according to Table 5.
4. Update the PDO number field as shown in Table 7, which is present at address 19h, that is line number 32.

Figure 9. Caps_mem.mem File



```
0801912C
000230F3
0002D0C8
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
A1000000
00000000
60000061
00000000
00000000
00000000
```

I2C Slave Controller

This I2C Slave controller can be used as is or can be replaced by an SPI slave or any other serial interface between the Register set module and AP. The I2C interface is only available in the CD/PD for Host/Devices. Refer to RD1209, Lattice USB Type-C Solution for I2C register information.

Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

Revision History

Date	Version	Change Summary
October 2016	1.4	General update on structure and contents.
		Updated Introduction section. Revised the solutions that are discussed in the document.
		Removed the following sections: — Module Level Diagram of Lattice USB 3.1 Type-C Solution — Lattice USB 3.1 Type-C Solution with Micro-controller (MICO8) — Lattice USB 3.1 Type-C Solution with an Application Processor — Structured VDM DFP DisplayPort Flow — Structured VDM UFP DisplayPort Flow — Appendix A. Intrinsyc Qualcomm Snapdragon 800 (APQ8074) Based DragonBoard Development Kit Pin Map
		Added the following sections: Implementation: CD/PD for Hosts/Devices - DRP Solution Implementation: CD/PD for Chargers – Source Only Customizing the Solutions
August 2015	1.3	Updated Lattice PD Driver APIs section.
July 2015	1.2	General update – applied editorial and style changes.
		Added the following sections: — Structured VDM DFP DisplayPort Flow — Structured VDM UFP DisplayPort Flow
		Updated Lattice USB 3.1 Type-C Solution with Micro-controller (MICO8) section. — Revised Figure 11, Module Level Diagram of Lattice USB 3.1 Type C Solution with Mico. — Changed subsection heading to DPM in Mico8 and revised contents.
		Updated Lattice USB 3.1 Type-C Solution with an Application Processor section. — Revised Figure 12, Software Architecture. Added two GPIO signals.
		Updated Lattice PD Driver APIs section.
		Updated Appendix A. Intrinsyc Qualcomm Snapdragon 800 (APQ8074) Based DragonBoard Development Kit Pin Map section.
June 2015	1.1	General update – removed version number and page references in specifications.
		Updated Introduction section.
		Updated System Block Diagram section.
		Updated System Level Functional Block Diagram section.
		Updated Module Level Diagram of Lattice USB 3.1 Type-C Solution section. The following subsections are revised: — Cable Detect (CD) — V_{BUS} Detection — DRP FSM — Device Policy Manager for Vendor Defined Message
		Added the following sections: — Lattice USB 3.1 Type-C Solution with an Application Processor — Appendix A. APQ8074 and PHY Board Setup
		Updated the References section.
		Updated the Technical Support Assistance section.
April 2015	1.0	Initial release.