RCE Generation 3: Example Project: Building and Running: Firmware

Introduction

This confluence page describes how to build and update the firmware. For this project, there are two flavors of firmware: "Simple" and "PGP".

Licensing

If you are building on a SLAC server:

```
$ cd /u1/ExampleRceProject/firmware

$ source setup_env.csh
```

If you are building on a remote server, you (or your IT department) will be responsible for setting up the FLEX_LM licensing server and install Vivado software

> **Vivado Version**
>
> Currently, all firmware targets are locked to Vivado Version 2014.4 and will not build if otherwise.

Creating a build output directory

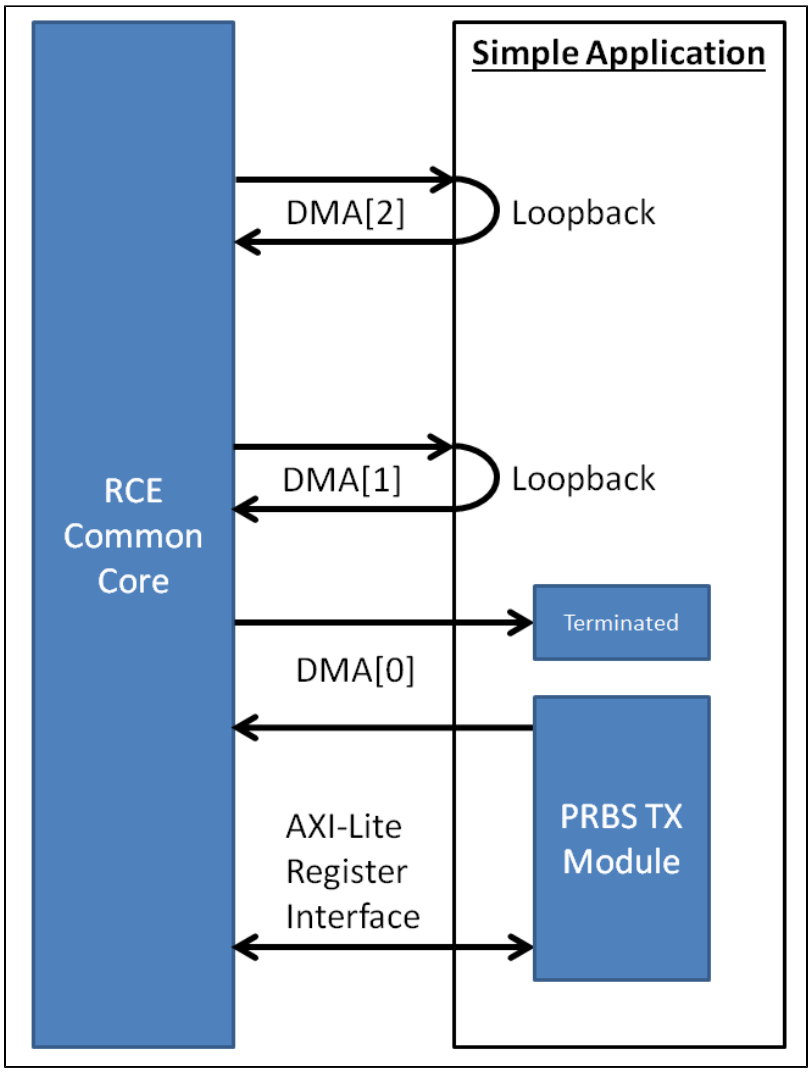
Before running the Makefile, a build directory needs to be created:

```
$ cd /u1/ExampleRceProject/firmware

$ mkdir build
```

> **Build Directory Not Included in SVN Checkout**
>
> A build directory is not included in the SVN checkout since some users build their software on a network drive (Example: AFS, NFS, and etc) and want to soft-link their build directory to a local hard drive mount point.

Building SimpleDpm and SimpleDtm Firmware



This is a simple example firmware that has no high speed IOs interfaces. DMA[2:1] are configured as loopback DMA channels. The outbound DMA[0] channel is terminated to prevent back pressure if sent from software. The inbound DMA[0] channel is connected to a PseudoRandom Binary Sequence generator (PRBS TX). By default, the PRBS TX module is constantly generating data on the inbound DMA[0] channel.

The following are the steps for building the firmware .bit image files:

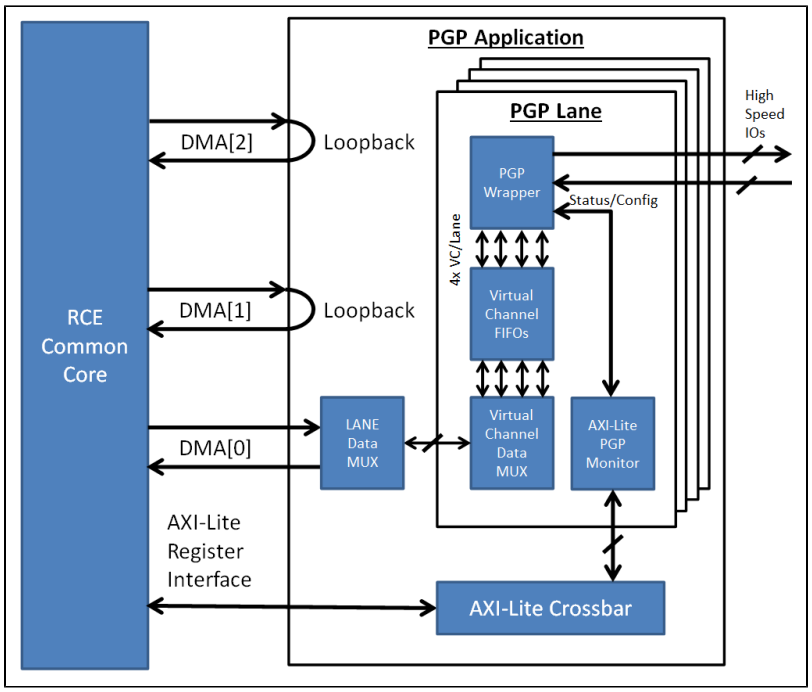
```
$ cd /u1/ExampleRceProject/firmware/target/SimpleDpm/

$ make

$ cd /u1/ExampleRceProject/firmware/target/SimpleDtm/

$ make
```

Building PgpDpm and PgpDtm Firmware



This is example firmware that uses the high speed IOs interfaces on the RTM. The communication protocol for the high speed IOs is Pretty Good Protocol (PGP). DMA[2:1] are configured as loopback DMA channels. Inbound and outbound DMA[0] channel are connected to a data multiplexer/demultiplexer, which multiplexes the data to/from the different PGP lanes. Each PGP lane has a data multiplexer/demultiplexer, which multiplexes the data to/from the different PGP virtual channels within a lane. In between the virtual channel data multiplexer/demultiplexer and the PGP IP core wrapper are FIFOs for buffering. Also attached to the PGP IP core wrapper is a AXI-Lite PGP monitor, which monitors the status and configures the PGP IP core wrapper. All the AXI-Lite PGP monitor modules are connected to a AXI-Lite crossbar module. The AXI-Lite crossbar handles arbitration between the different AXI-Lite slave register modules and the AXI-Lite master software register interface. So that this example doesn't require a RTM, each PGP lane will be configured in loopback module in software. There are a total of 12 high speed PGP links for the DPM and 1 high speed PGP link for the DTM. The PGP link speeds are all configured for 3.125 Gbps.

The following are the steps for building the firmware .bit image files:

```
$ cd /u1/ExampleRceProject/firmware/target/PgpDpm/

$ make

$ cd /u1/ExampleRceProject/firmware/target/PgpDtm/

$ make
```

The application register mapping for this firmware is as followed:

Address[0xBFFFFFFF:0x80000000] = RCE Common

Address[0xA000FFFF:0xA0000000] = PGP Monitor, Lane#0

Address[0xA001FFFF:0xA0010000] = PGP Monitor, Lane#1 (DPM only)

Address[0xA002FFFF:0xA0020000] = PGP Monitor, Lane#2 (DPM only)

Address[0xA003FFFF:0xA0030000] = PGP Monitor, Lane#3 (DPM only)

Address[0xA004FFFF:0xA0040000] = PGP Monitor, Lane#4 (DPM only)

Address[0xA005FFFF:0xA0050000] = PGP Monitor, Lane#5 (DPM only)

Address[0xA006FFFF:0xA0060000] = PGP Monitor, Lane#6 (DPM only)

Address[0xA007FFFF:0xA0070000] = PGP Monitor, Lane#7 (DPM only)

Address[0xA008FFFF:0xA0080000] = PGP Monitor, Lane#8 (DPM only)

Address[0xA009FFFF:0xA0090000] = PGP Monitor, Lane#9 (DPM only)

Address[0xA00AFFFF:0xA00A0000] = PGP Monitor, Lane#10 (DPM only)

Address[0xA00BFFFF:0xA00B0000] = PGP Monitor, Lane#11 (DPM only)

Memory Address

All AXI-Lite memory addresses are byte based. For Pgp2bAxi.cpp, addrSize = 4.

Updating the .bit image on the SD memory card

The .bit image on the SD memory card can be updated using the following commands:

```
$ scp $TARGET_PATH root@$IP_ADDRESS:/mnt/boot/fpga.bit

$ ssh root@$IP_ADDRESS

# sync

# reboot
```

Where $TARGET_PATH is the path to the new .bit file and $IP_ADDRESS is the IP address of the RCE that you want to update. The "sync" command synchronizes the SD memory card.