Baseband Architecture Evolution Application Note - 418

Notes

Introduction

Baseband architectures will evolve to be more scalable and modular in the future. In other published articles, we have talked about current baseband architectures, and proposed some ideal baseband architectures [1] [2]. In this application note, we will discuss the realistic progression from current architectures to an ideal one.

Current architectures, which are heavily based on sequential processing, are introduced in [1]. In this application note, we will first introduce an interim architecture, and show how IDT's Pre-Processing Switch (PPS) might bring value. Then we move into architecture 2, which is the ideal solution, and show how the PPS enhances performance and lowers the cost for this system.

Architecture 1 - Interim Architecture

In this architecture, as shown in Figure 1.1 there are front end ASICs or FPGAs handling the chip rate or other fast processing. Once this processing is done, symbol rate processing needs to be performed. Symbol rate processing is usually performed in one or more DSPs.

DSPs are connected by using serial RapidIO™ using the PPS. Symbol rate processing speeds are much slower (down to less than 100 Mbps), so most of the manipulation functions can be done by the DSP since data rate is slower. Symbol rate processing includes functions such as 3G framing, FEC, and interleaving.

Fast manipulation operations in the pre-processing functions integrated into the PPS may not be as useful in this architecture, but the PPS has other excellent functions that enable the users to shorten the development time of the whole baseband system, and address time-to-market issues. Also the next generation of the PPS, which is called as PPS-Lite will have real Finite Impulse Response filters in the Packet Processing Scenario (PPSc) functional blocks which can be used for filtering operations as samples traverse through the PPS.

The most useful functions of PPS/PPS-Lite in this configuration are:

1) Trace function: When the first 160 bits of a packet matches 1 of the 4 possible programmed values on each port, the packet is replicated to a designated port - the "Trace Port". This is really useful in debugging and monitoring operations

2) Real FIR filter: All PPScs can implement FIR filters with 8 taps at speeds up to 10Gbps

3) Design for Test functions: The PPS has extensive debugging features; essentially user can loop back packets anywhere in the device. This is really useful in debugging the whole basestation not just the baseband card, and will shorten development time significantly. See Figure 1.2 for more details.

4) Switching Capability: The PPS Switch capability (peak of 10Gbps) is more than enough for these slow rates, and the PPS does not have extra switch functions such as traffic monitoring or extensive flow control that increase the cost of the device.

Notes

5) Low latency multicast: Using PPScs, users can get lower latency for multicast: If there are 3 packets to be multicast to 3 ports, a regular switch will route 9 total packets, which will interfere with one another. In the PPScs, the same can be done by combining packets and multicasting one packet to 3 ports. This will offer much lower latency as there is no contention.

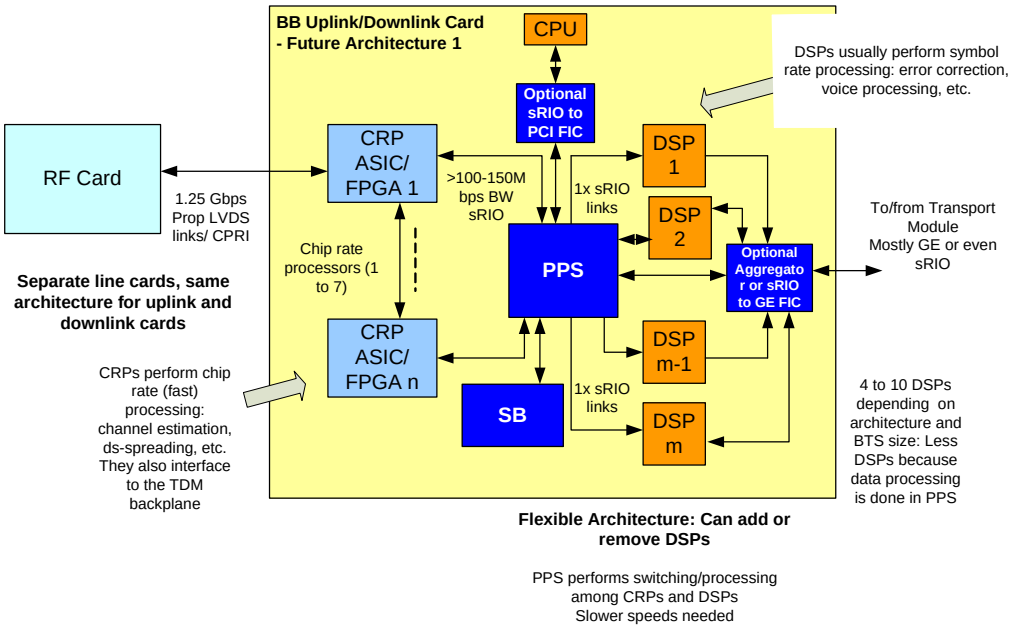

Figure 1.1 Interim Architecture

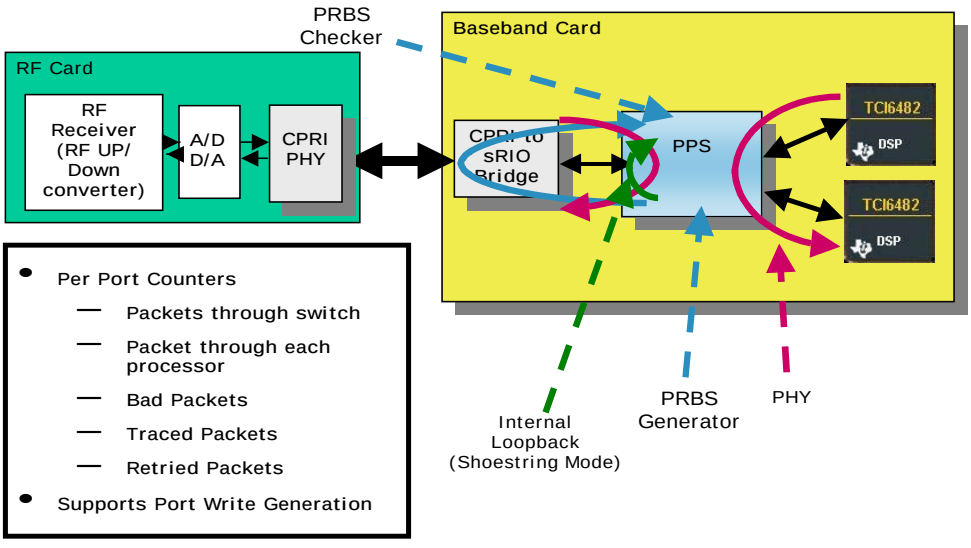

Figure 1.2 DFT Functionality

Notes

Architecture 2 - Ideal Architecture

This is a lot more modular and scalable [2] as the PPS moves to the forefront of the Baseband (BB) card, and handles both fast sample rate traffic, and slow symbol rate traffic. This is shown below in Figure 1.3, which uses the OBSAI protocol RP-3 as an example of how RF traffic might be brought to the base band board.

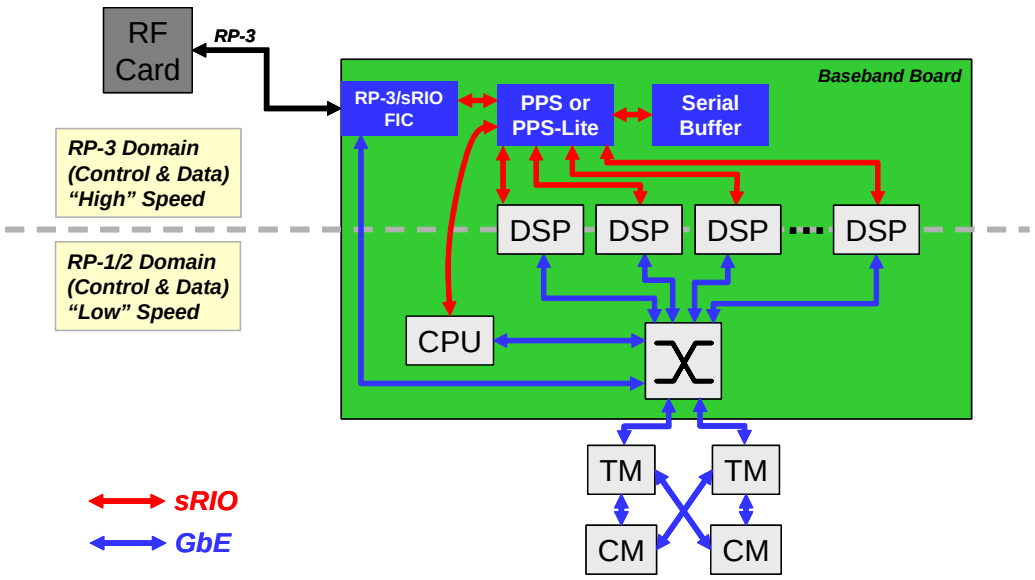

Figure 1.3 Ideal Architecture

There are two different ways to implement baseband algorithms (as shown in Figure 1.1) with this architecture:

2a) Sequential: Each DSP handles a specific portion of the algorithm for all users supported on this card (usually each card support about 128 users), and pass it onto the next DSP through the PPS. PPScs may offload at every step. It offloads more when data rates are faster (before the chip rate processing DSP)

Occasionally, FPGA or ASIC might replace some of the DSPs, since DSPs are usually not designed to handle parallel tasks for all users. Usually this FPGA or ASIC might be implementing front end algorithm blocks such as de-spreading, rake receiver, etc.

PPSc offload numbers are given in the following section for Architecture 2a.

2b) Parallel: Each DSP handles end-to-end functions for some number of users. This is more scalable as software can be replicated for each DSP. The assumption here is that DSPs are capable of handling end to end operations for some number (usually 24-32 users each) of users. This is becoming more feasible because of two reasons:

i. The latest Texas Instruments (TI) and Freescale DSPs are very powerful in implementing chip rate processing functions.

Notes

ii. This assumes that DSP can support sample data rates up to 1.5 - 2G from the RF cards. With the new sRIO interface, this is also feasible, because of its bandwidth, and its automatic handling of these operations without DSP core interference.

Another look at architecture 2 at the system level is shown below. Note that there is CPRI combiner card to eliminate routes between RF cards and BB cards. The PPS and PPS-Lite as well as CPRI-sRIO FIC are shown in support of this architecture.

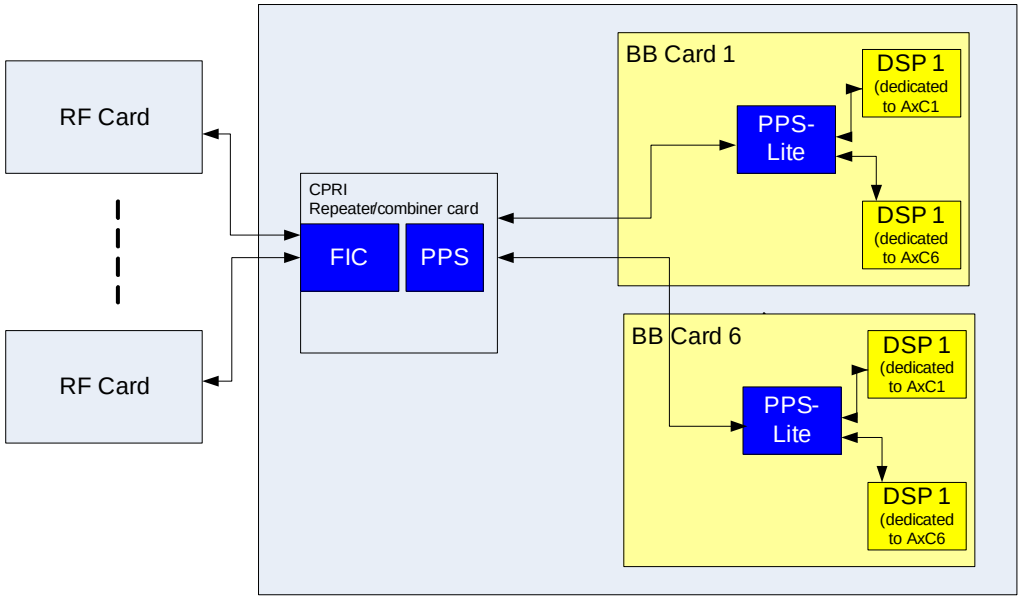

Figure 1.4 System Level Ideal Architecture

Summary of benefits of using PPS and PPS-Lite in Architecture 2a, 2b are:

1. The PPS can do downlink summing for multiple cards before samples are shipped to RF cards, and all other uplink and downlink sample manipulation operations. [2] The PPS-Lite has less ports, and can be used on individual BB cards for the same functions.

2. The PPS-Lite has better multicast properties than a regular switch - a PPSc has much less latency in multicasting multiple packets to multiple locations.

3. The PPS-Lite has up to 12 ports which is ideal as a line card switch and pre-processor solution

4. DSP software is the same for all DSPs. This results in easier management, and upgrading to more users is as easy as adding one more BB card. The BB card and RF card are exclusive. The PPS or PPS-Lite can be used to broadcast programming and maintenance packets to DSPs.

5. PPS and PPS-Lite work within DSP software, easily programmable from DSP development tools such as TI's Code Composer Studio™.

Using the PPS and PPS-Lite in Architecture 2, users can save DSP cycles. In the next section, we will give the number of cycles and resources saved by using PPScs:

Notes

Calculation of Offload

We need to calculate the offload of a PPSc from a DSP for fast data rate operations. This is valid only for architecture 2.

Assumptions are numbered:

1) Now assume that there are 6 sectors with 2 frequencies each in a big macro station. So there are a total of 12 antenna channels.

2) Assume that A/D oversamples by 2 with 8 bits I + 8 bits Q = 16 bits/sample, so there are 3.84 Mcps * 2 * 16 = 122.88 Mbps per antenna channel flow.

Since there are 12 antenna-channels, this makes about 12 * 122.88 Mbps = 1476.54 Mbps total data coming into each baseband card.

3) Further assume that all RF card data come to all BB cards, i.e. 1476.54 Mbps broadcast to all BB cards, so the front end device needs to handle this. This is where sRIO comes in; sRIO is used to push this data rate into the DSPs.

Assumption 3 might be correct for soft handover purposes. It might not be true for other architectures, and we will make a separate calculation for that further below.

4) Assume that for each sample coming in, a DSP spends 1-2 cycles (running at 1GHz) to format it (if not done in PPS).

The DSP receiving the sample data, needs to perform this formatting before it does anything else, such as rake receiver processing. It does not know where to look at for individual users. So let's look at the resources it needs to spend for this:

Total samples/sec coming into DSP: 12 (antenna channels) * 2 (oversample) * 3.84 Msamples/s = 92.16 Msamples/s

If the DSP spends 1-2 cycles per sample = 184.32 Mcycles/sec spent on this. It has 1 Gcycles/sec available, so it used 18.4% of its resources.

For the sequential architecture (architecture 2a), only the front end DSP spends 18.4%, the rest do not see this. They might spend resources on other formatting functions, but since the data may be at a slower rate, they cycles spent may be negligible.

For the parallel architecture, each DSP on the BB cards saves this much, i.e 18.4% saving for each DSP when using the PPSc instead of doing this on the DSP itself.

Now let's change assumption 3 to each line card. Instead of receiving 12 AxC, they receive a subset of AxCs, lets say each receive 2:

Instead of receiving 12 antenna-channels, let's say each BB card receives 2 antenna channels, then the saving is 1/6th of before: 18.4/6 = ~ 3% from each DSP

Notes

All of these numbers go up if we change assumption 4. It is very optimistic to think that DSP spends only 1-2 cycle to format each sample. This might go up to 3 or 4 cycles depending on the number of memory access cycles, and number of operations. As an example:

3 cycles: Saving of 27.6%

4 cycles: Saving of 36.8%

Conclusion

In this application note, we outlined a reasonable progression of baseband architectures from today's sequential architecture to a more modular and scalable one based on sRIO. We also showed the benefits of the PPS and PPS-Lite for these architectures.

IDT has end to end solutions for both architecture 1 and 2. The PPS and PPS-Lite are the first products that will be followed by CPRI-sRIO and CPRI - Parallel TDM FICs as well as the Serial Buffer as an sRIO end point memory and FPGA interface to sRIO.

References

[1] Bertan Tezcan, Bill Beane, "Modular baseband design - enabling a low cost, reusable wireless infrastructure", Portable Design Magazine, January 06.

[2] Bertan Tezcan, Bill Beane, "Modular baseband design - enabling a low cost, reusable wireless infrastructure - Part II", Portable Design Magazine, January 06.

Revision History

8/01/06: Initial release.