SIMULTANEOUS BASEBAND PROCESSING CONSIDERATIONS IN A MULTI-MODE HANDSET USING THE SANDBLASTER™ BASEBAND PROCESSOR

Babak Beheshti, b.beheshti@ieee.org
Sanjay Jinturkar, Sandbridge Technologies, sanjay.jinturkar@sandbridgetech.com

ABSTRACT

The Sandbridge SB3011 baseband processor is a SoC containing four C programmable Sandblaster™ DSP processors, and all peripherals necessary to provide a single chip baseband/multimedia and application processing solution for SDR based handsets. This SoC uniquely provides 4 simultaneous digital I/Q DMA channels to the RF front end, allowing for simultaneous multi-mode baseband processing from multiple RF front ends. This paper discusses techniques and considerations in the software development to allow for a short design cycle for implementation of such multi-mode systems. Software partitioning considerations, multi-threading the application to take advantage of the multi-threaded architecture of the baseband processor, double buffering and pipelining, as well as parallel DMA processing are discussed. Concurrency matrix development for evaluation of simultaneous modes is also described.

1. INTRODUCTION

Sandbridge's flagship product is the SB3011 Flexible Baseband Processor. The SB3011 uniquely provides the capability to operate on any network and any communication protocol.
Featuring the integration of FOUR Sandblaster™ cores into a single SoC, the innovative Sandblaster™ architecture enables the SB3011 to implement the latest 3G protocols including W-CDMA, CDMA2000, and TD-SCDMA. Additionally, since the physical layers of these protocols are implemented in software, creating 'derivative' device designs is a relatively inexpensive software task, rather than a costly hardware integration effort.
The SB3011 features the Sandblaster™ DSP for execution of baseband in software - including physical layer. It has a programmable RF interface, with the capability to capture raw data at 100 million samples/sec. It includes interfaces to LCD, keypad, USIM, SmartCard, Audio codec, IrDA, plus emerging 'critical' features such as add-on memory cards, camera interface, and USB.

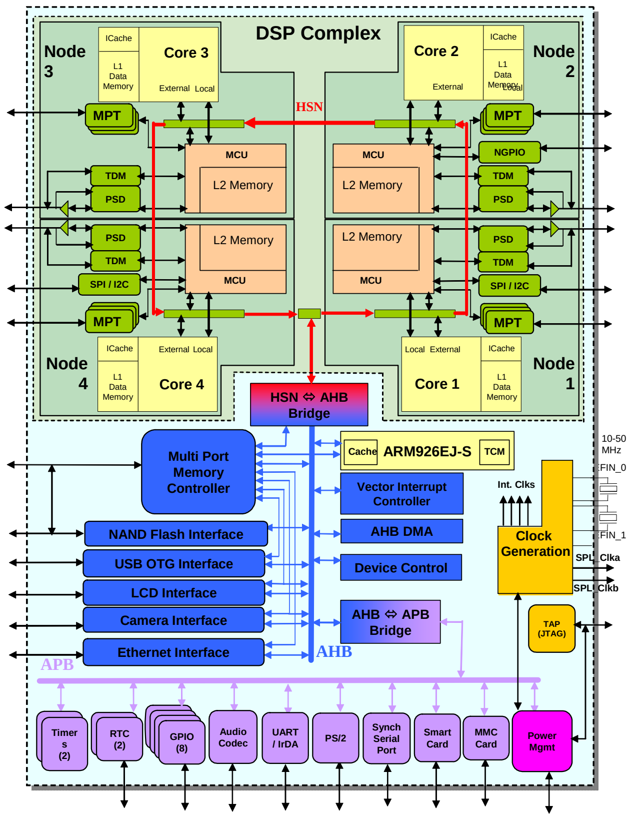


Figure 1 - SB3011 Internal Block Diagram

Figure 1 shows the SB3011 chip which has four SandBlaster™ DSP cores. The SB3011 has the following features:

- Low-power-consumption design
- Four SandBlaster™ DSP cores connected by a high speed bus in a ring topology
- Eight time-multiplexed hardware thread execution units for each DSP core
- SIMD/Vector operation unit
- 600MHz (1.67ns instruction cycle)
- 90nm CMOS process

- 32Kbytes instruction cache per core
- 64Kbytes L1 data memory per core
- 256Kbytes L2 data memory per core
- Integrated ARM processor

SB3011 DSP core has a SIMD vector operation unit which provides a parallel-execution capability of four 16-bit multiply-accumulate (MAC) operations per single instruction cycle.
The peripheral devices which require high-speed access are connected with Advanced High-Performance Bus (AHB) and most peripherals are connected via Advanced Peripheral Bus (APB). Both AHB and APB are compliant with Advanced Microcontroller Bus Architecture (AMBA).

2. PARALLEL STREAMING DATA (PSD) INTERFACE

All SandBlaster™ cores have a digital 16-bit PSD port that can perform either input or output transfers; the direction is set either externally, or programmed internally. Figure 2 shows a block diagram of the PSD. The PSD interface is the single most important feature that allows connection of multiple RF front ends to the SoC.

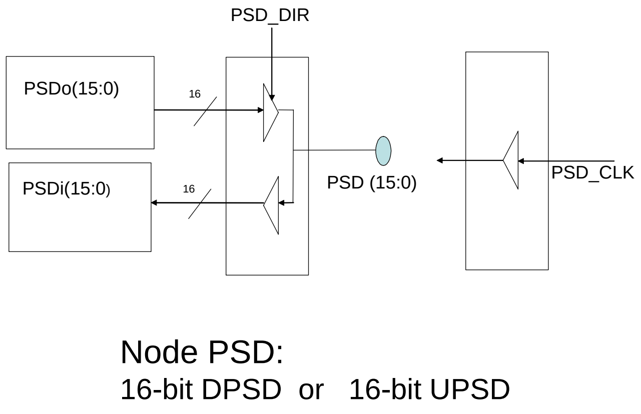


Figure 2: PSD Functional Block

2.1 Downlink Parallel Streaming Data DMA channel
The Downlink Parallel Streaming Data (DPSD) DMA interface provides the mechanism for capturing streaming data from an external parallel streaming data interface, e.g., an A2D converter. Data received through the device's parallel input port is expected in multiplexed form: I data sampled on the falling edge of the external interface clock; Q data sampled on the rising edge of the external interface clock.

2.2 Uplink Parallel Streaming Data DMA channel
This interface provides the mechanism for transmitting streaming data to an external parallel streaming data interface, e.g., a D2A converter. Data transmitted through the device's parallel output port will be generated in multiplexed form: I data is transmitted while the external interface clock is a logic '1'; Q data is transmitted while the external clock interface is low. For the external device receiving the data, I must be sampled in the falling edge, and Q must be sampled on the rising edge. The interface multiplexes and transfers the data from the memory I and Q buffers (as allocated by the programmer, through various control registers) to the external interface.

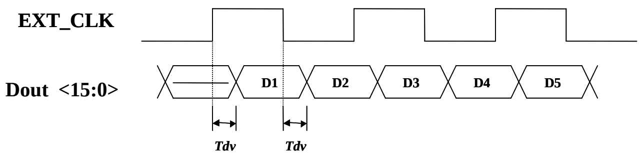


Figure 3: Digital to Analog Output Timing

2.3 Multi-Buffering in the PSD Interface

The PSD interface in both the receive and the transmit modes is based on a multi-buffered operation. For the Downlink PSD, the DMA transfers the input data (I & Q) to two separate buffers. Each of these two buffers are divided into an equal number of sub-buffers (Typically, in a program, the buffers are two dimensional arrays of shorts, where the first dimension indicates the size of the sub buffer and the second dimension indicates the number of sub-buffers). Once a sub- buffer is filled, an interrupt occurs. This interrupts can be polled using an operating system service call. In the Uplink PSD, data to be transmitted is put in appropriate buffers, and when that buffer is transferred to the RF interface for transmission, an interrupt is generated indicating that this buffer is free now.
Setting up a PSD DMA transfer in any direction involves setting of 4 Parameters:

1. Start address of the buffer where I data will be placed
2. Start address of the buffer where Q data will be placed
3. The number of sub-buffers inside each of the I & Q buffers

4. Size of each sub-buffer. This minimum size is 64 bits, and has to be aligned at a 64 bit boundary

The size and number of buffers in an implementation of a particular wireless standard are determined by a trade off between amount of memory allocated for PSD use, frequency at which the buffer transfer interrupts are to be services and consequently time available between consecutive interrupts to perform useful processing. Table 1 illustrates the symbol rates for several common wireless standards and the required sampling clock rates for the receive/transmit DMA assuming an over sampling rate of 4.

Standard	Chip/symbol Rate	Sample Rate (DMA Clock Rate)
WCDMA	3.8400E+06	1.54E+07
GSM	2.7083E+05	1.08E+06
TD-SCDMA	1.2800E+06	5.12E+06
TIA/EIA-95A/B	1.2288E+06	4.92E+06
cdma2000 (1xRTT)	1.2288E+06	4.92E+06
1xEV-DO	1.2288E+06	4.92E+06
1xEV-DV	1.2288E+06	4.92E+06
HSDPA	3.8400E+06	1.54E+07
TETRA	1.8000E+04	7.20E+04

Table 1 - Symbol Rate and Sampling Clock Rates for Some Standards

Naturally, the larger the buffer size, the longer time it takes to fill/empty that buffer before an interrupt is issued. This time period is available for the DSP engine to perform baseband processing operations on the data to be transmitted or received.
Table 2 shows a comparison of the number of interrupts per second required for a WCDMA (FDD) system based on various combinations of buffer sizes and number of buffers. The combinations are selected to consume 128k of memory consistently. Assuming a typical overhead of 1.33uSec for context switch and interrupt service by the operating system, the table illustrates the percentage of time the processor spends time processing interrupts. As seen in Figure 4 the selection of the buffer size should be somewhere between the extremes of too many interrupts, and very long buffers (that inhibit pipelining of processing chain with multi-buffering).

Buffer Size	Number of Buffers	Time between Buffer Transfer Interrupts	Number of Interrupts per Second	% Overhead
1024	128	6.67E-05	15000	0.0200%
2048	64	1.33E-04	7500	0.0100%
4096	32	2.67E-04	3750	0.0050%
8192	16	5.33E-04	1875	0.0025%
16384	8	1.07E-03	938	0.0013%
32768	4	2.13E-03	469	0.0006%
65536	2	4.27E-03	234	0.0003%

Table 2 – Comparison of Buffer Interrupt Processing overhead based on Buffer Size for WCDMA (FDD)

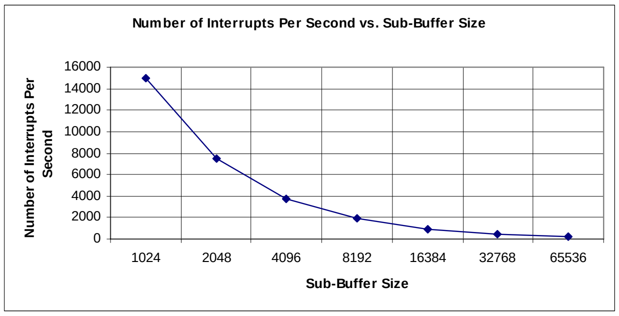


Figure 4: Number of Interrupts per Second vs. Buffer Size in a CDMA System (Total Memory Usage fixed at 128kBytes)

3. UNIVERSAL RF API FOR MULTIPLE STANDARDS

The SB3011 Board Support Package (BSP) contains an RF API designed to work with any wireless RF front end. This universal API allows for a consistent software design for interfacing RF front ends supporting different wireless standards. The RF API can be ported to any RF front end interfacing with the PSD ports as well as I2C and SPI ports for configuration. Examples of this universal API are:

RF_Drv_Rf_Error_t Drv_Rf_Set_Afc (int afc_val)
This API sets the value of AFC.

Parameters:	afc_val
Units:	Hz

Return Value:
- RF_DRV_RF_SUCCESS
- Or Error code

RF_Drv_Rf_Error_t
Drv_Rf_Set_PA_Level (int pa_level)
This API sets the value of Power Amplifier level.
Parameters: pa_level
Units: dB
Range: -55 to +23 dBm (approx.)
Return Value:

- RF_DRV_RF_SUCCESS
- Or Error code

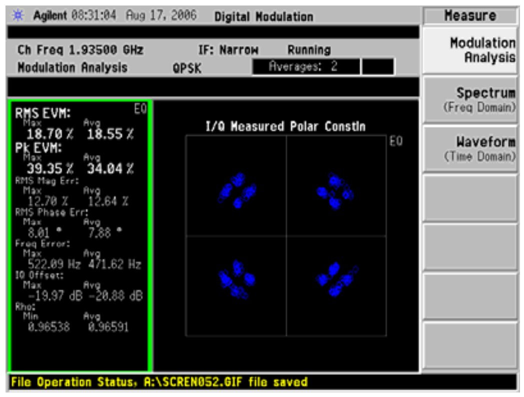
RF_Drv_Rf_Error_t
Drv_Rf_Set_Transmit_Channel (int channel_num)
With this API, through the SPI interface (for instance), one programs the LO synthesizer to set the uplink channel, then checks the PLL lock detect interrupt status.
Return Value:

- RF_DRV_RF_SUCCESS
- Or Error code

An example piece of C code using this API follows:

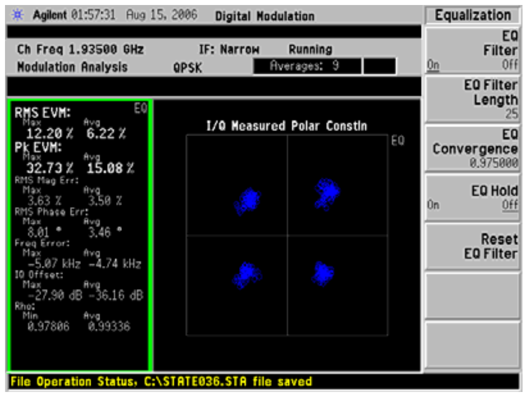
```
Rf_Init_all();
Drv_Rf_Switch_Transmit(0);//turns on transmitter
( 0 turns Tx off)
Drv_Rf_Switch_Receive(1);// Turns ON Rx (0 turns
off Rx)
Drv_Rf_Set_Receive_Channel(VAL_D);// Set Rx freq
band to 2140 MHz
Drv_Rf_Set_Rx_Agc(VAL_LNA,VAL_AGC);
```

Figure 5, Figure 6 and Figure 7 show captures of an EV-DO transmitter, a QPSK transmitter and Spectrum of a QPSK received signal, respectively. These figures demonstrate the utility of the universal RF API is interfacing with multiple RF standard front ends.



```
Agilent 08:31:04  Aug 17, 2006    Digital Modulation
Ch Freq 1.93500 GHz        IF: Narrow     Running
Modulation Analysis        QPSK           Averages: 2
RMS EVM:
  Max       Avg
  18.70 %   18.55 %
Pk EVM:
  Max       Avg
  39.35 %   34.04 %
RMS Mag Err:
  Max       Avg
  12.70 %   12.64 %
RMS Phase Err:
  Max       Avg
  8.01 °    7.88 °
Freq Error:
  Max         Avg
  522.09 Hz   471.62 Hz
IQ Offset:
  Max         Avg
  -19.97 dB   -20.88 dB
Rho:
  Min       Avg
  0.96538   0.96591
File Operation Status, A:\SCREN052.GIF file saved
```

Figure 5 - EVDO TX EVM with QPSK PSA (0dBm signal)



```
Agilent 01:57:31  Aug 15, 2006    Digital Modulation
Ch Freq 1.93500 GHz        IF: Narrow     Running
Modulation Analysis        QPSK           Averages: 9
RMS EVM:
  Max       Avg
  12.20 %   6.22 %
Pk EVM:
  Max       Avg
  32.73 %   15.08 %
RMS Mag Err:
  Max       Avg
  3.63 %    3.50 %
RMS Phase Err:
  Max       Avg
  8.01 °    3.46 °
Freq Error:
  Max         Avg
  -5.07 kHz   -4.74 kHz
IQ Offset:
  Max         Avg
  -27.90 dB   -36.16 dB
Rho:
  Min       Avg
  0.97806   0.99336
File Operation Status, C:\STATE036.STA file saved
```

Figure 6 - QPSK 24 dBm

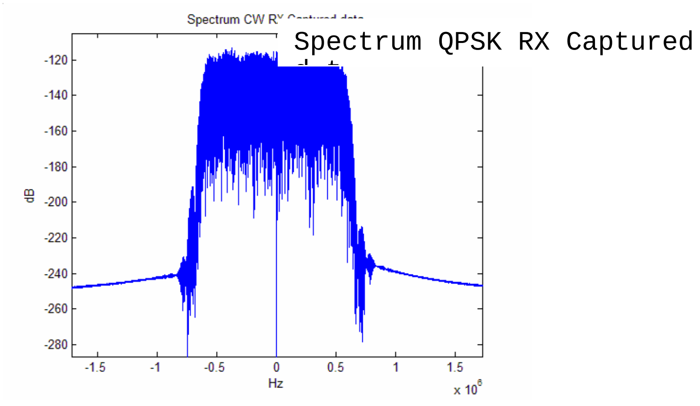


Figure 7 - QPSK Rx spectrum from captured data at -55 dBm

4. OPERATING SYSTEM SUPPORT

The operating system provides API's to transmit or receiver the data samples from the PSD's on the processor. These API's enable a user to control the amount per transfer, its direction, and its location. A sample of the low level API's are described here:

int osIqOutInit(void): This is called in the beginning of the DMA transfer to initialize the transmit DMA..

int osIqInStart(void *iaddr, void *qaddr, unsigned int nbufs, unsigned int bufsize)
This transfers the input data (I & Q) to two separate buffers. Each of these two buffers are divided into an equal number of sub-buffers [Typically, in a program, the buffers are two dimensional arrays of shorts, where the first dimension indicates the size of the sub buffer and the second dimension indicates the number of sub-buffers]. Once a sub- buffer is filled, an interrupt occurs. This interrupt should be cleared as soon as possible.
Parameters:

`iaddr`: Start address of the buffer where I data will be placed
`qaddr`: Start address of the buffer where Q data will be placed
`nbufs`: The number of sub-buffers inside each of the I & Q buffers
`bufsize`: Size of each sub-buffer. This minimum size is 64 bits, and has to be aligned at a 64 bit boundary

int osIqinStop(void)
This stops the A/D from filling up the input buffer

The operating system support of the DMA transfers from and to the RF front end allow for rapid application development by reusing legacy code, only modifying the API parameters.

5. SIMULATOR SUPPORT FOR MULTIPLE RF INTERFACES

The BSP for SB3011 provides a simulation environment within which the 4 PSD operation can be validated in absence of any hardware platform. The simulator can be configured for a specific sampling clock rate. Once the program is simulated, the developer can easily verify whether any of the 4 PSD interfaces meet their expected deadlines, and no buffer overrun occurs.

A sample configuration for the simulator is:
IQInput {trace 1} IQInput:2 {trace 0} IQInput:3 {file "foo.txt"}'

This will set tracing to 1 for all 4 IQInput (A2D) peripherals, override for core 2 and set its tracing back to 0, and finally tell IQInput on core 3 to use foo.txt as the input file.

10. CONCURRENCY MATRIX

Table 3 is a concurrency matrix for a number of wireless standards running on the SB3011 processor. Each of these standards consumes a percentage of the processor's processing power. These numbers have been compiled experimentally by implementation of physical layer protocols for the standard in fixed point ANSI C. The intersection of each column and row is a possible combination of two standards running concurrently on the SB3011. As long as the number in the matrix is less than 100 (percent), the two standards can run in real time in parallel. The diagonal of the matrix is blank, because the row and column intersects in the diagonal refer to the same standard. The matrix here has been presented as a guideline. It is not all inclusive of all possible multimedia and other standards that can run on the platform as a software application.

11. SUMMARY

In this paper we have illustrated a development environment, consisting of a BSP, operating system and a simulator that support a multi-RF, multi-protocol software based baseband processing. The rich development environment allows for a rapid development cycle, providing immediate feedback to the developer as to real time behavior of the baseband processing.

11. REFERENCES

[1] B. Beheshti, T. Raja, "Software Defined Radio Implementation Considerations and Principles Using the Sandblaster™ SDR Baseband Processor", Proceedings of Software Defined Radio Technical Forum, 16-18 November, 2005, Anaheim, California.
[2] B. Beheshti, "A Study of the Technology Migration Path of the Cellular Wireless Industry from 3G to 3.5G and Beyond", Proceedings of 2005 IEEE Long Island Systems, Applications and Technology Conference (LISAT2005), May 2005, Farmingdale, New York.http://www.sdrforum.org
[3] D. Iancu, J. Glossner, V. Kotlyar, H. Ye, M. Moudgill, and E. Hokenek, "Software Defined Global Positioning Satellite Receiver", Proceedings of the 2003 Software Defined Radio Technical Conference (SDR'03), HW-2-001, 6 pages, Orlando, Florida, 2003.
[4] J. Glossner, D. lancu, J. Lu, E. Hokenek, and M. Moudgill, "A Software Defined Communications Baseband Design", IEEE Communications Magazine, Vol. 41, No.1, pp. 120-128, Jan., 2003.
[5] J. Glossner, D. Iancu, V. Kotylar, H. Ye, E. Hokenek, and M. Moudgill, "Software Defined Global Positioning Satellite Receiver", Proceedings of Software Defined Radio Technical Forum, pp. HW-2-001, 1-5, Orlando, Florida, November 2003. .
[6] J. Glossner, S. Dorward, S. Jinturkar, M. Moudgill, E. Hokenek, M. Schulte, and S. Vassiliadis, "Sandbridge Software Tools", in Proceedings of the 3rd International Workshop on Systems, Architectures, Modeling, and Simulation (SAMOS.p3), July 21-23,2003, pp. 142-147, Samos, Greece.
[7] J. Glossner, T. Raja, E. Hokenek, and M. Moudgill, "A Multithreaded Processor Architecture for SDR", The Proceedings of the Korean Institute of Communication Sciences, Vol. 19, No. 11, pp. 70-84, November, 2002.

[8] J. Glossner, E. Hokenek, and M. Moudgill, "Multithreaded Processor for Software Defined Radio", Proceedings of the 2002 Software Defined Radio Technical Conference, Volume I, pp. 195-199, November 11-12, 2002, San Diego, California.

	Analog TV	Bluetooth	GPRS1	GPRS2	WCDMA1	WCDMA2	WCDMA3	WiMax	CDMA1	CDMA2	DVB-H1	DVB-H2	DVB-H3	DVB-H4	DVB-H5	DVB-H6
Analog TV		35	34	50	96	100	108	105	61	111	36	52	38	57	40	63
Bluetooth	35		19	35	81	85	93	90	46	96	21	37	23	42	25	48
GPRS1	34	19			80	84	92	89	45	95	20	36	22	41	24	47
GPRS2	50	35			96	100	108	105	61	111	36	52	38	57	40	63
WCDMA1	96	81	80	96				151	107	157	82	98	84	103	86	109
WCDMA2	100	85	84	100				155	111	161	86	102	88	107	90	113
WCDMA3	108	93	92	108				163	119	169	94	110	96	115	98	121
WiMax	105	90	89	105	151	155	163		116	166	91	107	93	112	95	118
CDMA1	61	46	45	61	107	111	119	116			47	63	49	68	51	74
CDMA2	111	96	95	111	157	161	169	166			97	113	99	118	101	124
DVB-H1	36	21	20	36	82	86	94	91	47	97						
DVB-H2	52	37	36	52	98	102	110	107	63	113						
DVB-H3	38	23	22	38	84	88	96	93	49	99						
DVB-H4	57	42	41	57	103	107	115	112	68	118						
DVB-H5	40	25	24	40	86	90	98	95	51	101						
DVB-H6	63	48	47	63	109	113	121	118	74	124						

Table 3 - Concurrency Matrix for Several Wireless Standards Running on SB3011