COM-8004SOFT
SIGNAL DIVERSITY SPLITTER
VHDL SOURCE CODE OVERVIEW

Overview

The COM-8004 ComBlock Module comprises two pieces of software:

- VHDL code to run within the FPGA for all signal processing functions.
- C/Assembly code running within the Atmel ATMega8515L microprocessor for non application-specific monitoring and control functions.

The VHDL code interfaces to the monitoring and control functions by exchanging byte-wide registers on the Atmel microcontroller 8-bit data bus. The control and monitoring registers are defined in the specifications [1].

The Atmel microprocessor code is generic (i.e. non application specific), not user-programmable and functionally transparent to the user. It is thus not described here.

The COM-8004 VHDL code runs on the generic COM-8000 hardware platform. The schematics [2] for this platform are available in this CD.

Reference documents

[1] specifications: com8004.pdf

[2] hardware schematics: com_8000schematics.pdf

[3] VHDL source code in directory
com-8004_003\src

[4] Xilinx ISE project files
com-8004_003\com-8004_ISE41.npl
com-8004_003\com-8004_ISE63.npl

[5] .ucf constraint files
com-8004_003\src\com8004.ucf

[6] .mcs FPGA bit files
8004_003\com8004_003.mcs

Configuration Management

The current software revision is 3.

Configuration Options

No option currently supported.

VHDL development environment

The VHDL software was developed using two development environments:

(a) Xilinx ISE 4.1 with Synopsis FPGA Express 3.6 as synthesis tool.
(b) Xilinx ISE 6.3 with XST as synthesis tool.

Target FPGA

The VHDL code was synthesized for the Xilinx Spartan-IIE XC2S300E-6PQ208 FPGA.

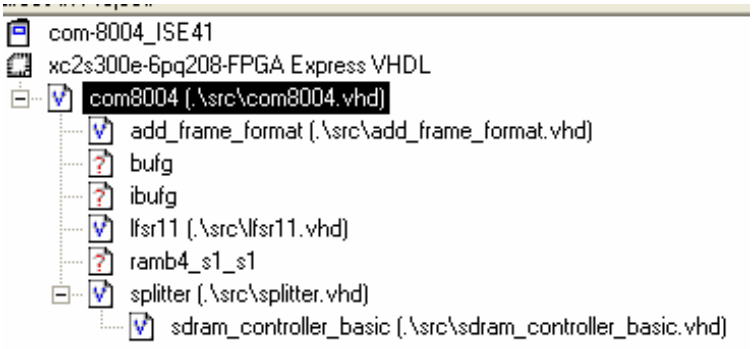
Xilinx-specific code

The VHDL source code was written in generic VHDL with few Xilinx primitives. No Xilinx CORE is used. The Xilinx primitives are:

- BUFG
- IBUFG
- RAMB4_S1_S1

MSS • 18221 Flower Hill Way #A • Gaithersburg, Maryland 20879 • U.S.A.
Telephone: (240) 631-1111 Facsimile: (240) 631-1676 www.ComBlock.com
 Issued 3/25/2005

VHDL software hierarchy

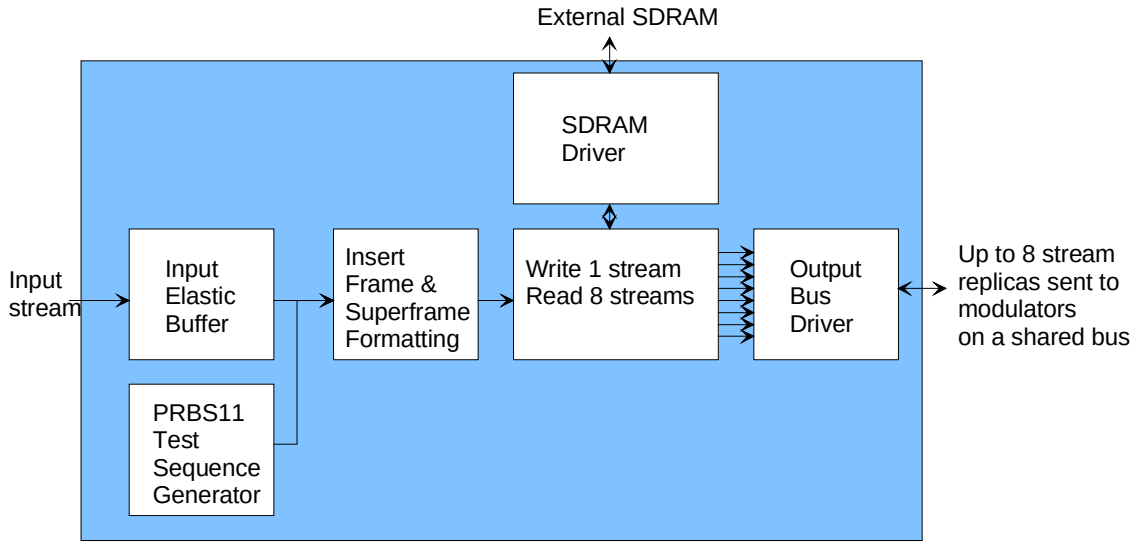


The code is stored with one, and only one, entity per file as shown above.

The root program (highlighted) is *com8004.vhd.*

The hierarchical nature of the VHDL code reflects the block diagram below:

- *com8004* is the root program which includes the input elastic buffer, the frame and superframe format insertion in *add_frame_format*, the PRBS-11 test sequence generator *lfsr11*, the splitter and SDRAM management functions in *splitter*, the output bus driver and ancillary functions such as monitoring and control functions (interface with microprocessor).
- The *add_frame_format* entity takes the input stream or the internally generated test stream and adds periodic frame synchronization markers called unique words. Each frame consists of 4096 bits of data preceded by the 32-bit unique word 0x5A 0F BE 66. The superframe length is user-defined. It is an integer number of frames. The superframe unique word is the inverted version of the regular frame unique word.
- The PRBS-11 pseudo-random test pattern is generated within the *lfsr11.vhd* entity.
- The *splitter* entity takes a single input stream and generates 8 replicas subject to user-defined delays. Delays are always integer multiples of the superframe length. Large delays can be implemented within the external 256MB SDRAM.
- The *sdram_controller_basic* entity is the driver for the external 256MB SDRAM. It manages the SDRAM initialization, the periodic refresh cycles and random access read and write cycles.



COM-8004 Software Functional Block Diagram

Implementation

Data Flow

The input data stream is first formatted with periodic frame and superframe synchronization markers. The resulting serial stream is converted from 1-bit serial to 64-bit parallel prior to writing to the SDRAM. The stream is written to the SDRAM one 64-bit word at a time. After each write transaction, the SDRAM write pointer (address) is incremented. The SDRAM is a circular buffer: once the write pointer reaches the last address in the 256MB memory, the write pointer is reset to the zero start address.

Each write transaction is immediately followed by up to eight read transactions. The exact number depends on the number of replicas desired (equal to the number of active modulators following this module). The user selects the number of active modulators/output streams in control register REG0.

The offset between the write pointer and a given read pointer represents a delay. Delays are user-defined independently for each stream, in control registers REG2 through REG15. Delays are always integer multiple of superframe lengths.

Output stream 0 is not subject to any delay. Thus, the read pointer is equal to the write pointer. For all other streams, the read pointer lags the write pointer (i.e. the offset is subtracted from the write pointer value to obtain the read pointer).

The eight 64-bit words read from the SDRAM (one for each output replica) are subsequently converted to eight 1-bit serial output streams.

The output streams are conveyed to the external modulators over a 16-bit shared bus. The COM-8004 is the bus master. As such, it controls the 4-bit bus address BUS_ADDR, the transaction type BUS_RWN, and the sequencing. This simple bus does not support interrupts.

Although the bus data width is 16-bit wide, each output data stream is currently transferred one bit at a time. A higher degree of parallelism can be easily implemented in the future if higher data rates are needed.

Flow Control

Like in most transmitters, the data rate is set by the modulator (here active modulators), then is propagated upstream using flow-control signals. The COM-8004 propagates the flow control information upstream by implementing the following steps:

- The flow-control signal SAMPLE_CLK_OUT_REQ is read from each active modulator over the shared data bus.
- Data is transferred from a 128-bit output elastic buffer to an active modulator if data is available (output elastic buffer is not empty) and if SAMPLE_CLK_OUT_REQ = ‘1’.
- A SDRAM read/write cycle is triggered if and only if
    - All active output elastic buffers have room to accept the next 64-bit word from the SDRAM.
    - The 128-bit input elastic buffer contains a 64-bit word ready to be written to the SDRAM.

  Upon trigger, the *splitter* entity conducts one 64-bit word SDRAM write transaction followed by several 64-bit SDRAM read transactions, one for each active modulator.
- The input elastic buffer requests additional input data from the *add_frame_format* entity when there is room for at least 32 additional bits.
- The *add_frame_format* entity forwards this request upstream to the input or to the *lfsr11* entity while the frame data field is being assembled.

The COM-8004 is fundamentally different from single-stream transmitters in that it forwards data to several (up to eight) modulators. For the flow control mechanism to work properly, it is essential that the following requirements be met:

- All active modulators must operate at the exact same data rate.
- Consequently, all active modulators must use the same reference clock.

SDRAM

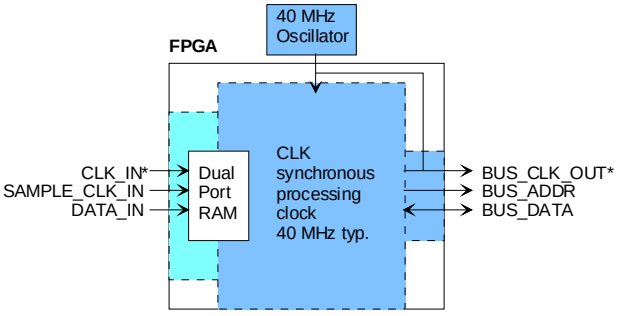
The code is written for interfacing with a generic 256MB SDRAM SODIMM memory module. A set of specifications can be obtained from Micron (MT48LC16M16A2 – 4 MEG X 16 X 4 BANKS). The clock provided to the SDRAM is 40 MHz, well below the specified maximum for PC100/PC133 SDRAMs. The read and write pointers representing SDRAM addresses are written with 27-bit precision so as to allow easy extension to 1GB SDRAM memory modules if needed.

In order to minimize the number of FPGA flip-flops used for serial to parallel conversions, a basic SDRAM write transaction deals with a single 64-bit word. Burst mode, although conducive to a higher SDRAM throughput, is not used because of the extra FPGA resources required.

In addition to the read, and write operations, the *sdram_controller_basic* entity takes care of the initial SDRAM initialization, of periodic auto refresh and of the contention avoidance mechanism between the asynchronous read/write transactions and the auto refresh transaction.

Clock / Timing

The clock distribution scheme embodied in the COM-8004 is illustrated below.



Baseline clock architecture
Light blue = user defined input clock
Darker blue = internal 40 MHz clock
**** indicates edge-trigger signal***

The core signal processing performed within the FPGA is synchronous with the 40 MHz processing clock locked onto a 40 MHz internal oscillator. The processing clock is not related to the CLK_IN clock.

A 4096-bit dual-port RAM elastic buffer is used at the boundary between input and internal processing area. Thus, the input clock frequency can be independent from the internal processing clock frequency.

The input signals at the J1 input connector are synchronous with the CLK_IN. This clock can be up to 40 MHz.

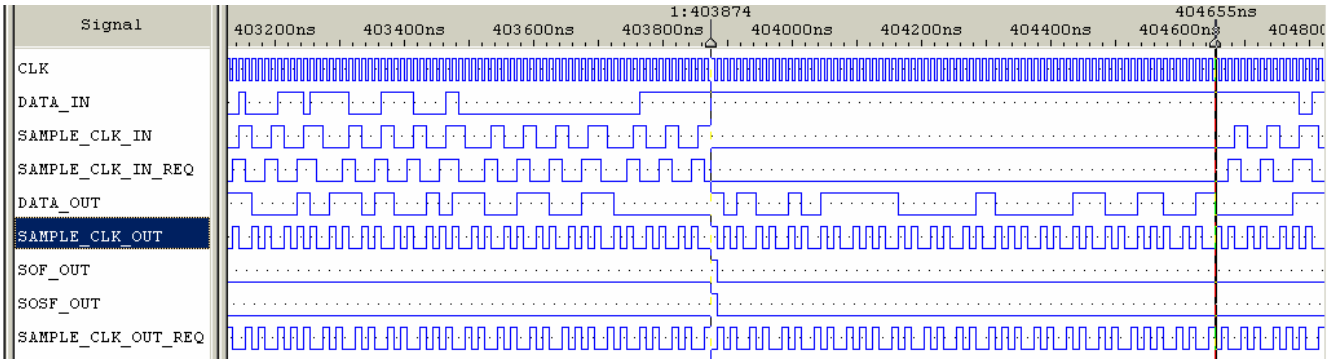
The output signals are synchronous with the rising edge of the 40 MHz reference clock BUS_CLK_OUT (i.e. all signals are stable at the rising edge of the reference clock BUS_CLK_OUT).

VHDL Simulation

Representative simulation screens for salient internal signals are captured and discussed below.

Frame format insertion

The *add_frame_format* entity inserts periodic start-of-frame synchronization markers to form fixed-length (4128-bit long) frames. The 32-bit unique word is 0x5A0FBE66, transmitted most-significant bit first. The unique word is inverted at the start of a superframe as illustrated in the simulation capture below. The simulation captured the key input and output signals for the *add_frame_format* entity. The DATA_OUT output stream includes a start-of-superframe unique word between the blue and red time cursors.

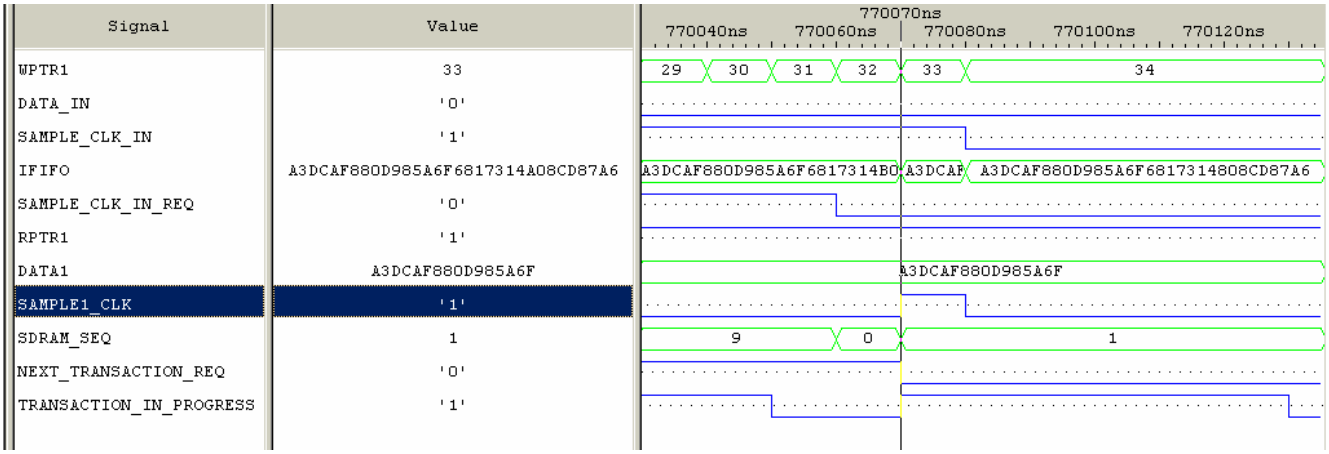


Frame format insertion

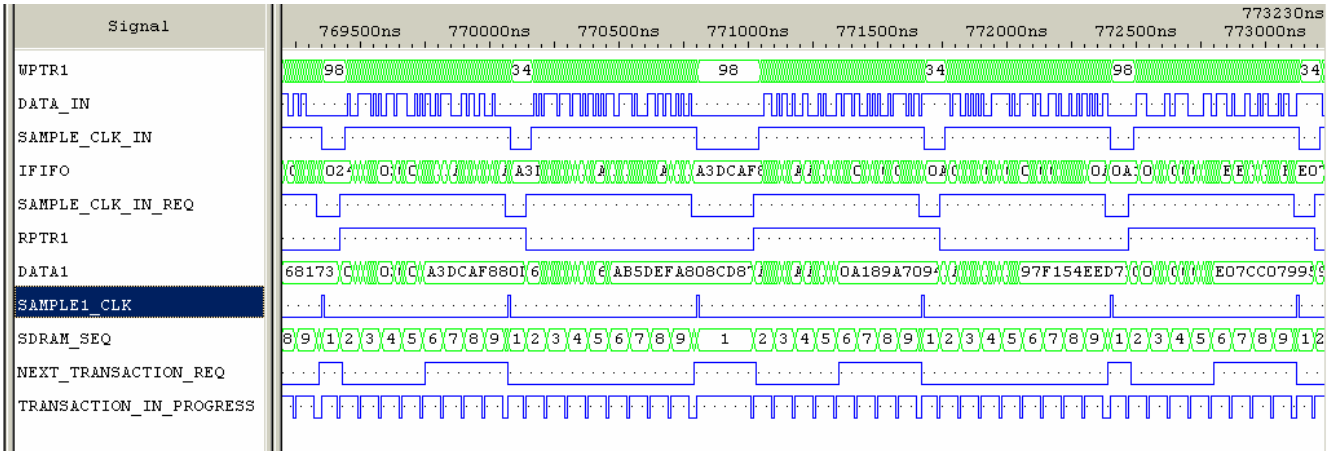
Input elastic buffer

The first function within the *splitter* entity is to perform a serial to 64-bit parallel conversion through the 128-bit elastic buffer IFIFO. Each input bit DATA_IN is written into IFIFO(WPTR1) when SAMPLE_CLK_IN = ‘1’, then the circular write pointer WPTR1 is incremented. Data is read from IFIFO 64-bit at a time into DATA1 from the upper half or lower half of IFIFO, depending on the read pointer RPTR1.

The flow control algorithms requests input data (SAMPLE_CLK_IN = ‘1’) when the IFIFO elastic buffer has room for at least 32 bits from the data source.



splitter.vhd input elastic buffer (zoomed in)



splitter.vhd input elastic buffer (zoomed out)

SDRAM write/read sequence

The second function within the *splitter* entity is to sequence one (64-bit) write transaction and up to eight (64-bit) read transactions with the external SDRAM memory module. This is the basic process by which delayed replicas are created from the original input stream. The SDRAM_SEQ signal indicates the sequence phase: 0 for idle, 1 for the single 64-bit write transaction, 2 through 9 for the follow-on read transactions.

splitter.vhd basic 1write/8read sequence (all 8 output streams enabled).

The simulation capture above shows that the SDRAM sequence exits the idle (SDRAM_SEQ = 0) condition when the following flow-control conditions are met:
(a) the input elastic buffer contains 64-bit ready to be read
(b) each enabled output elastic buffer has room to store a 64-bit word.
When this condition occurs, the NEXT_TRANSACTION_REQ flag goes high.

SDRAM controller

The VHDL code is designed to operate in conjunction with industry-standard 256MB SDRAM SODIMM modules. The address field is 27 bit long for compatibility with 1GB SDRAM SODIMM (just change the SDRAM_SIZE_MAX constant in the *com8004* root entity.

The SDRAM_controller_basic entity is dedicated to implementing random-access read and write transactions. Each transaction moves a single 64-bit word. The SDRAM mode register is thus initialized as follows:

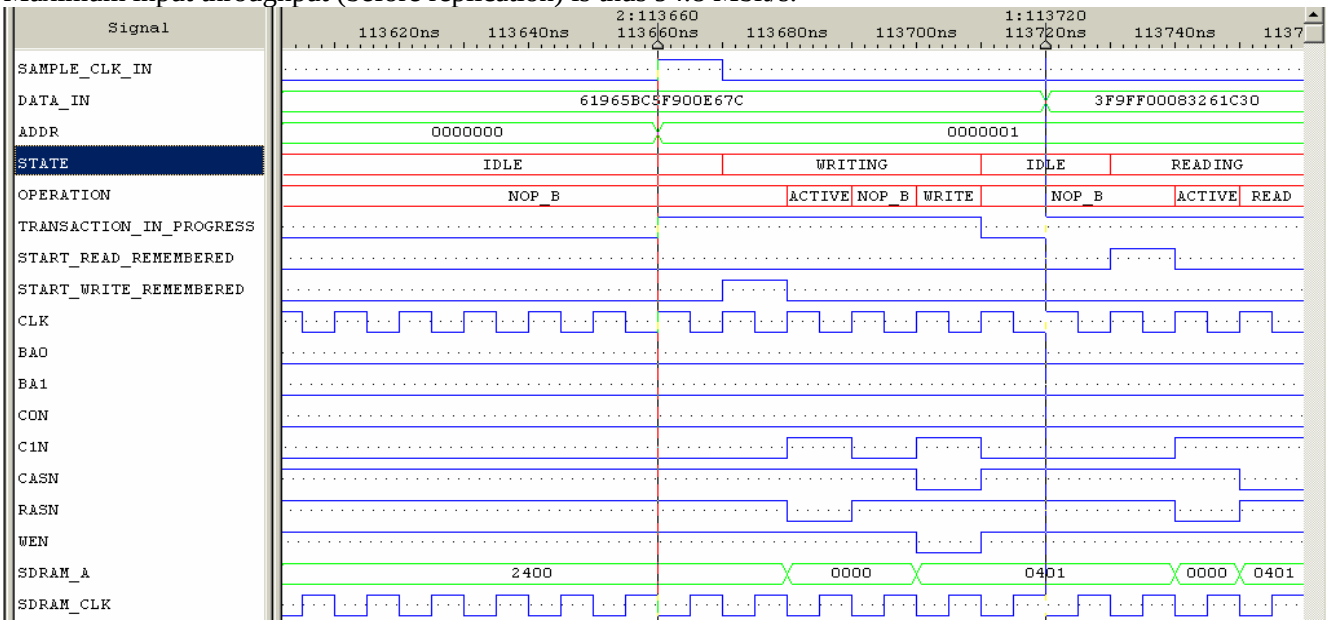
```
SDRAM_A(11 downto 10) <= "00";          -- Reserved
SDRAM_A(9) <= '1';                      -- single location access
SDRAM_A(8 downto 7) <= "00";            -- Op mode, always "00"
SDRAM_A(6 downto 4) <= "010";           -- CAS Latency, 2
SDRAM_A(3) <= '0';                      -- Sequential burst type
SDRAM_A(2 downto 0) <= "000";           -- Burst Length 1
```

All SDRAM transactions are synchronous with the 40 MHz CLK reference clock (significantly below the PC133 capability). The reference clock supplied to the SDRAM is inverted with respect to the FPGA reference clock CLK: the FPGA reads and writes at the rising edge of CLK while the SDRAM reads and writes at the falling edge of CLK to avoid race conditions.

An auto-refresh transaction is triggered every 15.6 us. The auto-refresh transaction has priority over read and write transaction requests. One read request and one write request can be queued while waiting for the current transaction to complete.

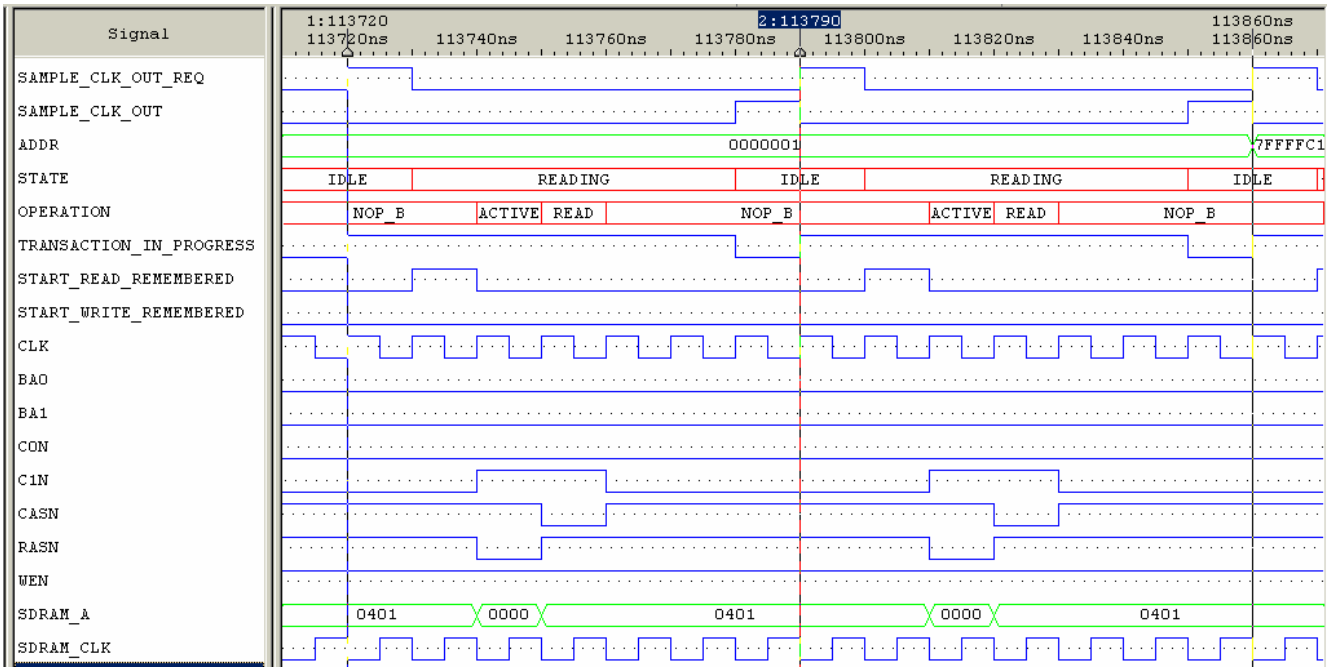
The write transaction comprises four elementary operations: ACTIVE, NOP_B, WRITE, NOP_B.
The read transaction comprises five elementary operations: ACTIVE, READ, NOP_B, NOP_B, NOP_B.
The auto-refresh transaction comprises nine elementary operations: PRECHARGE, NOP_B, AUTO_REFRESH_B, NOP_B, NOP_B, AUTO_REFRESH_B, NOP_B, NOP_B, NOP_B.

The maximum throughput at the SDRAM is the result of a tradeoff between the number of flip-flops used in elastic buffers surrounding the SDRAM and burst mode length. We chose a single 64-bit word burst to minimize the number of flip flops. The resulting maximum throughput is computed below in the case of 3 output replicas:
One 64-bit write transaction (inclusive of all latencies): 6 CLKs
Three 64-bit read transactions (inclusive of all latencies): 21 CLKs.
Total: 27 CLKs @ 40 MHz for a 64-bit data word.
Maximum input throughput (before replication) is thus 94.8 Mbit/s.



SDRAM_controller_basic: single 64-bit word write transaction (between cursors)

The write transaction is triggered by the SAMPLE_CLK_IN pulse. This causes the STATE to go from IDLE to WRITING. Consequently, OPERATION goes through the writing sequence ACTIVE, NOP_B, WRITE, NOP_B. An operation is defined as a specific combination of CS#, RAS#, CAS#, WE#, DQM, ADDR#, DQ. See SDRAM specifications for details.



SDRAM_controller_basic: two successive 64-bit word read transactions (between cursors)

The read transaction is triggered by the SAMPLE_CLK_OUT_REQ pulse. This causes the STATE to go from IDLE to READING. Consequently, OPERATION goes through the reading sequence: ACTIVE, READ, NOP_B, NOP_B, NOP_B.

The SDRAM is initialized for a CAS latency of 2 CLKs after the READ command. Thus, the read data (and SAMPLE_CLK_OUT) is available at the output two clocks after the READ command (it is reclocked first at the falling edge of CLK, then at the rising edge of CLK).

Output Data Bus

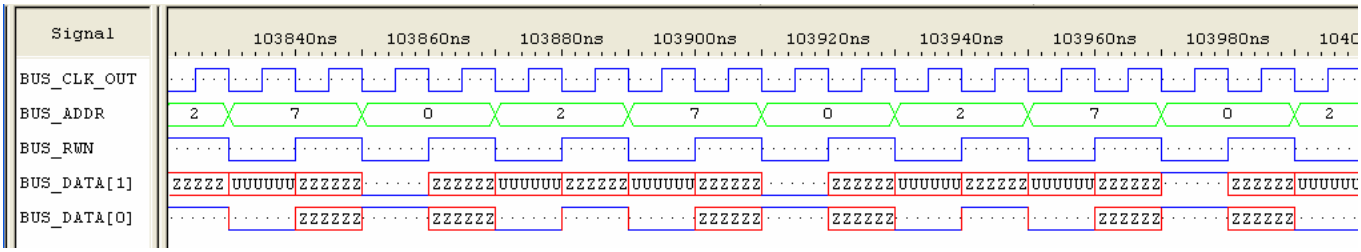
Definition:

Output Module Interface	**Definition**
BUS_CLK_OUT	40 MHz output reference clock for use on the synchronous bus.
BUS_ADDR[3:0]	Bus address. Output (since this module is the bus Master). Designates which slave module is targeted for this read or write transaction. All 1's indicates that the write data is to be broadcasted to all receiving slave modules. Read at the rising edge of BUS_CLK_OUT.
BUS_RWN	Read/Write#. Output (since this module is the bus Master). Indicates whether a read (1) or write (0) transaction is conducted. Read at the rising edge of BUS_CLK_OUT. Read and Write refer to the bus master's perspective.
BUS_DATA[15:0]	Bi-directional data bus. Output when BUS_RWN='0'. Input when BUS_RWN='1'. Read data latency is 2 clock periods after the read command. Functional definition during write: • bit 0 SAMPLE_CLK_OUT. '1' when DATA_OUT is available • bit 1 DATA_OUT data stream to modulator.

	• bits(15:2) undefined Functional definition during read: • bit 0 SAMPLE_CLK_OUT_REQ requests data from the source. Used for flow control. • bits(15:1) undefined

The COM-8004 is capable of forwarding multiple data streams to multiple modulators over a shared data bus. The COM-8004 is bus 'master', i.e. it drives the BUS_CLK_OUT, BUS_ADDR, BUS_RWN signals. The 16-bit BUS_DATA is bi-directional. The simulation below assumes that three modulators are available to receive stream 0,2 and 7 respectively. For simplicity, only one modulator, responding to the bus address 0, is simulated.

For each stream, a write transaction is immediately followed by a read transaction. During the write transaction, BUS_DATA(1) conveys the data if available, and BUS_DATA(0) indicates whether BUS_DATA(1) includes data. During the read transaction, BUS_DATA(0) indicates whether the recipient modulator is ready to accept further input data bits.



Data bus simulation. COM-8004 -> COM-1019 DSSS modulator

FPGA Occupancy

```
Design Summary
--------------
   Number of errors:      0
   Number of warnings:    3
   Number of Slices:                1,744 out of  3,072   56%
   Number of Slices containing
      unrelated logic:                  0 out of  1,744    0%
   Number of Slice Flip Flops:      1,788 out of  6,144   29%
   Total Number 4 input LUTs:       1,989 out of  6,144   32%
      Number used as LUTs:                           1,933
      Number used as a route-thru:                      56
   Number of bonded IOBs:             129 out of    142   90%
      IOB Flip Flops:                                 221
   Number of Block RAMs:                1 out of     16    6%
   Number of GCLKs:                     4 out of      4  100%
   Number of GCLKIOBs:                  4 out of      4  100%
Total equivalent gate count for design:  45,263
Additional JTAG gate count for IOBs:  6,384
```

Acknowledgments

This product development was funded by SBIR Phase II research contract "Signal Diversity Combining for Improved Satellite Communications" awarded by the US Air Force Research Laboratory, Rome, NY.

Contact Information

MSS • 18221 Flower Hill Way #A •
Gaithersburg, Maryland 20879 • U.S.A.
Telephone: (240) 631-1111
Facsimile: (240) 631-1676
E-mail: info@comblock.com