TS Data Tagger Module

5/20/19
E.J.
(preliminary)

Normal TS-ROC Communication

The Trigger Supervisor (TS) makes information about the trigger it accepted available to the Read Out Controllers (ROC) in the form of a 4-bit (V1 TS) or 6-bit (V2 TS) *Event Type* that is generated from the trigger lookup memory. The trigger pattern accepted serves as an address to this memory; the data of the memory is programmed by the user. Like data from the front-end modules, this data is presented to the ROCs in buffered fashion. The TS supports a total of 32 ROCs on its four independent ROC branches. Each branch consists of an 8-deep buffer memory (FIFO), buffer counter, and read sequencer on the TS, and an external cable that links up to 8 ROC interface cards. The *Event Type* and two status flags (*Sync, Late Fail*) form the data transmitted along the branch (6 or 8 bits total).

The communication of trigger information occurs concurrently and independently on all four branches. Once the read sequencer on a branch notices that its buffer is not empty, the transmission of trigger information for the event to the ROC interface cards begins. The read sequencer places the *Event Type* and status flags from the first valid buffer location onto data lines *Data(0-7)* (V2) and qualifies these with *Strobe*. The ROCs along the branch receive this data and proceed to read out the event fragments according to the function defined by the *Event Type*. Each ROC is assigned a unique *Acknowledge* line on the branch cable. When a ROC on the branch is finished processing the event it asserts *Acknowledge*. Upon receipt of *Acknowledge* from all enabled ROCs on the branch, the read sequencer negates *Strobe* and decrements its buffer counter. Each ROC negates its *Acknowledge* upon detecting the negation of *Strobe*. The read sequencer cycle is completed when it detects that *Acknowledge* has been negated by all enabled ROCs on the branch. These cycles on the branch will continue as long as there is valid trigger data available in the buffer.

For systems that have front-end modules with no buffering capability or if buffering is not desired, the depth of the branch buffers can be set to 1 (*lock* mode). In this configuration the TS is able to accept new triggers only when the previous event has been completely read out. In *lock* mode the communication of trigger information still occurs independently on all four branches, but all branches are working concurrently on the same trigger. (This is not necessarily the case when the TS is operating in *buffered* mode.)

TS Data Tagger Module

An application has arisen where it is crucial to confirm that data read from the distributed ROCs originated from the same trigger. A simple idea would be for the TS to send the ROCs an *event ID* with each *Strobe* rather than an event type based on the trigger accepted. With up to 8 data bits carried on the branch cable, up to 256 unique *event ID*s would be available. Consecutive triggers would be forced to have different *event ID*s (for example a simple counting sequence). When the complete event is built from the ROC event fragments, the *event ID*s from the distributed ROCs can be compared to verify synchronization. Other methods such as timestamping data in each ROC are also possible and can be pursued in parallel.

It would be difficult to modify the existing TS to perform the function described above. It is possible to achieve the desired behavior by interposing a new Data Tagger Module between the TS and the ROCs. Located close to the TS it passes through branch *Strobe* signals to the ROCs but replaces the up to 8-bit data pattern originating from the TS with an internally generated pattern. The data patterns (*event ID*s) are stored in a memory on the Data Tagger Module that is programmed by the user. The pattern driven from the Data Tagger Module advances by one location in memory for each *Strobe* received. The *Acknowledge* signals returning from the ROCs are passed directly through the Data Tagger Module back to the TS.

A single TS Data Tagger Module handles all 4 TS-ROC branch connections. Because the communication on the 4 branches is independent (see above), there must be a separate *event ID* memory for each branch. They are programmed simultaneously by the user. They can be independently read to confirm that the data stored in the 4 memories match as they must.

Four 32-bit scalers count the *Strobe* signals on the 4 branches. These can be read at any time. The scaler values should be identical in count at the end of a run or whenever the run is paused.

An option to pass the *Sync Flag* generated by the TS through the Data Tagger Module to the ROCs as *Data(0)* exists. For example, the TS can be programmed to alternate the *Sync Flag* on consecutive events. Together with the remaining bits driven by the Data Tagger Module we have 2 independently generated tags for the event.

Note: The TS *forced sync* function is defined by driving the *Strobe* with *Sync Flag* asserted and all other data bits <u>not</u> asserted. This is the only case where the *Strobe* is not associated with a trigger. The TS *forced sync* function cannot be used with the TS Data Tagger Module.

Procedure to Operate Module (see also Module Registers)

(1) Set VME A24 base address using rotary switches on board.

(2) Reset module: write value 0x80000000 to CSR.

(3) Enable memory write: write value 0x1 to CSR.

(4) Write MEMORY WRITE register with target memory address and data. Repeat for all 1024 memory locations.

(5) Disable memory write and enable user memory read: write value 0x2 to CSR.

(6) Verify Branch 1 - 4 memory:
   a. write value 0x80000000 to Branch 1 MEMORY READ register to reset address to 0. Repeat read of MEMORY READ register yields address and data stored in memory (address increments after each read).
   b. write value 0x80000000 to Branch 2 MEMORY READ register to reset address to 0. Repeat read of MEMORY READ register yields address and data stored in memory (address increments after each read).
   c. write value 0x80000000 to Branch 3 MEMORY READ register to reset address to 0. Repeat read of MEMORY READ register yields address and data stored in memory (address increments after each read).
   d. write value 0x80000000 to Branch 4 MEMORY READ register to reset address to 0. Repeat read of MEMORY READ register yields address and data stored in memory (address increments after each read).

(7) Disable user memory read: write value 0 to CSR.

(8) Reset memory read address to 0: write value 0x10000000 to CSR.

(9) Enable memory read via TS strobe: Make sure Trigger Supervisor is configured and ready. Then write value 0x4 to CSR.

Module Registers

The module is programmed by the user through VMEbus protocols (ANSI/IEEE STD1014-1987). The device meets all VMEbus standards. The TS Data Tagger module is categorized as an A24 – D32 VMEbus slave. All storage locations can be accessed as both Supervisory and Non-privileged data (AM = 3D, 39).

The base address (A23 – A8) is selected by four rotary switches on the board. The module occupies 256 bytes of VME address space (64 32-bit registers). Only 12 registers are currently defined. The remaining space is reserved for testing and future use.

VERSION (local address = 0x0)

[31…16] – (R) – Data Tagger identifier = 0x2000

[15…8] – (R) – board revision = **0x01**

[7…0] – (R) – firmware revision

CONTROL/STATUS (CSR) (local address = 0x4)

0 – (R/W) – enable memory write

1 – (R/W) – enable user memory read

2 – (R/W) – enable memory read via strobe

3 – (R/W) – use sync flag as data bit 0

[4…27] – reserved (read as zero)

28 – (W) – reset memory read address to 0

29 – (W) – clear memory contents

30 – (W) – module soft reset

31 – (W) – module hard reset

MEMORY WRITE – Branch 1-4 (local address = 0x8) (Write only)

[25…16] – (W) – 10-bit memory address to write

[7…0] – (W) – 8-bit data to write

MEMORY READ – Branch 1 (local address = 0xC)

31 – (W) – reset 10-bit read address to 0

[25…16] – (R) – 10-bit memory address read

[7…0] – (R) – 8-bit data read

(Consecutive reads increment 10-bit read address)

MEMORY READ – Branch 2 (local address = 0x10)

31 – (W) – reset 10-bit read address to 0

[25…16] – (R) – 10-bit memory address read

[7…0] – (R) – 8-bit data read

(Consecutive reads increment 10-bit read address)

MEMORY READ – Branch 3 (local address = 0x14)

31 – (W) – reset 10-bit read address to 0

[25…16] – (R) – 10-bit memory address read

[7…0] – (R) – 8-bit data read

(Consecutive reads increment 10-bit read address)

MEMORY READ – Branch 4 (local address = 0x18)

31 – (W) – reset 10-bit read address to 0

[25…16] – (R) – 10-bit memory address read

[7…0] – (R) – 8-bit data read

(Consecutive reads increment 10-bit read address)

SCALER 1 (local address = 0x1C)

[31…0] – (R) – strobe count TS branch 1

31 – (W) – write ‘1’ resets SCALERS 1-4

(Read of SCALER 1 also latches count of SCALERS 2-4)

SCALER 2 (local address = 0x20)

[31…0] – (R) – strobe count TS branch 2

SCALER 3 (local address = 0x24)

[31…0] – (R) – strobe count TS branch 3

SCALER 4 (local address = 0x28)

[31…0] – (R) – strobe count TS branch 4

TEST (local address = 0x2C)

[31…0] – (R/W) – data for test of data transfer integrity