The Jefferson Lab High Resolution TDC Module

F.J. Barbosa, E. Jastrzembski, J. Proffitt, J. Wilson

Thomas Jefferson National Accelerator Facility, 12000 Jefferson Ave., Newport News, VA 23606

Abstract

We discuss the design of a high resolution TDC module for use in nuclear physics experiments at Jefferson Lab. Preliminary results on the performance of the prototype module are presented.

I. INTRODUCTION

A high resolution TDC module has been designed for use in nuclear physics experiments at Jefferson Lab. The Continuous Electron Beam Accelerator Facility at Jefferson Lab is currently delivering up to 5 GeV electrons to three experimental halls. These experiments use Fastbus based high resolution TDCs that are no longer available commercially. New high rate experiments are also being planned that place additional demands on data acquisition components. Our design is targeted to meet the requirements of current experiments, as well as address the anticipated needs of future experiments at Jefferson Lab.

A recent trend in large experiments is the design of custom integrated circuits that perform the desired measurement task. Generally these are low power multi-channel chips allowing them to be mounted directly on the detector elements. Many circuits include complex filtering features to suppress data unrelated to the trigger. Some of these integrated circuits are marketed by commercial firms.

One such chip is the F1 TDC which was originally developed for the Compass collaboration at CERN and is now marketed by acam-messelectronic gmbh [1]. We have designed our TDC module using eight of these integrated circuits.

Our design is implemented as a VME64x slave module. This bus standard was chosen because it is already in use at Jefferson Lab, has good (and evolving) data transfer capabilities, and reasonable channel densities are possible.

II. THE TDC MODULE

A. F1 TDC Chip

The core of our TDC module is the F1 TDC chip. This chip uses purely digital delay techniques to measure time. Each of the pipelined F1 chips provides eight channels at 120 pS LSB or four channels at 60 pS LSB when a reference clock of 40 MHz is used. Internal FIFOs allow for storage of 16 hits per channel in leading and/or trailing edge modes. The following outline pertains to the use of the F1 chip in our TDC module.

A key feature of the F1 chip is a Trigger Matching processing unit, which allows for selection of hits within a programmable time window and latency from the occurrence

of a valid trigger input. Hits that fall outside of the window and latency settings are suppressed from the output buffer and cleared from the hit FIFO. The trigger matching feature is used in common start/stop and synchronous measurement modes. In common start/stop mode, a start signal resets the internal measurement counter and a trigger signal sets the measurement window. Hits falling within these two signals will always be accepted by the trigger matching unit. In synchronous mode, a syncres signal is used to reset the internal measurement counter and thus synchronize all TDCs in an experiment. Internal start signals are automatically generated at a programmable rate. The trigger matching unit validates hits within the programmed window and latency. Headers and trailers identifying the channel, chip, trigger time and event number can be output to delineate events. The dynamic range is 7.8 μ S at 120 pS LSB and 3.9 μ S at 60 pS, with a 40 MHz reference clock.

To ensure stability and measurement reliability, each F1 chip is PLL-regulated against temperature drifts and manufacturing tolerances. The feedback loop employs a phase-frequency detector, a loop filter and a voltage regulator, which drives the substrate or core of the F1 chip. In this manner, delays within the internal Delay Locked Loop (DLL) are kept constant which translates into a stable LSB resolution or bin size.

The F1 chip is configured via a serial interface port which accesses 16 registers. Readout of data is via a 24-bit parallel port.

B. Module Architecture

Figure 1 shows a block diagram of the TDC module. The eight F1 TDC chips on our module provide 64 channels in normal mode, or 32 channels in high resolution mode. Front panel input signal levels are differential ECL to be compatible with existing systems at Jefferson Lab. Timing control signals are also available through backplane connections for ease of system integration. A 128K word deep FIFO capable of simultaneous reads and writes is attached to each F1 TDC chip to buffer its output data. The VME interface and all control logic and registers are implemented in a single Field Programmable Gate Array (FPGA). All VME bus transceivers are compatible with the high-speed protocols of the VME64x standard [2]. The RAM shown in Figure 1 is not present on this prototype. It is discussed in the section on our future plans.

Output data from the F1 TDC chip is ordered according to channel number (channel 0 first, channel 7 last). Within each channel, hit data is ordered from earliest to latest. We configure each F1 TDC chip to supply a header word for channel 0 and a trailer word for channel 7 in its output data

Figure 1: TDC module block diagram.

stream whenever a trigger occurs. This takes place even if there is no valid hit data for the trigger. During readout, control logic uses these markers to switch between the FIFOs that supply the data. In this way, data from multiple chips that are associated with the same trigger may be assembled into an event fragment for the module. Individual channels may be disabled during configuration of the F1 TDC chips. The FIFO for an individual F1 TDC chip may also be bypassed from the readout sequence.

The module can be configured to set a flag in a register when the data from a programmed number of triggers is available in its buffers. This flag may be optionally used to interrupt the crate controller. Readout is accomplished most efficiently using a block transfer protocol. Because the number of data words currently stored in the module is not available to the crate controller, a slave terminated block transfer is used instead. The crate controller is programmed to read out, via block mode, a number of words that is beyond the storage capacity of the module. During readout the module will provide the block of data associated with the programmed number of triggers. When this data is depleted the module terminates the block transfer by issuing a bus error (BERR). A register bit in the module is set when this occurs. The crate controller can query this bit to verify that the bus error was generated deliberately by the module and does not signify a system failure.

To enhance system performance a set of TDC modules may be read out as a single logical read using a multiblock protocol. This involves passing a token between modules along a VME bus daisy-chain line (BGN). It is implemented

in the following way. All TDC modules in the set are programmed to respond to a Common Address Range used exclusively for multiblock transfers. Only the TDC having the token will actually respond to such an address cycle. The TDC module of the set that is furthest left in the crate is configured to be the *first* module and initially claims the token. The module furthest to the right is set as the *last* module. All modules except the *last* one are configured to pass the token when they have transferred the programmed number of events. The *last* module is instead configured to respond with BERR. In addition, the *first* module may be configured to initiate readout by interrupting the crate controller when the data from a programmed number of triggers is available in its buffers. A slave terminated block transfer using the Common Address Range will accomplish the readout of the data from the programmed number of triggers for all boards of the set. When BERR occurs signaling the completion of this transfer, the *first* module seizes the token again.

C. Configuring the F1 TDC

The TDC module contains eight F1 TDC devices that must be configured prior to operation. Each device contains sixteen 16-bit write-only configuration registers that are accessed through a synchronous serial bus. The TDC configuration control logic is implemented in the FPGA. It contains a serializer for the TDCs, a register file RAM, a configuration sequence RAM, an EEPROM controller, and host interface logic. An overview of the configuration control logic architecture is shown in Figure 2.

Figure 2: F1 TDC Configuration Control Logic

A 24-bit frame is transmitted to 8 TDCs on a synchronous serial bus. This frame consists of a 3-bit device address, one broadcast bit, a 4-bit register address, and 16-bit data. A TDC will accept a frame if its unique hardware address matches the device address transmitted in the frame.

The TDC configuration registers are write-only. Read-back is accomplished using a local copy of the registers stored in FPGA RAM. A 128x16 register file RAM stores all 16 registers from each of 8 TDCs. Any write to the register file will store the data in RAM and serialize the data to the TDCs. The serializer will construct the TDC data frame based on the register file address and write data.

For the TDCs to function properly, the TDC registers must be configured in a specific order. A configuration sequence RAM is provided to remove this burden from the host software. The 128x24 sequence RAM permits configuration of any register on any TDC in any order. This is accomplished by storing entire 24-bit serial data frames sequentially in RAM. A programmable end address is used to mark the end of sequence. When the configuration sequence is triggered, the RAM is read sequentially from address 0 to the end address. The 24-bit data is decoded and passed to the register file where it is stored and serialized.

A 512x8 serial EEPROM stores the sequence RAM and other user data. The EEPROM is accessed through an EEPROM controller and an SPI compatible interface. The controller provides a random access read/write interface to the EEPROM. It manages device opcodes, status, and data. The SPI interface contains a clock manager and a serializer/deserializer. Since the Sequence RAM and end address occupy only 385 bytes of the 512-byte EEPROM, the remaining space is available for user data.

The Master Controller module provides a host interface to the control logic and interconnects the register file, sequencer,

and EEPROM modules. It permits read and write access to the Register File RAM, Sequence RAM, and EEPROM. The host can trigger TDC configuration from Sequence RAM and transfer of Sequence RAM to or from the EEPROM. At power-up, the Sequence RAM and end address are automatically loaded from EEPROM then passed to the register file and serializer to configure the TDCs. This automatic power-up configuration will remove considerable software overhead for a large number of TDC boards.

D. Implementation

The TDC module is fabricated as a single 12-layer printed circuit board. Signal characteristics are preserved throughout the front end of the circuit board by extensive use of differential PECL strip lines. The strip lines are properly sized for 50 Ohm termination and to provide proper signal deskewing. Except for connectors, components are surface mounted using both sides of the board. The module uses +3.3V and thus requires a VME64x backplane. Total power dissipation is 38 W.

III. MEASUREMENTS

A. Test Setup

We use the Highland V851 Digital Delay Generator [3] as our signal source. This VME module drives up to six outputs with programmable delays relative to a reference output T0. All outputs are NIM levels. Four of these outputs are fanned out 8 ways to create 32 timing signals. These are converted to ECL levels and serve as inputs to 32 of the TDC module channels. The T0 output is converted to ECL and is used as the *Start* input to the TDC. Another output is converted and is used as the *Trigger* input to the TDC. This setup allows us to study half of the TDC module channels simultaneously.

The delay of *Trigger* relative to *Start* is fixed to be slightly less than the full dynamic range of the TDC (3.9 μ s or 7.8 μ s). Delays of the 4 source signals (from which the 32 input timing signals are derived) are adjusted to study the TDC response. Trigger matching mode is enabled for the F1 TDC chips, with latency and window parameters programmed to be slightly less than the *Trigger* delay.

With this setup the module functions as a multi-hit common start TDC. A hit on an input channel that occurs between the *Start* and the *Trigger* times will be measured.

The quality of the TDC test signals was checked with the Stanford Research System SR620 Time Interval Counter [4]. The SR620 has a resolution of 25 ps. The jitter of the TDC test signals relative to *Start* was measured to be 33 ps (RMS). The SR620 also confirmed the linearity of the V851 over the range in which it was used.

B. Results

Figure 3 shows the timing distribution for an input signal that has a fixed time relationship to the *Start* signal. The TDC was operated in high resolution mode. Unfolding the uncertainty of the input signal (33 ps) from the measured distribution yields a TDC resolution (RMS) of 61.2 ps. Figure 4 shows the equivalent distribution for a TDC operated in normal resolution mode. The resolution is 86.2 ps.

Figure 3: Resolution = 61.2 ps.

Figure 5 shows the transfer function for the TDC in high resolution mode. The linear fit gives a slope of 58.8 ps/channel. Residuals for the fit are shown in Figure 6. For this figure, note that one TDC channel corresponds to about 0.06 ns. Figure 7 displays the transfer function for the TDC in normal resolution mode. Figure 8 displays the TDC resolution across the dynamic range of the TDC operated in high resolution mode. Figure 9 displays the resolution of the TDC as a function of trigger rate.

Figure 4: Normal Mode. Resolution = 86.2 ps.

Figure 5: High Resolution Mode Transfer Function

IV. FUTURE PLANS

Testing the performance of the F1 TDC chip in this environment will continue. Additional rate and noise studies will be performed. Multiple board systems will be assembled.

The VME interface of the prototype will be upgraded (by firmware) to include the 64-bit and dual-edge (2eVME) data transfer protocols. The current interface supports only 32-bit block transfers.

The next version of the TDC module will include the RAM buffer shown in Figure 1. It will be organized as 1M x 64 bits to take advantage of the 64-bit data transfer modes of VME.

Figure 6: Residuals for the linear fit.

Figure 8: Resolution across the TDC dynamic range.

Figure 7: Normal Mode Transfer Function

Figure 9: Resolution as a function of trigger rate.

V. ACKNOWLEDGEMENTS

This work was supported by DOE Contract #DE-AC05-84ER40150.

VI. REFERENCES

- [1] acam-messelectronic gmbh - Am Hasenbiel 27 - D-76297 Stutensee-Blankenloch, Germany.
- [2] ANSI/VITA 1.1-1997.
- [3] Highland Technology, 320 Judah Street, San Francisco, CA 94122.
- [4] Stanford Research Systems, 1290-D Reamwood Avenue, Sunnyvale, CA 94089.

Figure 1: TDC module block diagram.

stream whenever a trigger occurs. This takes place even if there is no valid hit data for the trigger. During readout, control logic uses these markers to switch between the FIFOs that supply the data. In this way, data from multiple chips that are associated with the same trigger may be assembled into an event fragment for the module. Individual channels may be disabled during configuration of the F1 TDC chips. The FIFO for an individual F1 TDC chip may also be bypassed from the readout sequence.

The module can be configured to set a flag in a register when the data from a programmed number of triggers is available in its buffers. This flag may be optionally used to interrupt the crate controller. Readout is accomplished most efficiently using a block transfer protocol. Because the number of data words currently stored in the module is not available to the crate controller, a slave terminated block transfer is used instead. The crate controller is programmed to read out, via block mode, a number of words that is beyond the storage capacity of the module. During readout the module will provide the block of data associated with the programmed number of triggers. When this data is depleted the module terminates the block transfer by issuing a bus error (BERR). A register bit in the module is set when this occurs. The crate controller can query this bit to verify that the bus error was generated deliberately by the module and does not signify a system failure.

To enhance system performance a set of TDC modules may be read out as a single logical read using a multiblock protocol. This involves passing a token between modules along a VME bus daisy-chain line (B_{Gn}). It is implemented

in the following way. All TDC modules in the set are programmed to respond to a Common Address Range used exclusively for multiblock transfers. Only the TDC having the token will actually respond to such an address cycle. The TDC module of the set that is furthest left in the crate is configured to be the *first* module and initially claims the token. The module furthest to the right is set as the *last* module. All modules except the *last* one are configured to pass the token when they have transferred the programmed number of events. The *last* module is instead configured to respond with BERR. In addition, the *first* module may be configured to initiate readout by interrupting the crate controller when the data from a programmed number of triggers is available in its buffers. A slave terminated block transfer using the Common Address Range will accomplish the readout of the data from the programmed number of triggers for all boards of the set. When BERR occurs signaling the completion of this transfer, the *first* module seizes the token again.

C. Configuring the F1 TDC

The TDC module contains eight F1 TDC devices that must be configured prior to operation. Each device contains sixteen 16-bit write-only configuration registers that are accessed through a synchronous serial bus. The TDC configuration control logic is implemented in the FPGA. It contains a serializer for the TDCs, a register file RAM, a configuration sequence RAM, an EEPROM controller, and host interface logic. An overview of the configuration control logic architecture is shown in Figure 2.