

Application Note: AN101

Scaling and Synchronizing Multiple StreamStors for Higher Combined Bandwidth

Rev. A, November 20, 2003

Overview and Background

A single StreamStor PCI-816XF2¹ controller provides sustained minimum recording and playback rates up to 200MBytes/s (1.6Gbits/s). While this rate is adequate for most applications, it may not be sufficient for others. This application note demonstrates a method of operating multiple StreamStor controllers in parallel so as to produce a recording and playback system with the combined bandwidth of all the StreamStor controllers in the system.

FPDP Interface Overview

FPDP² (Front Panel Data Port) is a 32-bit parallel 5V TTL bus with flow control. In general, there is one TM (Transmit Master) and one RM (Receiver Master) on a FPDP bus. These devices are located at the physical ends of the FPDP bus and are responsible for terminating the signals. By its very nature, the interface is simple to implement. The specification defines the interface electronics. The user need only supply the glue logic that is specific to the application.

FPDP data transfer is synchronous to the FPDP clock which is generally supplied by the TM. The TM must change data so as to meet the specified data setup and hold times with respect to the FPDP clock that it generates.

FPDP has a data flow control system that allows both the TM and RM to throttle the flow of data. The TM uses the DVALID* signal to indicate when it is sending data that the RM should capture. The RM controls a SUSPEND* signal which is used to tell the TM that it can not accept more than 16 data words before overflowing its data FIFO. When SUSPEND* is detected by the TM, the TM is obligated to stop sending data anywhere within 16 additional data transfers until SUSPEND* is withdrawn. In general, the FPDP clock continues to run though throttling by either device, TM or RM, is occurring.

FPDP is a non-arbitrated bus. Thus, every clock cycle can be used to transfer data so long as all the participants on the bus are, depending on their roles, able to send or receive data. For these reasons, a system that is architected with enough TM and RM bandwidth can produce very precise and predictable behavior.

¹ While this discussion describes an implementation based on StreamStor PCI-816XF2, the same or similar scaling techniques may apply to other interfaces on other StreamStor products. Please contact Conduant Corporation for applicability to other products.

² FPDP refers to ANSI/VITA Standard 17-1998. Additional information can be found at www.fpdp.com. In addition, Conduant application note AN-102 fully describes the StreamStor implementation of FPDP which includes several capabilities that are beyond the requirements of the official specification.

Scaling to Record at High Speeds

Referring to Figure 1, scaling multiple StreamStors to record a higher speed data source is accomplished in five steps:

- 1) Split the high-speed data stream into multiple slower-speed split data streams,
- 2) Deposit each slower-speed stream into its dedicated dual-clock FIFO,
- 3) Implement a FPDPTM interface for each split stream,
- 4) Implement FC (Flow Control) logic, and
- 5) Sequence commands to hardware to move data from the TM to RM.

Recording Step 1: Data Splitting

Figure 1 shows a basic architecture for splitting a high-speed source into “n” number of lower-speed data streams. Data is clocked into the splitter at data rate “r” with the system clock and it emerges from the splitter as “n” streams, each with an average³ data rate of “r/n”. The customer’s application defines the source interface, the data rate, and how many split streams must be created. Though not required, it is assumed that the splitter will follow a round-robin 32-bit splitting algorithm.

Recording Step 2: Input Side of Write FIFO

Each split stream is deposited into its associated FIFO with a clock that is synchronous to the system clock⁴ at an average data rate of “r/n”. This completes the recording data path in the high-speed clock domain. It is assumed that the source data stream is based on a real-time event and must always be accepted for storage to disk. For this reason, there is no flow control on the “write” side of the FIFO. The customer could, however, detect if a deposit is made by the splitter into a FIFO while the FIFO is full. If this were to occur, it would be an error condition that should be investigated.

³ Average rate takes duty cycle of WRREQ into account. The burst rate that data is deposited into the FIFO is based on the clock speed if WRREQ was always on. The average rate is the proportional value of the burst rate for which WRREQ is active.

⁴ Though shown as the same clock in figure 1, the actual clocks driving the FIFOs may be lower-speed, phase-locked clock derivatives of the original high-speed clock. This is likely because of the speeds that would force a user to use multiple StreamStors in the first place.

Recording Step 3: FPDP Implementation

A FPDP TM interface is easily implemented. The official standard is relatively short. While it includes some options, it is not necessary to implement them in most cases. The standard specifically identifies the off-the-shelf ICs and resistors that are required to implement a physical TM interface. These components simply drive and receive signals and are not shown in Figure 1.

The user implementation will provide a FPDP clock⁵ that is not synchronized to the system clock. It is likely that the output side of each FIFO will use its respective FPDP clock since data reads out of the FIFO are synchronized to transmitting data over FPDP. The FPDP clock frequency should be higher⁶ (at least 1 percent) than the input side clock. This will eliminate the possibility of accumulation of data that could lead to a FIFO overflow. Thus, the output side of each FIFO is part of and is synchronous to each associated FPDP machine.

Recording Step 4: FC (Flow Control) Logic

Flow control logic is what keeps the FIFO from overflowing and underflowing. The algorithm for this FC logic is very simple. When the FIFO is not empty (as indicated by EF), the logic should read data out of the FIFO, place it on the FPDP bus, and assert DVALID* signal for the clock edge for which the data is valid. When the FIFO is empty, no more data should be taken from the FIFO and DVALID* should be deasserted. If SUSPEND* is asserted by the StreamStor connected to that FIFO, then the logic should also stop the flow of data from the FIFO within 16 data reads in the same manner it would if the FIFO was empty. Data flow would then continue if both SUSPEND* and EF are inactive. Depending on the user implementation, AEF may also be useful in scheduling the movement of data out of the FIFO.

Recording Step 5: Sequence Commands to Hardware

With a parallel hardware solution in place, the procedure to start recording is as follows:

- 1) Ensure that splitter is halted and can not write data to recording FIFOs.
- 2) Ensure that the recording FIFOs, are all flushed (empty).
- 3) Ensure that all StreamStor recorders are stopped. Failure to do so could cause one or more to prematurely record erroneous data and thus would create data misalignment among them.
- 4) Enable each StreamStor as a RM.
- 5) Enable each port on the splitter as a TM. Because the FIFOs are flushed, DVALID* should automatically be driven to an inactive state.
- 6) Set the FPDP clock speed driven by each TM port to be slightly higher (+1-3%, for instance) than the sustainable data rate of each individual data stream.
- 7) Enable each StreamStor to start recording.
- 8) Enable the splitter to start spitting data, starting with the first fragment going to the first StreamStor, the second to the second, and so on.

To stop the recording concisely:

- 1) Stop the splitter on a boundary that is an integral multiple of the number of StreamStor recorders. This step should not prevent the FIFOs from dumping their data out to their respective FPDP busses.
- 2) Allow adequate time for each of the FIFOs to dump their data to their respective FPDP busses. The required time delay is dependent on the FIFO sizes and the FPDP clock speed.

⁵ Though shown as a single FPDP clock wired to each FPDP port, a better design is to redrive the clock from the oscillator with a clock driver chip, thus making each clock connection a point-to-point connection from the driver chip to the FPDP circuit for each port. This will produce much “cleaner” clocks and will minimize problems.

⁶ While StreamStor documents specify a 50MHz limitation on the FPDP clock, it is recommended that the FPDP clock be driven slightly faster than 50MHz when data is sourced out of dual clock FIFOs when flow control is present.

- 3) From the time each FIFO empties, ensure that the FPDP clock associated with that TM is allowed to run for at least 1100 FPDP clock cycles plus 20uS. This ensures that data will exit the StreamStor FIFOs that are dependent on the FPDP clock.
- 4) The STOP command may then be sent to that StreamStor. This command will return control to the application only after all customer data has moved from FIFOs and RAM to disk.
- 5) Once this procedure is completed for all StreamStors, each StreamStor will have exactly the same number of words recorded, or "1/n" of the total data.

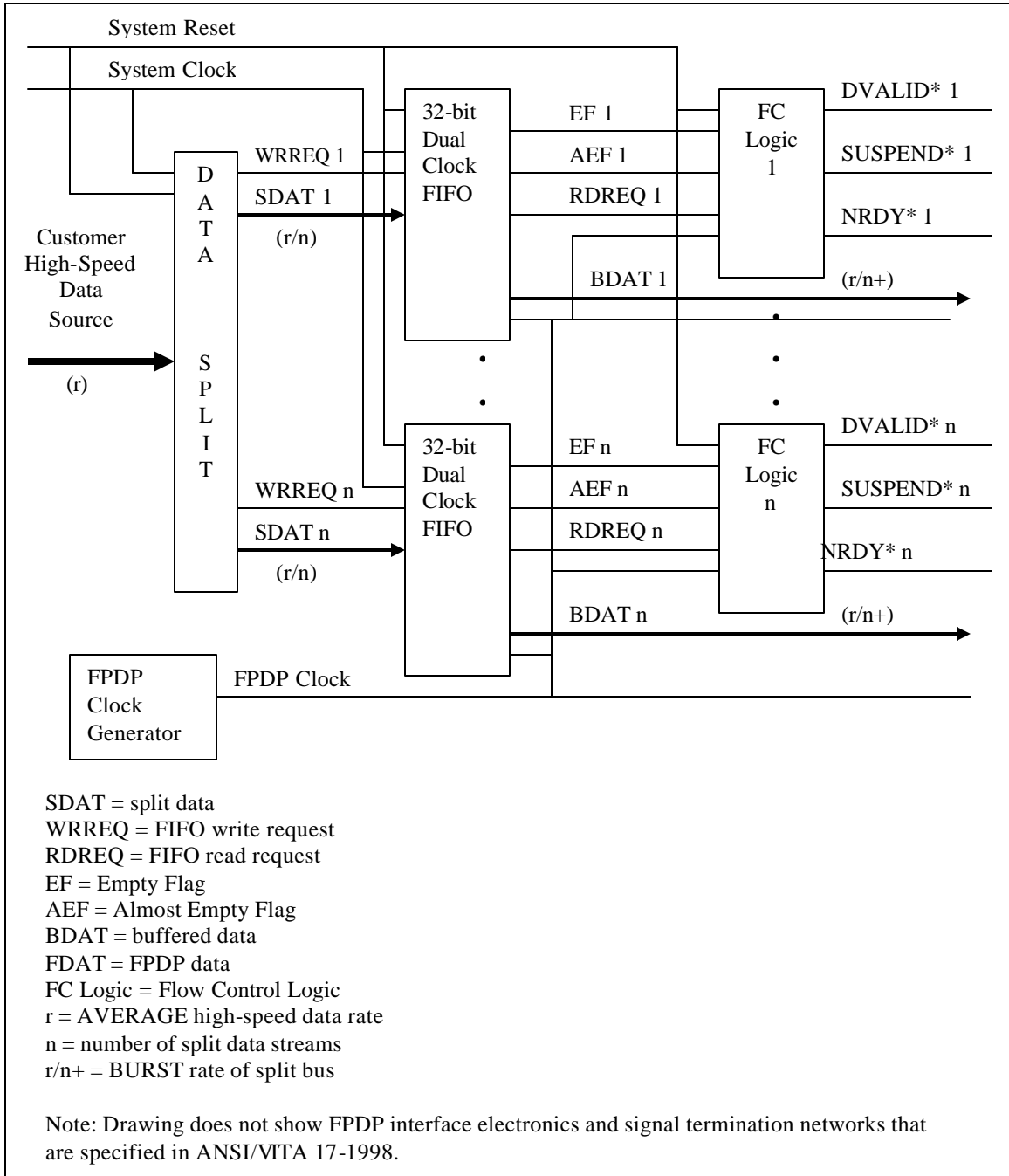


Figure 1 FPDP Parallel Write Architecture

Scaling to Play Back at High Speeds

Referring to Figure 2, scaling multiple StreamStors to play back a higher speed data stream from multiple lower speed sources is accomplished in five steps:

- 1) Implement a FPDP RM interface for each split stream,
- 2) Insert a FIFO for each split stream,
- 3) Add FC (Flow Control) logic for each RM, and
- 4) Combine the split streams into a single high-speed stream.
- 5) Sequence commands to hardware.

Playback Step 1: FPDP RM Interface for Each StreamStor

Using the FPDP specification for guidance, implement a FPDP RM for each StreamStor. As shown, each StreamStor will drive DVALID* and the FPDP clock, while each RM will drive SUSPEND* and NRDY*. Each RM will operate synchronous to the FPDP clock supplied by StreamStor that it services. Thus, data will be deposited into a FIFO synchronous to the FPDP clock of the attached StreamStor. There is no need to synchronize the StreamStor clocks together at this point (refer to Playback Step 3). This simplifies the implementation. Much of the interface electronics (not shown in the figures) from the TM implementation can be reused since several of the ICs have bidirectional capability. Note, however, that some of the signals require different termination depending on the direction that data is being transferred. Switching in the proper termination can be performed can be accomplished with crossbar (FET) switches.

Playback Step 2: Input Side of FIFO

Each split stream is deposited into its associated FIFO synchronous to the FPDP clock for that particular StreamStor. The StreamStor FPDP clock driving this RM should be at a frequency **higher than** “r/n”. This, in conjunction with the flow control logic described in Playback Step 3, makes it possible for each StreamStor to keep each FIFO “topped off” with data and from becoming data starved on its output side. The FIFO must be larger than 16 words since the FPDP rules for SUSPEND* require a 16 word advance warning. The number of required additional words required depends on how the flow control logic generates SUSPEND*. An easy way to generate SUSPEND* is to drive it using the HF (Half Full) flag. Most off-the-shelf FIFO chips have one of these. Some FIFOs also support a PAF (Programmable Almost Full) bit. If the HF flag is used, then the FIFO must be at least 32 words to generate the 16 word advance warning. Off-the-shelf FIFOs that are much larger than 32 words are inexpensive and readily available. Larger FIFOs add more protective buffer space and are recommended if there is no compelling reason to use a tiny one.

Playback Step 3: Flow Control Logic

The flow control for each RM will control the SUSPEND* signal such that it will cause StreamStor to store data in the FIFO without overflowing it. Control of the SUSPEND* signal can be almost as simple as driving it from one of the FIFO flags (half full or almost full). SUSPEND* must be asserted such that the receipt of 16 additional FPDP words will not overflow the FIFO. When the FIFO is not near full, SUSPEND* should be deasserted. StreamStor will begin to deposit data in about 5 FPDP clocks from when SUSPEND* is deasserted. This flow control, in conjunction with a FPDP clock that is slightly higher than the clock rate used to read data out of the FIFO for resynchronization, ensures that data will be combined properly in playback step 4.

Playback Step 4: Combining the Data

The output sides of all FIFOs are synchronized to the system clock, allowing the original stream to be reproduced by symmetrically interleaving (combining) the data from the split streams into the original source stream. This technique works because the higher FPDP clock speed and the flow control logic for playback ensure that the playback FIFOs always have data for the combiner. It would be an error condition, except at the end of the playback, if any of the playback FIFOs do become data starved. The detection of such an error would suggest the need to determine the cause.

Playback Step 5: Sequence Commands to Hardware

To start the parallel playback process:

- 1) Ensure that combiner is halted and does not request data from playback FIFOs.
- 2) Assert and hold "FIFO clear" to all playback FIFOs to ensure that the FIFOs remain empty during the rest of system configuration.
- 3) Ensure sure that all StreamStors are stopped.
- 4) Configure each FPDP port as a RM.
- 5) Force each RM port to assert the SUSPEND*. This control would be a signal into the flow control logic for each port. This prevents any StreamStor from prematurely sending any data and creating a data out-of-sync problem.
- 6) Enable each StreamStor as a TM.
- 7) Set the FPDP clock driven by each StreamStor to be slightly higher (+1-3%, for instance) than the sustainable data rate that each split stream will be removed from a playback FIFO. This ensures that no playback FIFO becomes empty except possibly at the end of the recording.
- 8) Enable each StreamStor to start playback. Data will not yet start to move across the FPDP bus because the SUSPEND* signals are asserted to all FPDP busses. However, each StreamStor will start reading from disk and will fill up its data buffer with playback data.
- 9) Remove the "FIFO clear" signal from all playback FIFOs in the combiner, thus allowing them to hold playback data.
- 10) Remove the SUSPEND* signal from all FPDP busses. It is not necessary to remove them simultaneously, however. Each playback FIFO will immediately fill with data sent from each StreamStor. The flow on each bus will again be automatically throttled, as needed, by the flow control circuits.
- 11) Reset the combiner so that it starts taking data from the playback FIFOs in the order that data was originally sent to the StreamStors.
- 12) Release the combiner to start combining. The data path will be primed with sufficient data and each StreamStor and the associated flow control will work such that no data path to the combiner becomes starved for data.

To stop the playback concisely:

- 1) Stop the combiner. This will automatically stop the flow of data from the StreamStors due to flow control. When stopped in this manner, the playback can restart at exactly the point where it stopped without losing data synchronization.

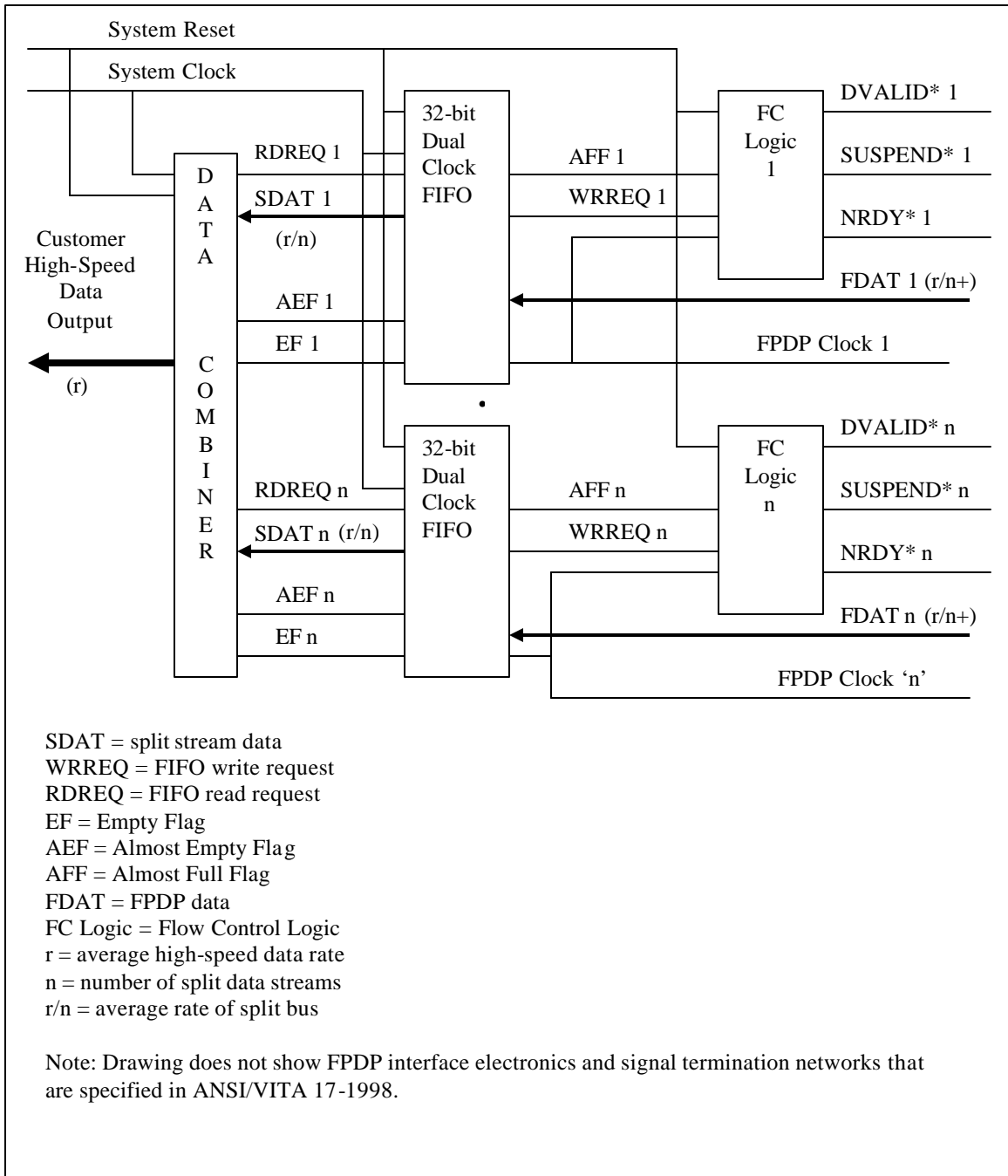


Figure 2 FPDP Parallel Playback Architecture

Summary

This application note shows a method of producing a recording/playback system with the combined bandwidth of the individual StreamStor controllers. It is intended to be used as a general overview of things to be considered and should not be considered a complete design. Each user may have different requirements regarding total bandwidth, the specifics of the high-speed interface, and the preferred electronic implementation. Please contact Conduant Corporation, as needed, for additional information.

Additional References

1. www.fpdp.com
2. ANSI/VITA 17-1998.
3. Conduant StreamStor Application Note AN-102, "StreamStor PCI-816XF/XF2 FPDP Implementation Details".
4. Conduant StreamStor Application Note AN-100, "Connecting to StreamStor CPCI-816 With a Physical Channel Link Connection and a FPDP Protocol".
5. Channel Link Data Sheets and Application Notes at www.national.com

Revision History

- A. Initial Release.

Rights and Trademarks

StreamStor™ is a trademark of Conduant Corporation.

Channel Link™ is a trademark of National Semiconductor Corporation.

CompactPCI® is a registered trademark of the PCI Industrial Computers Manufacturers Group.