Multiple Clock Domain Bridging

James Blake
University Of Utah
blake@eng.utah.edu

ABSTRACT
Multiple clock domains are often present in designs that interface with external IO sources. Specialized video and audio processing codecs generally have specific clock requirements and must interface with processing units which often have other asynchronous bus clock requirements, requiring some sort clock domain bridging. As more IO sources are integrated into single chips, it is difficult to efficiently synchronize all sources from a single clock, thus making efficient bridging of the domains desirable.

Categories and Subject Descriptors
B.4.3 [**Interconnections**]: Aynsynchronous/Synchronous operation, interfaces, physical structures.

General Terms
Measurement, Documentation, Performance, Design, Reliability.

Keywords
Clock Domain Bridging, Audio, Video.

1 INTRODUCTION
The impetus of this project is to produce a video and audio processing codec which will interface to a standard microprocessor. The microprocessor bus operates at a very different bus speed than the video codec which is driven by a standard 27 MHz clock. Audio IO is very slow (44.1 kHz) and using a high speed clock to do the audio processing would create an unnecessary waste of power. Since this should work for any embedded domain (including those built with FPGA devices), a large amount of customized logic circuit is not desireable.

In the past, the clock bridging problem has been approached in two different ways. A synchronous handshaking protocol can be used to regulate data flow, but such protocols generally waste many clock cycles, requiring clock rates significantly higher than needed for the functional block. The other method relies on asynchronous FIFO logic to bridge the clock domains. Data propagates down the FIFO and only needs to be synchronized at the point at which it interfaces directly to the clock domain.

In order to keep things simple and focus on the primary concern of bridging the clocks, the audio processing is limited to a simple low pass filter and the video processing only strips unnecessary blanking lines and sync signals. The audio filter implementation uses very basic fixed point arithmetic.

The golden model result for video processing is shown in Figure 1. Note that the vertical sync lines are still in this image, but will not be in the processed result. This is a test pattern generated from two BT656 video fields and has been stripped, with exception of the vertical sync lines, according to the process followed by the design implementation. The audio golden model is a uniform random data set run through the same filter and processed through a C implementation. This was chosen as it is easy to verify filter function by plotting the output against a FFT of the resulting data. When operating correctly, the FFT will show the same behavior as the filter response. Both models were designed to be easily validated via an automatic difference utility.

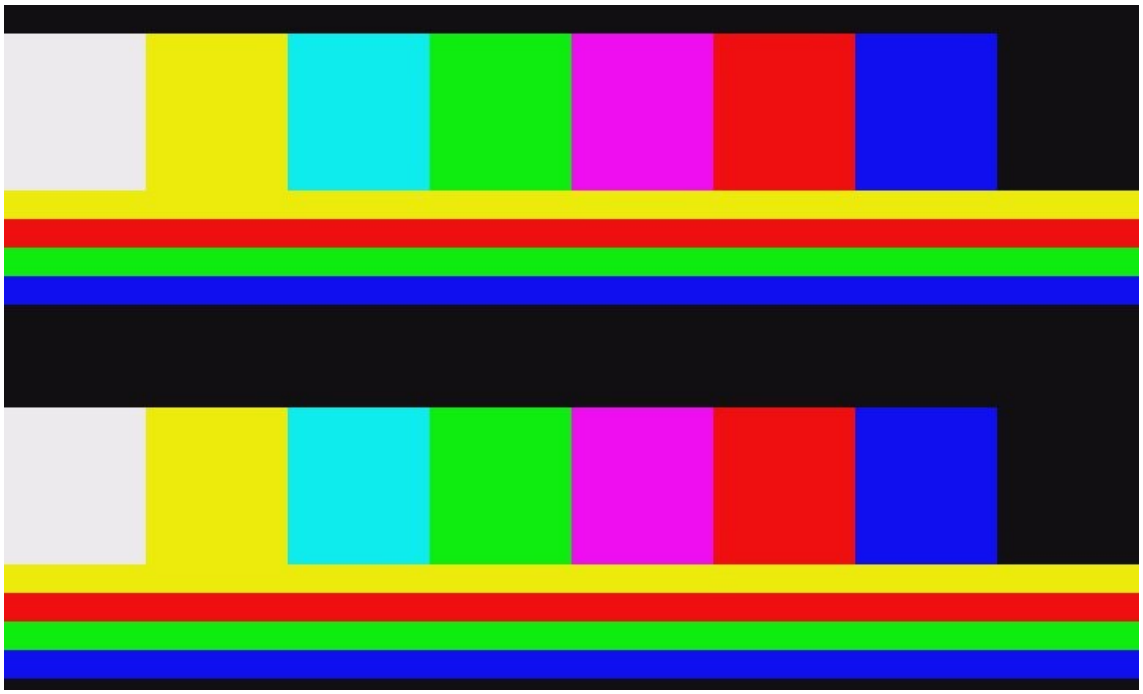

Figure 1: Processed Video Test Image

2 SYSTEM DESIGN

2.1 Overall Layout

This section details the top level design of the device and details the IO interface. There are three major sections in the design: the audio processing, the video processing, and the processor IO bridge that ties the clock domains to an external processor interface.

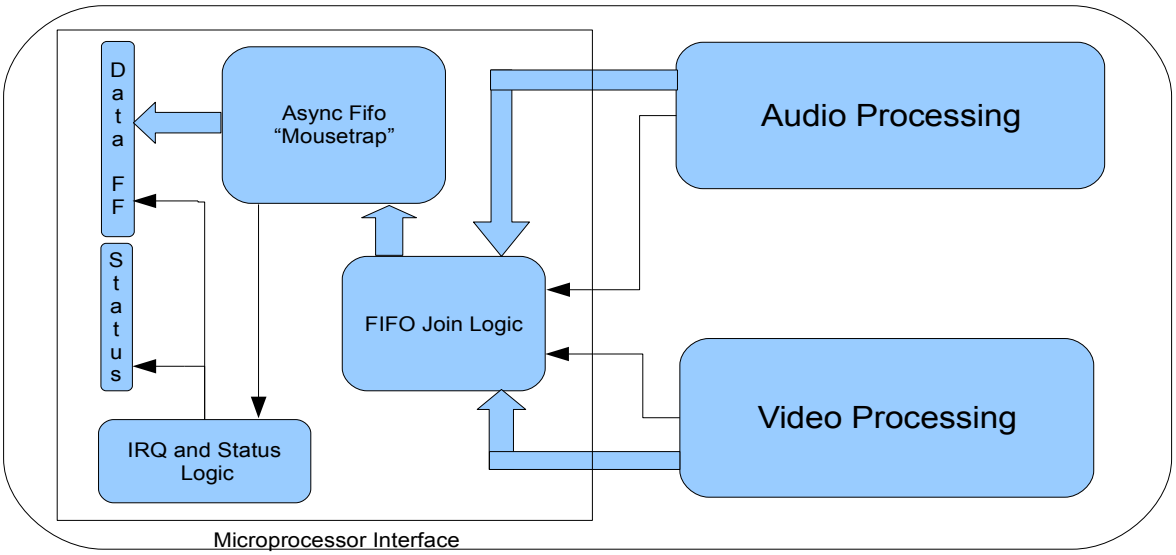

Figure 2: Top Level Layout and Block Diagram

2.1.1 Area Estimates

The following table gives the actual layout area from the post-route reports for the modules of the design.

Table 1: Area Estimates

Design Block	Area (square micrometers)
Video Sequencer	182800
Audio Processor	1433841
Processor Logic and FIFO	756390

2.1.2 Power

This design has two main power advantages. First, high resolution clocks are not being distributed to sections that run at much lower frequencies. This is a huge advantage for the Audio processing which has a relatively slow clock. Also, obtaining the video and bus clocks from a single clock would likely have required a much higher rate internal clock.

The second advantage is the lack of active clocking in the output FIFO. Since data only moves when it is available and movement is possible, the gates are not required to switch with the clock. The FIFO was grouped with the processor logic as the final modules were synthesized in that manner.

Table 2: Estimated Power For Functional Areas

Design Block	Power Estimate (milliwatts)
Video Sequencer	18.7
Audio Processor	36.1
Processor Logic and FIFO	76.4

2.1.3 Performance

The primary performance enhancement of this design is that the data may be moved into the FIFO on every video clock or every audio clock. Likewise, data can be clocked out every processor clock assuming that the FIFO logic propagates forward at least one word slot faster than the clock cycles. This is possible because the two phase signaling of the selected FIFO only requires one change per audio, video, or microprocessor clock. This also simplifies the design because a D Flip-Flop can be used for synchronizing logic.

In a fully clocked handshaking system, several clocks would be wasted for each word moved. To compensate for this, the words could be made larger, thus involving more wiring complexity and likely making the design much slower.

2.1.4 Process Technology

This design was originally built using a 130 nm process technology. However, as the design progressed, this seemed to be overkill. Since cost is a major driving factor in embedded designs, the older, cheaper technology seemed to be a better fit. Thus the final cells selected were based on an older 0.5u technology. One possible change that could drive using a newer process would be inclusion of the embedded processor into the chip design. High performance encoding operations could justify newer, more expensive process technologies.

2.1.5 External Pins

Figure 3 shows the external pins of the design. Pins are placed on the segments with which they associate. The Reset pin is common to all segments.

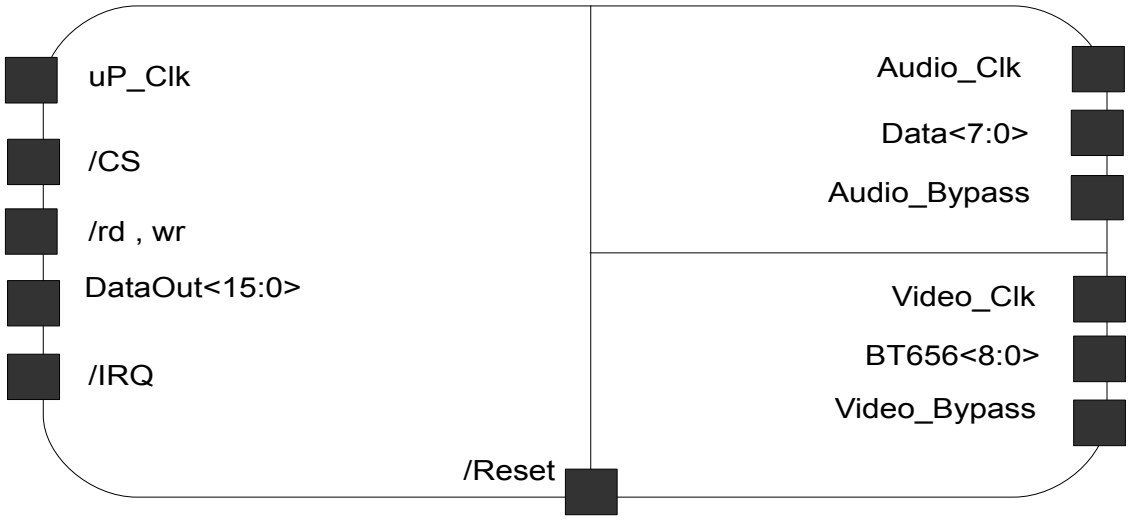

Figure 3: External Pin Interface

Table 3 gives a description of each pin's function within the functional system.

Table 3: Pin Names and Descriptions

Pin	**Description**
up_clk	This is the microprocessor input bus clock pin. Nominal rate for this external bus is 66 MHz minimum.
CS	This is a low asserted chip select pin. Read and write requests are ignored if chip select is asserted high.
/rd, wr	If chip select is low asserted, then this pin indicates a read or write request depending on its asserted level. A high assertion indicates a write request. This does nothing more than clear the interrupt until a read request is performed.
DataOut<15:0>	The lower byte of the DataOut is the actual data being read. The upper byte is used to indicate status. Bit 9 indicates the data type. Bit 10 indicates whether the data is valid (i.e. empty queue). Bit 11 will indicate FIFO overflow. The remaining bits are currently unassigned and may be used for testing.
/IRQ	When pulsed low, this indicates an active interrupt request to the processor indicating that sufficient data is buffered to do a burst read.

Pin	Description
/Reset	When pulled low, this pin resets the chip state machines and initializes the FIFO states to not busy.
Audio_Clk	This is a standard CD quality audio clock at a nominal 44.1 kHz clock. rate.
Data<7:0>	Audio data samples are input through this bus.
Audio_Bypass	Bypass the audio state processing when asserted
Video_Clk	This is the video clock rate at the standard 27 MHz for BT656 video.
BT656<7:0>	This is the BT656 video data input.
Video_Bypass	Bypass the video state processing when asserted

2.2 Audio Processor

In a multimedia device the audio processor would perform encoding or decoding applications. This audio processor does a simple 8-tap low pass FIR filter. This was done mostly to keep the implementation function small and decrease the area already required by the audio module. This is likely the largest block in the entire design as it has eight separate multipliers. The block diagram of the audio block is shown in figure 4.

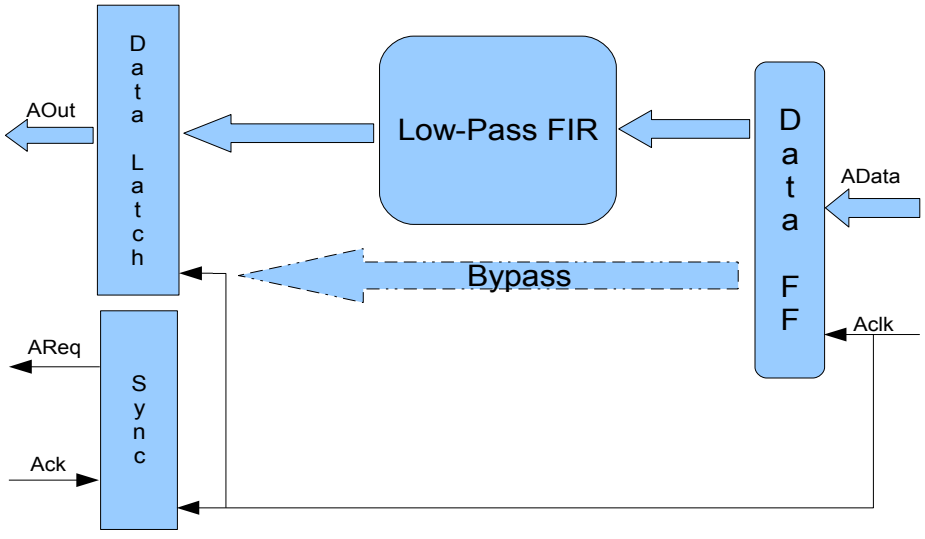

Figure 4: Audio Module Block Diagram

When audio arrives, it is locked into a data flip-flop to keep it valid. This allows the input data to change after the required hold time of the flip-flop. Data is clocked through a sequence of flip-flops to store it while it is being processed through the FIR filter. On the next audio clock, the filtered data will be latched for output to the processor FIFO and the Sync latch will be set to cause the A-REQ signal to

be asserted. This is turn indicates to the FIFO MUX that audio data is available. Whenever the ACK signal asserts, the Sync latch will reset the A-REQ signal. The asynchronous logic in the FIFO MUX will then reset the ACK signal.

2.3 Video Processor

Due to the difficulty of high speed video processing and limited development time for this design, video processing is limited to a simple extraction state machine. Additional processing would be placed at the output of this extraction. Example processing might include up-sampling or down-sampling, noise filtering, and other common image applications.

Data input into the video processor is in ITU BT.656 standard format for NTSC video. The output format is in ITU standard YUYV or 4:2:2 video format. As with the audio case, the sync logic produces a request signal on the clock edge and is cleared by the assertion of the ACK signal.

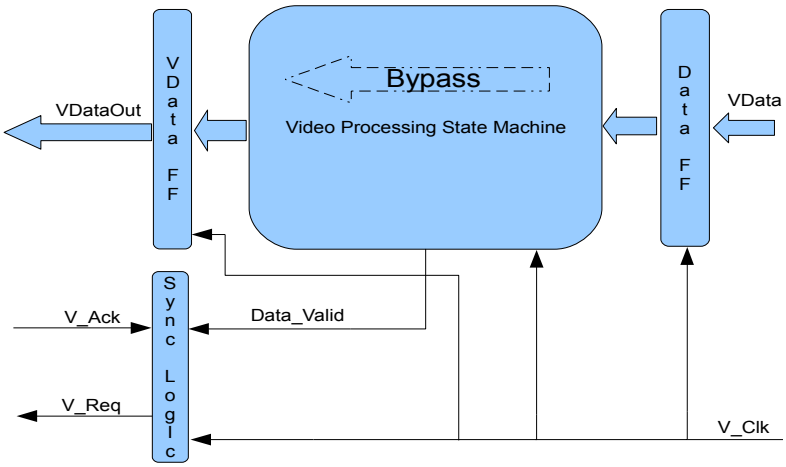

Figure 5: Video Module Top Level Diagram

2.4 Processor Interface

Three main blocks compose the processor interface. Nearest to the video and audio processing is the data join stage. This stage holds a very simple state machine that joins the data from the two sources into the next section asynchronous output FIFO. An asynchronous FIFO was chosen because it is not constrained by either the processor clock or the input data clocks. Finally, nearest to the processor interface, there is a block of logic meant to preserve the output data throughout the high phase of the input clock.

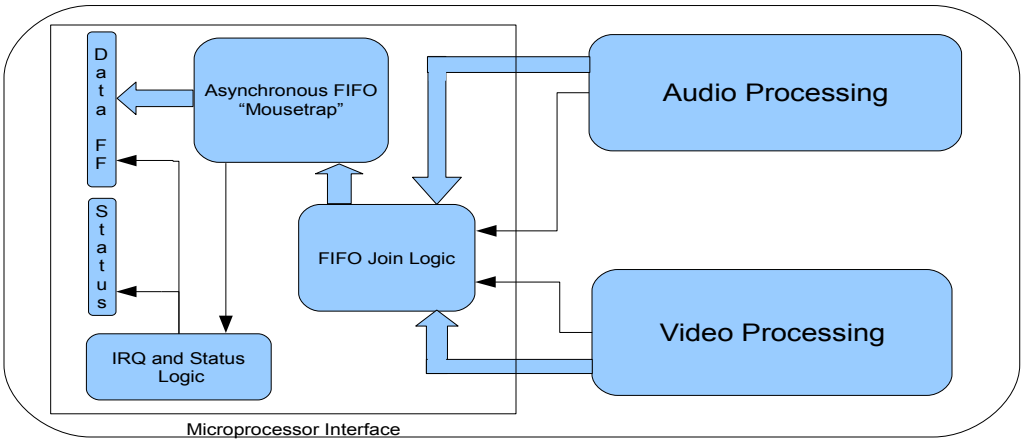

Figure 6: Processor Interface Module in System Context

2.4.1 Input Data Join

This stage is driven by the microprocessor clock. A pure combinational logic approach was very difficult to implement such that it met setup and hold time requirements. An arbiter was ruled out because it relies on a metastable analog circuit effect to obtain mutual exclusion of the signals. Employing this mechanism would not be conducive to some embedded environments and would require extensive development and test time, which raise the cost of an embedded component. Thus for this particular design, the best option seemed to be a synchronous clock join.

One of the limiting factors of the clock driven join was the goal to be able to clock in one video and one audio data item for each of their respective clocks. To meet this requirement, it was necessary to use the microprocessor clock rather than the much slower video clock. In reality, this constraint could have been relaxed at the cost of increased testing difficulty since one audio sample will last through several consecutive video frames and no data is produced during the video blanking intervals.

On the negative edge of the join clock, the data is selected and steered to the Asynchronous FIFO. This is safe because if the FIFO is busy, then the latch is closed. It will be transparent otherwise. Next, during the positive edge of the clock, the request line is driven if the FIFO is not actively busy pending on a request. It is here that overflow and data loss may occur. If the FIFO remains busy for more than a single clock, video data may be lost since it is only valid for 2.5 clocks and the signal may arrive too late for the first clock. Video data is thus preferred over audio data during the source selection process.

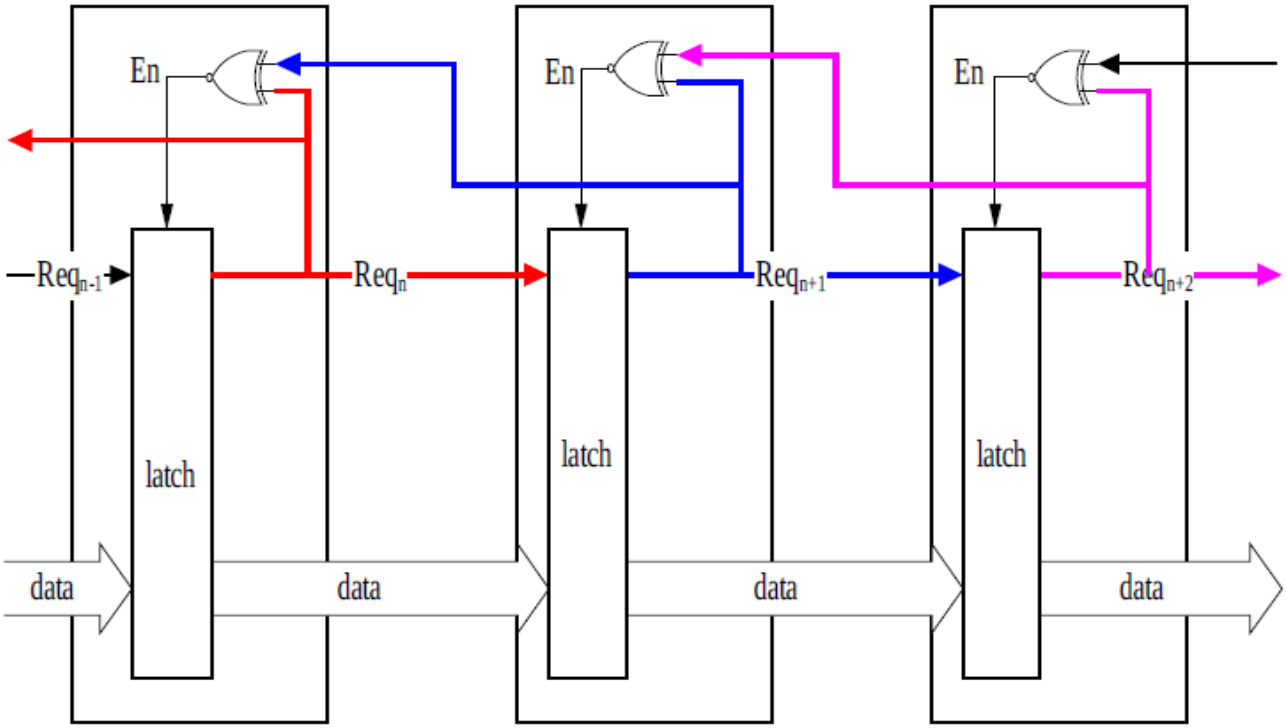

Figure 7: Asynchronous FIFO Design

2.4.2 Asynchronous FIFO

Since data is produced in two extremely different clock domains and consumed in yet another relatively prime clock domain, the simplest mechanism for bridging the domains is to use asynchronous logic for the data FIFO. A two-phase protocol makes the synchronization step much easier since only one transition must happen. That transition can then be keyed to a specific clock edge. The FIFO is composed of a series of self-timed mousetrap gates. The mousetrap is shown without request line buffers in Figure 7.

A mousetrap works by keeping a transparent latch whenever it is not occupied. Whenever a request is received, it is passed through the latch and used to disable the latch. As soon as an ACK from the next sequential gate is received, the gate is enabled, thus allowing another request to arrive.

Due to the timing of the processor interface explained in the next section, the FIFO is 32 entries deep. Additionally, the request line is buffered to slow it in relation to the data path.

2.4.3 Processor Interface Logic

At the interface to the processor, there is block that determines output status and interrupt state. Overflow is set if the positive audio or video clock edge occurs while the request line is still high. The interrupt is set when there are 16 data items in the FIFO. Unfortunately, it is very hard to determine how many items are in an asynchronous FIFO, so this design used a somewhat inefficient approach of waiting until all of the last 16 FIFO slots indicated that they were busy (the enable signal was not on).

Since the FIFO fills stacks up near the end, the data only needs to move one spot each clock in order to do burst reads from the FIFO. This design assumes that this movement will always occur in less than the single clock period for which it is required, with enough time remaining to meet setup time requirements. Hold time requirements are met by latching the return handshake signal to the FIFO. This implies that the clock skew within this module must be small.

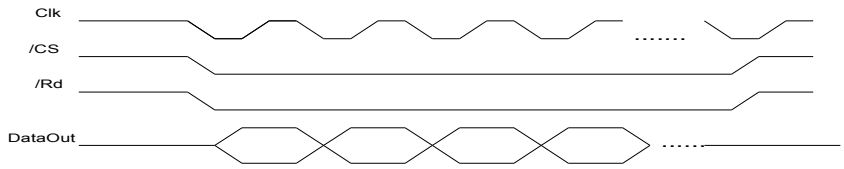

Figure 8: Read Cycle Timing

Assuming that the interrupt handling takes four bus cycles to clear the interrupt and up to 16 bus cycles before data reads begin, this allows up to four more items to arrive after the interrupt is asserted. Additionally, a maximum of seven elements may be added to the bus during the read cycle. This indicates the FIFO must be at least 28 entries long to prevent accidental overflow flagging. The interrupt state machine is show in Figure 9.

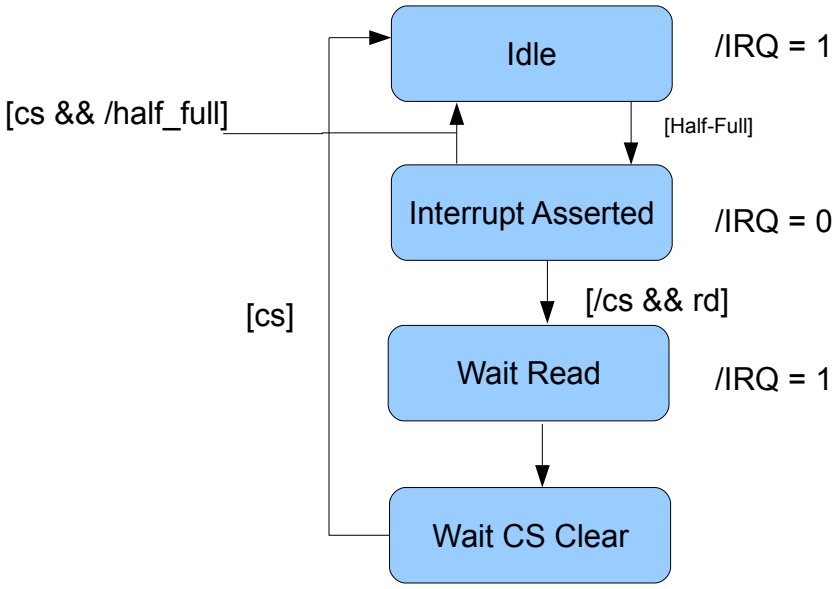

Figure 9: Interrupt State Machine

3 TESTING

3.1 Design For Test

The current design includes two features for design test. Both the audio and video module have a bypass signal. This signal causes the module to pass the input data directly to the FIFO join stage and bypass the processing. These features are beneficial both from a simulation standpoint and for circuit verification. Another bypass for the actual FIFO was considered, but time did not permit that addition.

3.2 Module Testing

For the audio and video testing, each module was simulated using a simple unit delay model and the final structural Verilog. The timing constraints on these modules were loose enough that these gave the correct answers, even with full speed simulated clocks. The Verilog test input read in the golden model files and processed them through the modules. Then the results were compared against the models' output for errors.

When testing the top module, it is not desirable to run the simulation long enough to get a significant amount of audio since the audio data comes very slowly. The resulting data sets would be unwieldy at best. So, the individual audio and video modules are tested against their golden model data sets, and the top level module is fed with those modules in the bypass state. This tests the join stage, FIFO, interrupt logic, and processor output more vigorously than a real stream of audio or video might since the buffers can be forced to always operate in a worst case scenario.

3.3 Test Results

3.3.1 Video Module

The video module passed the test exactly as expected. The following is the visual result of the video processing. Note how removing the sync lines has shifted the image.

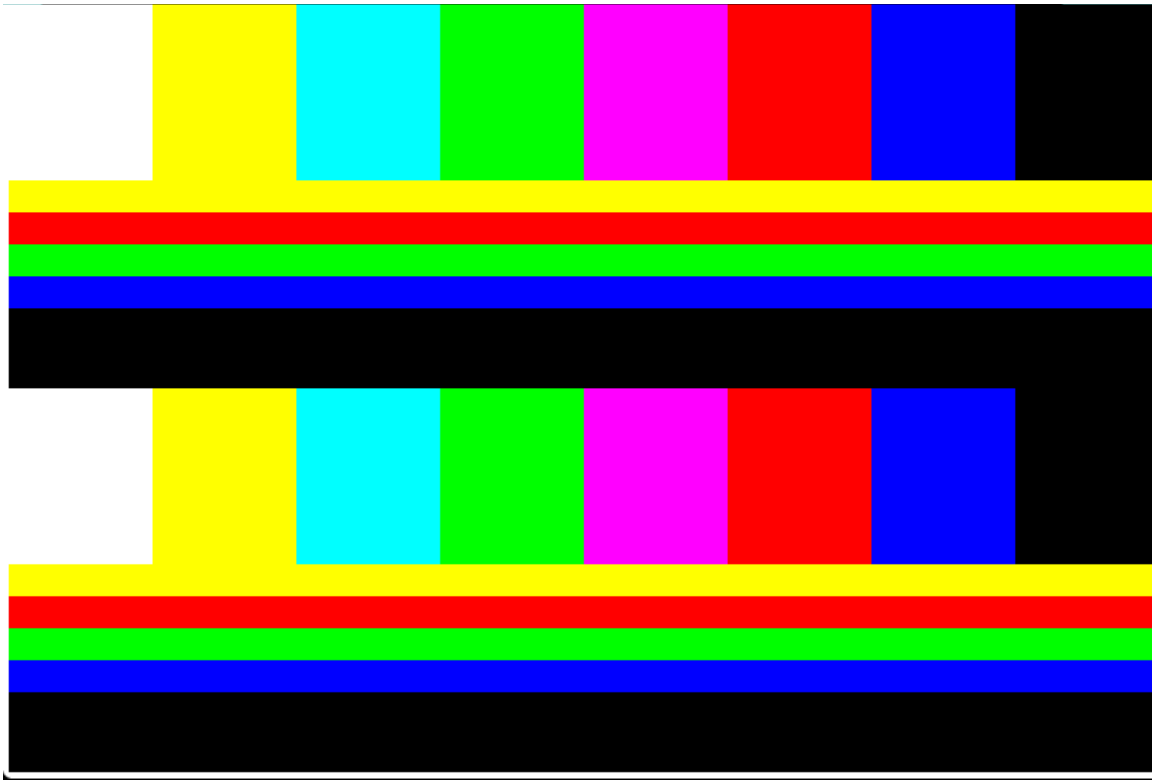

Figure 10: Visual Display of Video Results

3.3.2 Audio Module

A small discrepancy was found in the audio data. Occasionally, one or more of the values would come out exactly one off from the original. The cause of this problem is related to the golden model test set assumptions. The golden model assumed that the output would be truncated, but apparently the Synopsys design compiler actually built rounding into the circuit. This caused the off by one errors. This error magnitude is not of concern because the actual audio would still sound correct.

3.3.3 Top Level Module

Instead of testing the microprocessor module by itself, it was aggregated into the top level module test. With the audio and video bypass active, this was essentially the only item being tested. The top level module had too many gate delays to run with a unit delay simulation at full speed. However, it did function correctly if all the clocks were reduced by half. To get better results, the SDF annotation was inserted into Modelsim Verilog. However the library used for this test set did not function correctly with the SDF annotation, so the test was not completed.

4 Evaluation of Performance and Alternate Design

4.1 Design Performance

Overall, the FIFO performed as expected, but the input join stage ended up not working quite as well as was hoped. Instead of a simple logic join, the final join ended up being clocked as described in the design section above. With this change, the design no longer had any advantage over a typical clocked handshaking protocol. Further it was much harder to get the entire design working than it would have been had the Asynchronous FIFO been left out entirely.

4.2 Lessons Learned

The asynchronous FIFO performed very well given its specification. It was necessary to buffer the timing of the FIFO request signal to manage the overall timing correctly. However, the join stage is very, very difficult without a clock. Since the clocking was very different in speed, a simple C-element was not sufficient for the task and all attempts at a basic logic solution failed to perform to meet required timing for all cases. In the end, it was better and simpler to perform the join operation with a clock, to insure the data validity given the control signal timing.

Another major difficulty came when trying to determine the IRQ state from the FIFO. Since the FIFO is asynchronous, there is no simple way to figure out how much data is really in there. Counters were considered, but the added complexity was not desired. In the end, the IRQ was only set if all of the final sixteen stages had data.

One major advantage of the asynchronous FIFO is that it provides timing slack to the clocked design. By putting the join after the asynchronous FIFO, the IRQ control becomes much simpler as it is solely state based on how many items have been read from the FIFO. It also simplifies the timing for putting data into the FIFO as we assume by the nature of clocked design that it is present on the latching edge of the clock. The join then becomes a prioritized check of the request state of each FIFO at the selected clock edge. The timing is also relaxed because the FIFO has some room to queue up data.

4.3 Alternate Design

With the lessons learned from this project, the following design provides a more interesting follow up for accomplishing the goal of this paper. Note that the state logic would include a clocked FIFO for holding data during interrupt processing. This FIFO is nominally the maximum size of a burst read from the processor host device. Asynchronous FIFO blocks provide timing slack and storage during the read stage when output to the host processor is pending. The video and audio modules would remain as currently designed since their outputs can easily feed directly into an Asynchronous FIFO.

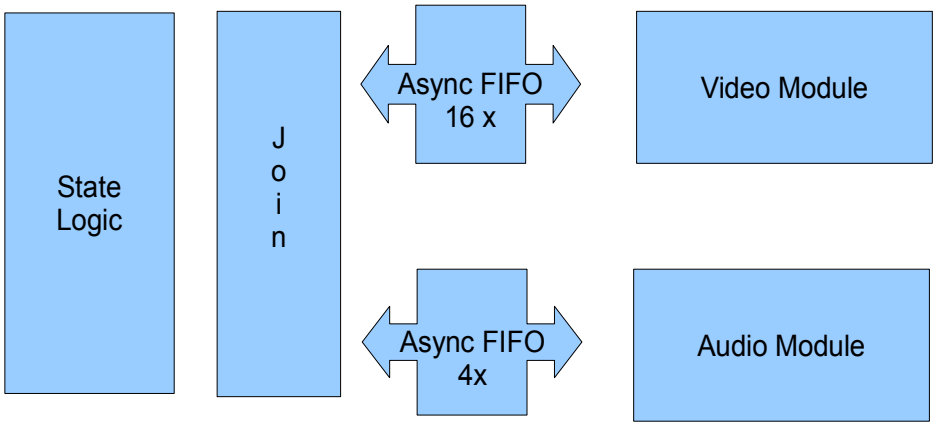

Figure 11: Alternate Design Hierarchy

5 LVS and DRC

The modules were not joined into a single layout, but for each module layout, LVS and DRC were performed against the output of the place and route tool. For all of these circuit, LVS and DRC was successful with no errors and all design layouts matching the imported structural layout from the tool.

6 Schedule Performance

Overall the schedule was fairly well met. Tasks that came in behind schedule were the audio golden model generation, design synthesis, and final layout. The design synthesis was late because of difficulties with the join stage logic and some odd behaviors due to an invalid structural file for the Asynchronous FIFO that caused the Verilog simulators to hang. These in turn caused the layout to be deferred until the design could be completed.

7 References

[1] Montek Singh & Steven M. Nowick, 2004. *Ultra-High-Speed Transition-Signaling Asynchronous Pipelines*. http://nthucad.cs.nthu.edu.tw/TingTingHwang/paper_present/2004.09.17_gunking/20040917gunking.ppt

[2] Bowhill and Fox, 2001. "*Design of High-Performance Microprocessor Circuits*". IEEE Press.

[3] Spacewire, UK 2001. *A Brief Introduction to Digital Video* http://www.spacewire.co.uk/video_standard.html