Final Report:

Ad Remover

ECE532 – Digital System Design

By: Gianluca Sottile, Michal Porzuczek, Hua Sun
4/2/2007

Table of Contents

1.0 Overview 1
1.1 Goals 1
1.2 System Block Diagram 1
1.3 Brief Description of IP 2
1.4 Brief Description of Software 2
1.5 Clock Domain 2
2.0 Project Outcome 2
2.1 Review of Original Plan 2
2.2 Results 4
2.3 Difficulties and Solutions 4
2.3.1 External Memory Limitation 4
2.3.2 Memory Access Delay 4
2.4 Suggestions for the future 5
3.0 Description of Hardware Blocks 6
3.1 Xilinx IP 6
3.1.1 Microblaze_0 (MicroBlaze) 6
3.1.2 OPB (opb_v20) 6
3.1.3 FSL_GET_LENGTH / FSL_SET_THRESH (fsl_v20) 7
3.1.4 ILMB / DLMB (lmb_v10) 7
3.1.5 ILMB_CNTLR / DLMB_CNTLR (lmb_bram_if_cntlr) 8
3.1.6 RS232_UART_1 (opb_uartlite) 8
3.1.7 I2C (opb_iic) 8
3.1.8 LMB_BRAM (bram_block) 9
3.1.9 DCM_0 (dcm_module) 9
3.2 Custom or Modified IP 9
3.2.1 VIDEO_CAPTURE_0 (video_capture) 9
3.2.2 BS_WRAPPER_0 (bs_wrapper) 11
4.0 Description of Software 13
4.1 Video Capture Initialization 13
4.2 Threshold Assignment 13
4.3 Real-Time Commercial Notification 13
5.0 Directory Tree Description 14

6.0 References 14
6.1 Module References 14
6.2 Document References 14
Appendix A: ASM Charts 15
A-1 Black Screen Detect (bs_detect.v) ASM Chart 15
A-2 Ad Detect (ad_detect.v) ASM Chart 16
Appendix B: Simulation Waves 17
B-1 bs_detect.v 17
B-1 ad_detect.v 18

Table of Figures

Figure 1 – Final System Block Diagram 1
Figure 2 - Original System Block Diagram 3
Figure 3 - Fast Simplex Link (FSL) Block Diagram 7
Figure 4 - Video Capture Block Diagram 10
Figure 5 - bs_wrapper Block Diagram 11
Figure 6 - VGA Timing Signals 12
Figure 7 - Black Screen Detect ASM Chart 15
Figure 8 - Ad Detect ASM Chart 16
Figure 9 - bs_detect Simulation 17
Figure 10 - ad_detect Simulation 18

1.0 Overview

1.1 Goals

The goal of the project was to develop a device that would filter out advertisements from television signals. This device could then be combined with a PVR or VCR to record television shows without any commercials for a more enjoyable viewing experience.

The system is designed based on the assumption that commercial advertisements have black screen transitions between them and that they are multiples of 15 seconds in length.

1.2 System Block Diagram

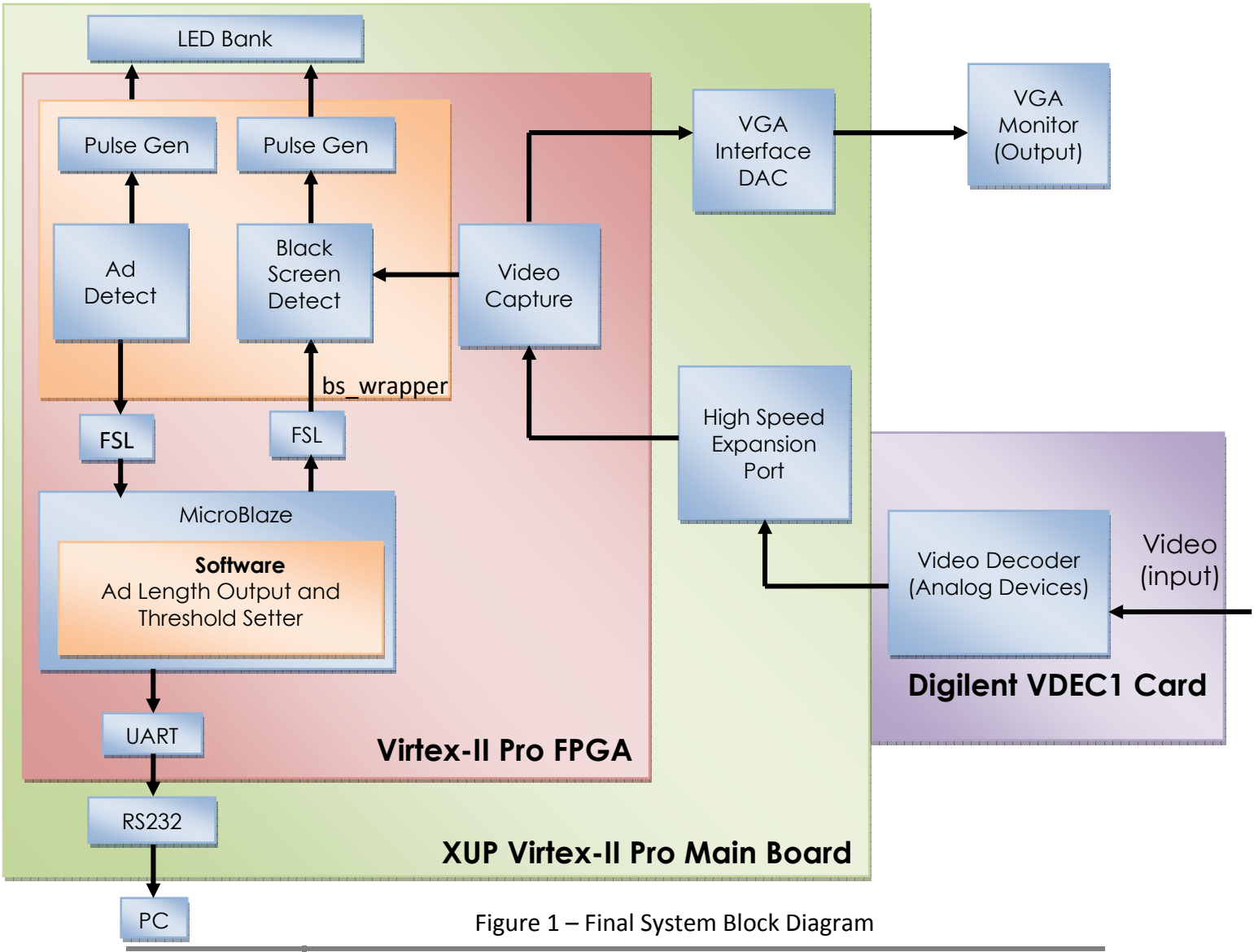

Figure 1 – Final System Block Diagram

1.3 Brief Description of IP

Hardware Block	Function	Origin
MicroBlaze_0	See section 3.1.1	Xilinx IP
OPB	See section 3.1.2	Xilinx IP
FSL_GET_LENGTH/FSL_SET_THRESH	See section 3.1.3	Xilinx IP
ILMB/DLMB	See section 3.1.4	Xilinx IP
ILMB_CNTLR / DLMB_CNTLR	See section 3.1.5	Xilinx IP
RS232_UART_1	See section 3.1.6	Xilinx IP
I2C	See section 3.1.7	Xilinx IP
LMB_BRAM	See section 3.1.8	Xilinx IP
DCM_0	See section 3.1.9	Xilinx IP
VIDEO_CAPTURE_0	See section 3.2.1	Custom IP
Ad Detect	See section 3.2.2.3	Custom IP
Black Screen Dectect	See section 3.2.2.2	Custom IP
Pulse Gen	See section 3.2.2.1	Custom IP

1.4 Brief Description of Software

The software component running on the MicroBlaze has three purposes. The first is to configure the video decoder board, the second is to print out the lengths of the commercials and the third is to set the threshold of the black screen detector.

1.5 Clock Domain

The use of the video capture block forces the design to have two different clock domains. The first domain, 27 MHz, was necessary for the video capture components along with bs_wrapper, and the second, 100 MHz, for the MicroBlaze. These domains were effectively connected using asynchronous FSLs.

2.0 Project Outcome

2.1 Review of Original Plan

The original plan was to detect and remove advertisements from television signals. The video frame would be captured using the video capture block, and then sent using an FSL to the MicroBlaze. From the MicroBlaze the information would be sent to an External Memory buffer. In the mean time, the MicroBlaze would access the written frame information from the memory and examine it for black screens and the existence of commercials. Finally the frames would be sent back through an FSL to the video capture block and output to a VGA Monitor. (See Figure 2 below)

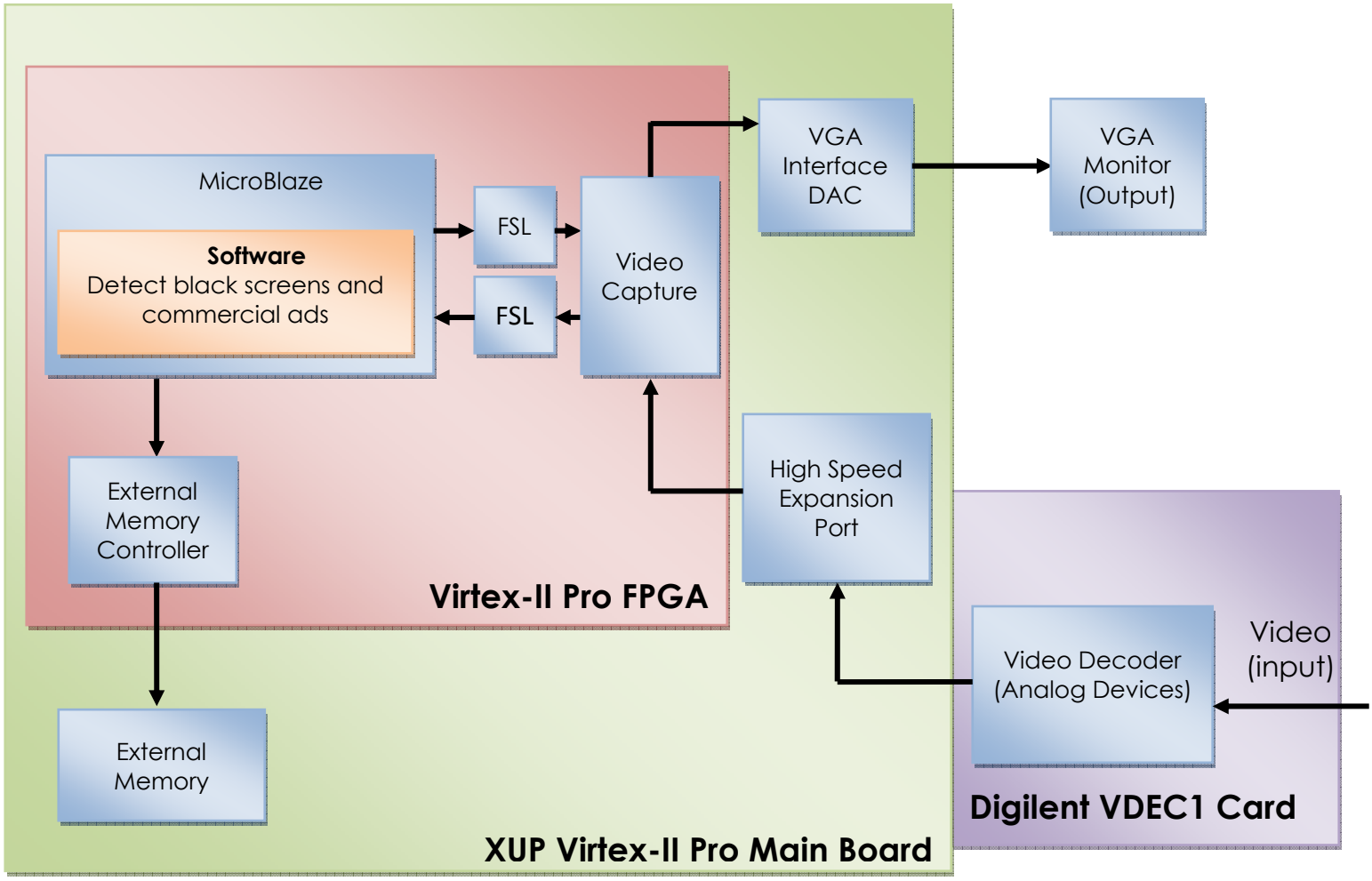

Figure 2 - Original System Block Diagram

2.2 Results

The final result, although not exactly as planned originally, is still a viable solution to the problem the project aimed to solve: removing commercials from a television signal.

The final product does not buffer the video and eliminate the commercials from the output stream itself, but instead flags the beginning and end of a commercial and its length. The device, if connected to a PVR, could pass the flags and commercial length to the recording device which can subsequently write over a commercial after it has been recorded. Thus, the proof of concept remains as initially proposed.

2.3 Difficulties and Solutions

A number of difficulties were encountered throughout the development of the project and will be discussed in detail in the following sub-sections.

2.3.1 External Memory Limitation

The external RAM available with the XUP VIRTEX II Pro is 256 MB. Each frame, acquired every 33 msec, has a resolution of 720 by 480 pixels, whereby each pixel is represented by an 8 bit red, green and blue channel. The group calculated that without any data loss, a single frame of video would occupy 1.04 MB, and therefore the external memory could only store 8.23 seconds of information. This is not sufficient to catch even the shortest commercial.

The group looked into a variety of solutions: interlacing, cutting down on colour bits, decreasing the capture rate, and using grayscale video. It was decided that grayscale video was more visually appealing than any of the other alternatives. The use of grayscale would allow storage of 25 seconds of video. This would be enough to capture the shortest commercials and provide proof of concept that with larger memory the system would operate equally well for longer commercials.

2.3.2 Memory Access Delay

The second major problem the group encountered was a delay in writing to and reading from memory. The group decided on using the MicroBlaze's to write to memory instead of implementing custom hardware to interact between the video capture block and the memory. The method involved connecting the video capture block, operating in the 27 MHz domain, to the MicroBlaze, operating in the 100 MHz domain, using two asynchronous FSLs.

The idea was to send the frame information from the video capture block to MicroBlaze using one FSL, and then write to memory. Once in memory, the frame could be processed and sent back to the video capture block using the second FSL. Upon

receiving the information, the data would be sent through the DAC and out to the VGA monitor.

Timing constraints hindered the success of this stage. The MicroBlaze requires 2 clock cycles to read from the video capture module through the FSL and another 2 to write. Furthermore it requires 1 clock cycle to either read or write to memory. This meant that a total of 6 clock cycles were accumulated in this process. With the MicroBlaze clock frequency at 100 MHz and the video decoder requiring information at 27Mhz the delay incurred by the processor's read/write operations were too much for the system to operate correctly.

This realization occurred at the 4th milestone in the project and left the group with only one alternative: to remove the buffering aspect of the project. The input video signal would be flagged for black screens and detect commercial ads in real-time. The solution alleviates the first difficulty the group faced, so the system could again operate in colour and could detect commercials of any length.

2.4 Suggestions for the future

The first suggestion is to resolve the buffering problem. One proposition would be to implement a custom interface between the external memory and the video controller to allow buffer reading/writing to be done in hardware.

A second suggestion, if the buffering had been completed successfully, would be to actually write over the detected commercials that were stored in the RAM. This would increase the useful buffer size.

Finally, if 2 GB of RAM were put into the system then it would be possible for the system to buffer data in colour and capture commercials of any length. In addition to this increase in memory capacity, video compression could be used to reduce the amount of space occupied by raw video frames.

3.0 Description of Hardware Blocks

3.1 Xilinx IP

This section contains a brief description of Xilinx-created modules and how they were used in the context of the Ad Remover system.

3.1.1 Microblaze_0 (MicroBlaze)

The MicroBlaze is an embedded processor that was created and optimized for use in Xilinx FPGAs. It has 32 general purpose 32-bit registers, 32-bit instructions with three operands and two addressing modes [MicroBlaze]. Some of the optional features of the MicroBlaze that were used include:

- On-chip Peripheral Bus (OPB) data and instruction side interfaces
- Local Memory Bus (LMB) data and instruction side interfaces
- Fast Simplex Link (FSL) interfaces
- Instruction/Data cache over CacheLink (IXCL/DXCL interfaces)

System Use

The MicroBlaze is used to execute the Software modules in the system using the LMB instruction and data bus interfaces to connect to memory. The IXCL/DXCL interfaces are used to decrease instruction and data retrieval times through caching.

The Digilent VDEC1 video decoder board needs to be configured before it can be used. Software is used to connect to the IIC module over the OPB bus in order to complete this configuration. Two FSL interfaces on the MicroBlaze are enabled to connect with the bs_wrapper module. One is used to set the bs_detector's noise threshold and the other, to retrieve the commercial length from the ad_detector.

3.1.2 OPB (opb_v20)

The On-Chip Peripheral Bus (OPB) is used to connect multiple pieces of hardware together using the an OPB Arbiter (opb_v20). It is responsible for managing what hardware has access to the bus resource at what time. It can support up to 16 masters and any number of slaves [OPB].

System Use

The OPB is the backbone of the system. It is used to connect vital hardware peripherals to the MicroBlaze processor. The DEBUG_MODULE, RS232_UART_1 and I2C modules are all connected as slaves on the OPB with the MicroBlaze as the only master.

3.1.3 FSL_GET_LENGTH / FSL_SET_THRESH (fsl_v20)

The Fast Simplex Link (FSL) Bus is a one way communication channel between two hardware peripherals. It operates as a first-in-first-out (FIFO) data queue as seen in Figure 3. The master of the bus pushes data in when FSL_M_Write is set and the FSL is not full (FSL_M_Full not asserted). The slave can read out data when data exists (FSL_S_Exists is asserted) and the FSL_S_Read signal is set [FSL].

The FSL can operate in synchronous or asynchronous mode. Synchronous mode requires that FSL_M_CLK and FSL_S_CLK be the same clock signal where as asynchronous mode allows two different clock signals to be used for master and slave.

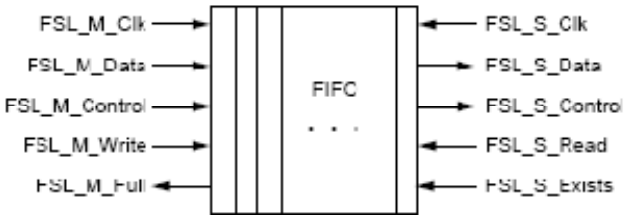

Figure 3 - Fast Simplex Link (FSL) Block Diagram

System Use

FSL_SET_THRESH is used to set the black screen detector's noise threshold in bs_wrapper from the MicroBlaze. This feature was implemented to reduce the compile time needed when the noise threshold was being altered frequently. The threshold is only written once at the beginning of the codes execution. See section 4.2 for details of the software implementation.

The threshold is written to the FSL in software and is registered by the bs_wrapper. This FSL operates in asynchronous mode because of the different clock domains between the MicroBlaze and the bs_wrapper.

FSL_GET_LENGTH is used to get the detected commercial lengths from the ad_detector in bs_wrapper to the MicroBlaze. The 4 least significant bits of the 32-bit bus are used to write the commercial length indicators at 27Mhz to the MicroBlaze reading at 100Mhz. See section 4.3 for details of the software implementation.

3.1.4 ILMB / DLMB (lmb_v10)

The Local Memory Bus (LMB) is used as a fast was to connect the instruction and data ports of the MicroBlaze to the on-chip blocks of memory (BRAM) [LMB].

System Use

The ILMB and DLMB are used to connect the instruction and data ports of the MicroBlaze to the LMB_BRAM block in the system via LMB BRAM Interface controllers. See section 3.1.6 for more details of these controllers.

3.1.5 ILMB_CNTLR / DLMB_CNTLR (lmb_bram_if_cntlr)

The LMB BRAM Interface Controller is the interface between the LMB and the bram_block peripheral. The MicroBlaze can only therefore connect to a Bram block through such a controller [BRAMIC].

System Use

The bram block used has two ports which correspond to instruction and data sections in memory. Two LMB BRAM Interface Controllers are needed to connect each of these ports to the LMB buses in the system. ILMB_CNTLR connects the instruction port (Port A) of the bram block to the ILMB bus while DLMB_CNTLR connects the data port (Port B) of the bram block to the DLMB bus.

3.1.6 RS232_UART_1 (opb_uartlite)

The OPB UART Lite module connects the OPB bus to the serial port of the board. This allows the user to interact with the MicroBlaze by sending and receiving characters over this port via the OPB [UART].

System Use

The Ad Remover system is configured so that software running on the MicroBlaze processor that prints to ‘stdout’ will have the characters forwarded to the RS232_UART_1. Any characters received from ‘stdin’ also come from the RS232_UART_1.

3.1.7 I2C (opb_iic)

The OPB IIC Bus Interface (opb_iic) connects the Philips IIC bus v2.1 to the OPB. This IIC bus can be tailored to various applications depending on its configuration settings [I2C].

System Use

The system uses the I2C Module to configure the Digilent VDEC1 Video Decoder board. Software running on the MicroBlaze writes to this interface which in turn configures the VDEC1. See section 4.1 for details on the software implementation.

3.1.8 LMB_BRAM (bram_block)

The Block RAM (BRAM) is a configurable memory block that can be connected to several different BRAM Interface Controllers. The HDL of the block is generated by the EDK based on the configuration of the interface controller connected to it [BRAM].

System Use

The LMB_BRAM module connects to the ILMB_CNTLR and DLMB_CNTLR to provide the MicroBlaze with an instruction and data memory space over two separate LMB buses (ILMB/DLMB).

3.1.9 DCM_0 (dcm_module)

The Digital Clock Manager (DCM) Module implements a delay locked loop, frequency synthesizer, digital phase shifter or digital spread spectrum [DCM].

System Use

The DCM_0 is used as a delay locked loop that ensures the output clock signal (sys_clk_s), which drives many hardware blocks in the system, has the frequency specified (100Mhz). The hardware blocks driven by the DCM_0's 'sys_clk_s' output include:

- MicroBlaze Processor
- OPB
- LMB Buses
- FLS Modules
- I2C Module

3.2 Custom or Modified IP

This section contains an in-depth description of the custom hardware implemented in the Ad Remover system.

3.2.1 VIDEO_CAPTURE_0 (video_capture)

The video_capture block is a Xilinx pcore which was taken from the Demonstration & Reference Designs section of the Xilinx website [vidcap]. The main purpose of the block is to get interlaced 4:2:2 YCrCb digital video from the video decoder and convert it to a

24 bit de-interlaced RGB video to be sent to the VGA monitor. The block operates at a 27 MHz clock rate.

First the block generates the necessary Hsync, Vsync, Blanking and Pixel Clock signals. It then does a chroma sub-sampling conversion from 4:2:2 to 4:4:4, followed by a colour space conversion from YCrCb to RGB. Once the conversions are complete then the system de-interlaces the digital video by using line doubling. The video with its Hsync, Vsync, Blanking and Pixel Clock signals are sent to the VGA Interface DAC to be output to a VGA Monitor. The block also contains an I2C Master which interacts with the decoder and allows the user to select either, composite video output, component video output, or super video output. See Figure 4.

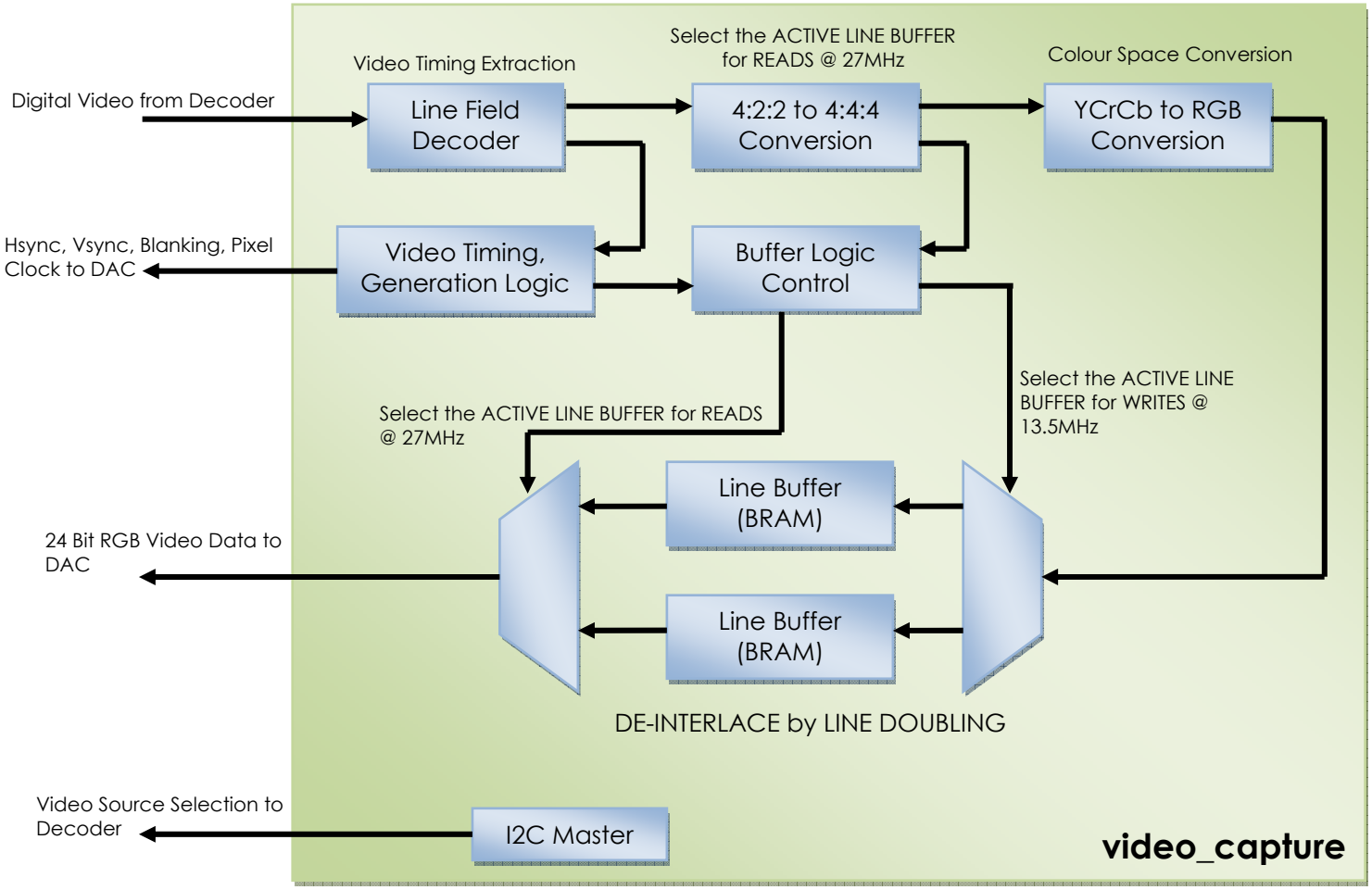

Figure 4 - Video Capture Block Diagram

3.2.2 BS_WRAPPER_0 (bs_wrapper)

The entire bs_wrapper is clocked with same 27Mhz clock used in the video_capture module (PIXEL_CLOCK). The RGB pixel values are changing at this rate and therefore the block must operate the same rate in order to sample all the incoming pixel values.

The bs_wrapper module houses the three main components:

1. Black Screen Detect (bs_detect.v)
2. Ad Detect (ad_detect.v)
3. Pulse Gen (pulsegen.v)

Each component is described in detail in the following subsections. Simulation testing was performed on all modules. See Appendix B for simulation waveforms of the relevant modules.

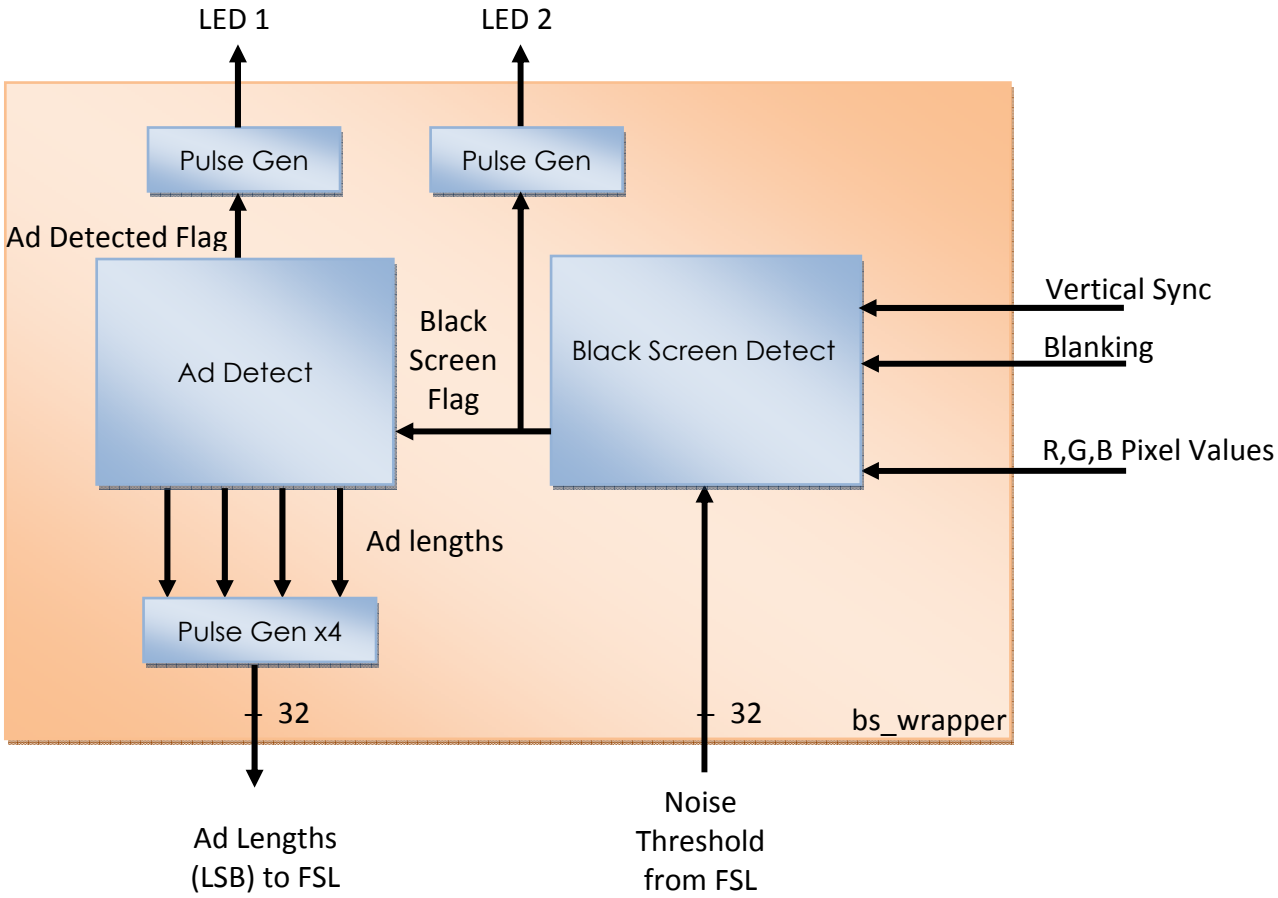

Figure 5 - bs_wrapper Block Diagram

3.2.2.1 Pulse Gen Module (pulsegen.v)

As described above bs_wrapper is clocked at a rate of 27Mhz. Since both the Black Screen Detect and Ad Detect modules are also clocked at this rate, the output flags they

produce are only asserted for 37ns (one 27Mhz clock cycle). If the flag was directly connected to an LED light, it would produce no visually perceptive change in the LED's state. This is why a pulse generator module was created. It is a simple FSM that waits for the input flag to be asserted and using a counter to generate a pulse that remains asserted for 1 second.

Due to the simplicity of this module there was no relevant simulation testing performed.

3.2.2.2 Black Screen Detect Module (bs_detect.v)

The concept behind the Black Screen Detect Module is to use a saturating accumulator that adds the amount of "whiteness" in a video frame. If at any point the "whiteness" exceeds a certain noise threshold the screen will be considered not black. Conversely if the amount of "whiteness" in the entire frame does not exceed the same threshold, the frame is considered a black screen. The RGB pixel values being outputted to the VGA monitor are used to measure this qualitative idea of "whiteness".

Three counters were used to accumulate each of the Red, Green and Blue pixel values. The VGA blanking and vertical sync signals are inputs to this module because they determine when the values on the RGB lines are valid. The vertical sync determines the frame boundaries while the blanking determines if pixels are being written to the screen. See Figure 6.

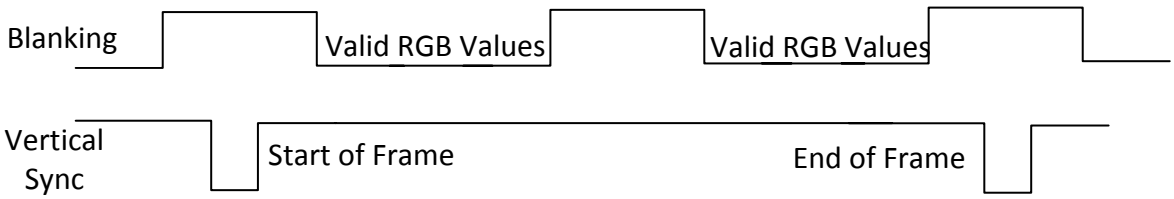

Figure 6 - VGA Timing Signals

The three counters described above are enabled only when blanking is not asserted. They are also reset at the end of each frame. At any moment if all of the counter values are greater than the input noise threshold (received from the FSL) the counters are reset and the detector waits for the end of the frame to re-enable them. If, however, the end of the frame is reached without the counter values being less than the threshold, the black screen flag is set. See Appendix A-1 for the ASM chart of this module.

3.2.2.3 Ad Detect Module (ad_detect.v)

This module builds on the progress made by the implementation of the Black Screen Detect module. Its only input is the Black Screen Flag. It uses a counter to measure the

temporal distance between successive flags. The counter is enabled at the occurrence of a black screen and continues counting until the occurrence of the following black screen. If the counter value corresponds to 15, 30, 45 or 60 seconds the appropriate ad length signal is asserted. The output signal that drives LED2 is just the "or" of these four length signals passed through a pulse generator. See Appendix A-2 for the ASM chart of this module.

4.0 Description of Software

4.1 Video Capture Initialization

This section of code was adapted from the original Xilinx sample project [vidcap]. Initially the code required user input to decide which input source (composite, component or super video) the video capture board would use. Since the input video source used for testing and demonstration purposes had a composite output, the code was made to initialize the decoder to only use the composite input.

4.2 Threshold Assignment

In the black screen flagging block, three counters are used to calculate the total RGB values of a video frame. A totally black pixel has RGB values of 0, 0, 0. Ideally, a totally black screen would have its RGB counter values all equal to 0. However, due to noises and disturbances, some pixels in a black screen might appear non-black. To deal with this variation, the program allows the user to adjust the threshold value.

This part of the program sets the noise threshold for determining whether a frame is black to the FSL using the "putfsl()" command. Since the slave interface to this FSL (bs_wrapper) only reads from the FSL if data exists, the data only has to be written once.

4.3 Real-Time Commercial Notification

This piece of code constantly checks for a change in the least significant bits (LSB) being read from the FSL. The program will output the corresponding messages to the UART: "15 Second Commercial", "30 Second Commercial", "45 Second Commercial", or "60 Second Commercial" accordingly.

The outgoing pulse generators in bs_wrapper causes the software to detect the same ad several times. This is resolved by the Boolean flag 'once' that determines if the add was displayed once before printing it again. See source code for implementation details.

5.0 Directory Tree Description

A README file has been included in the design tree to mirror the content of this section. The following table illustrates the important folders in the design tree and what purpose they serve.

Folder Name	Description
doc	Contains a copy of this document and the final group presentation.
archive	Contains several versions of the project working directory through the various stages of development.
ref	Contains the working directories of the base Xilinx project.
ad_remover	Contains the final version of the projects working directory. The HDL of all custom hardware modules is located in the 'pcores' subfolder. The software source code is located in the 'config_decoder_revb' subfolder.

6.0 References

6.1 Module References

[vidcap] video_capture_1_1
http://www.xilinx.com/univ/XUPV2P/Reference_Designs/edk_7_1_builds/video_capture_rev_1_1/video_capture_rev_1_1.zip

6.2 Document References

[BRAM] Xilinx Inc. Block RAM (BRAM) Block (v1.00a). San Jose, 21 August 2006.
[DCM] —. Digital Clock Manager (DCM) Module (v1.00a). San Jose, 2 December 2005.
[FSL] —. Fast Simplex Link (FSL) Bus (v2.00a). San Jose, 12 July 2006.
[BRAMIC] —. LMB BRAM Interface Controller (v2.00a). San Jose, 3 May 2006.
[LMB] —. Local Memory Bus (LMB) V1.0 (v1.00a). San Jose, 22 February 2006.
[MicroBlaze] —. MicroBlaze Processor Reference Guide. San Jose, 29 August 2006.
[OPB]—. On-Chip Peripheral Bus V2.0 with OPB Arbiter(v1.10c). San Jose, 31 August 2006.
[I2C] —. OPB IIC Bus Interface (v1.02a). San Jose, 28 September 2006.
[UART]—. OPB UART Lite (v1.00b). San Jose, 24 January 2006.

Appendix A: ASM Charts

A-1 Black Screen Detect (bs_detect.v) ASM Chart

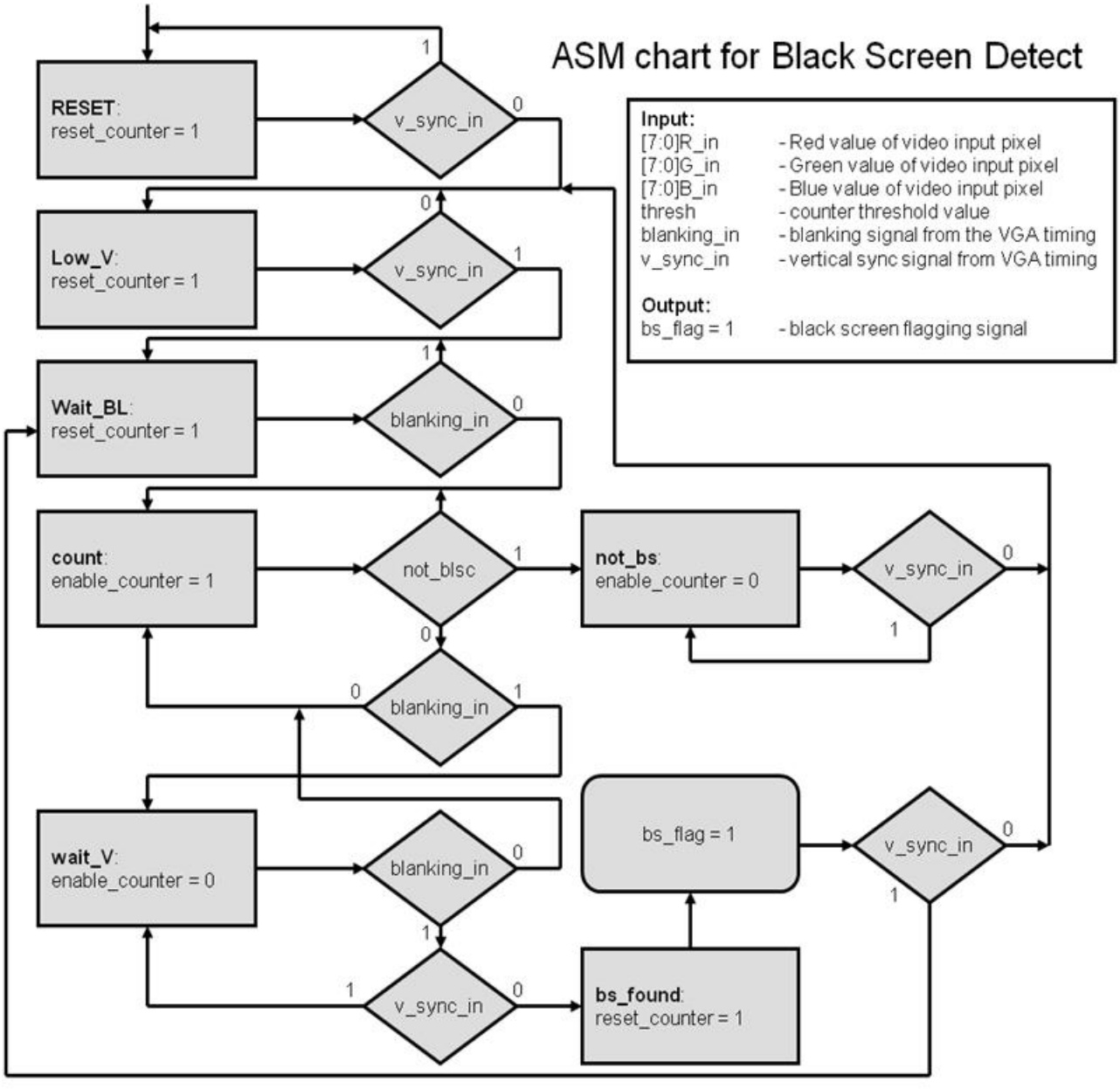

Figure 7 - Black Screen Detect ASM Chart

A-2 Ad Detect (ad_detect.v) ASM Chart

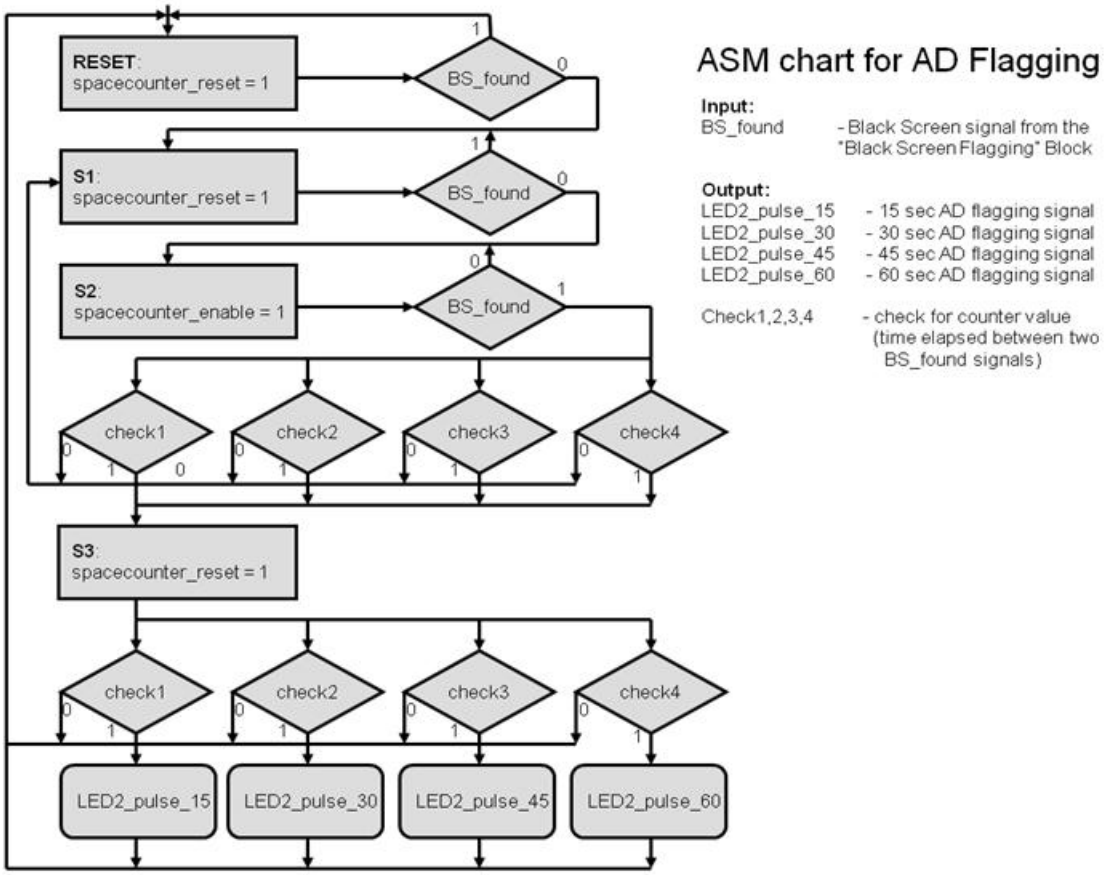

Figure 8 - Ad Detect ASM Chart

Appendix B: Simulation Waves

B-1 bs_detect.v

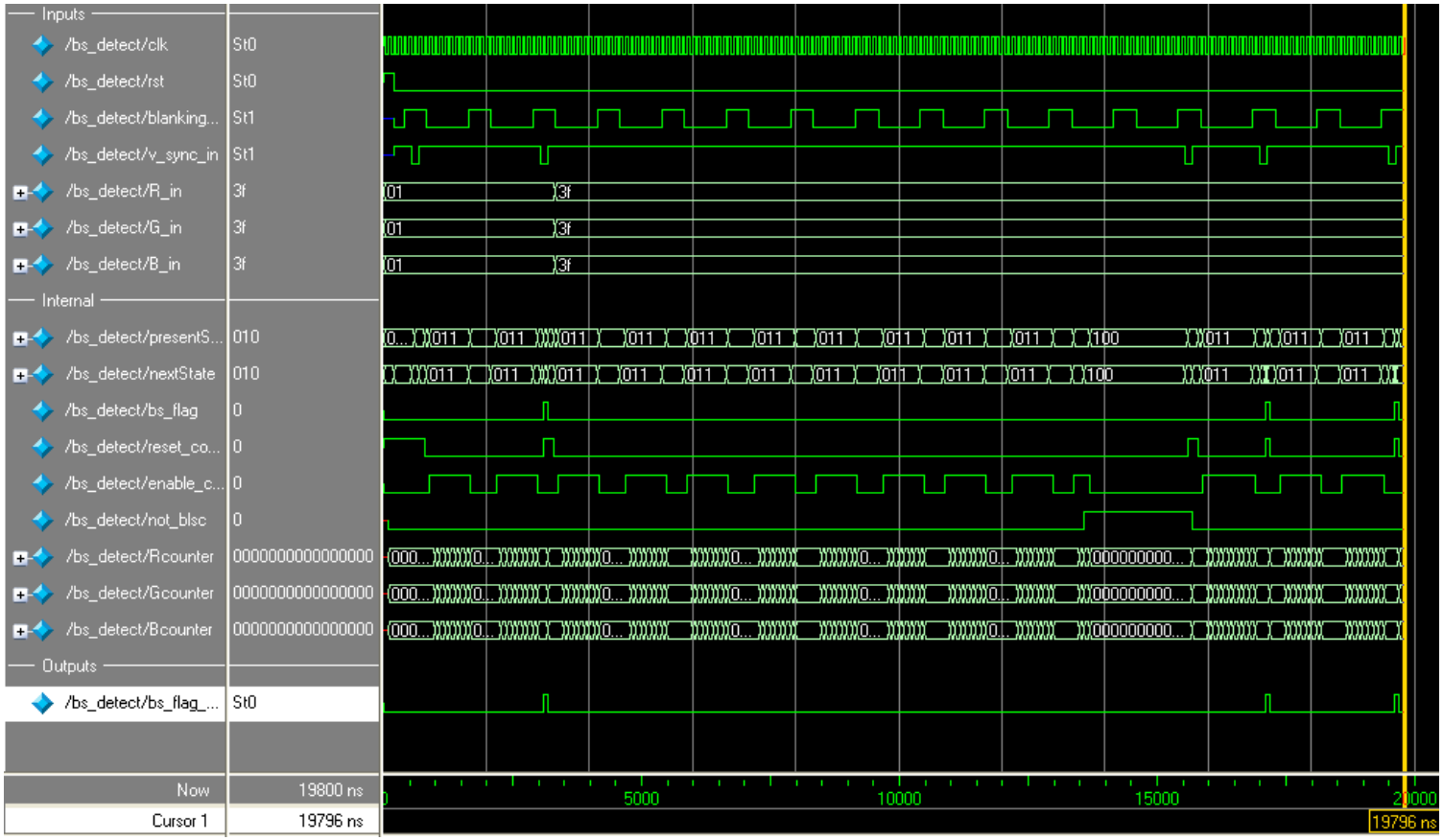

Figure 9 - bs_detect Simulation

B-1 ad_detect.v

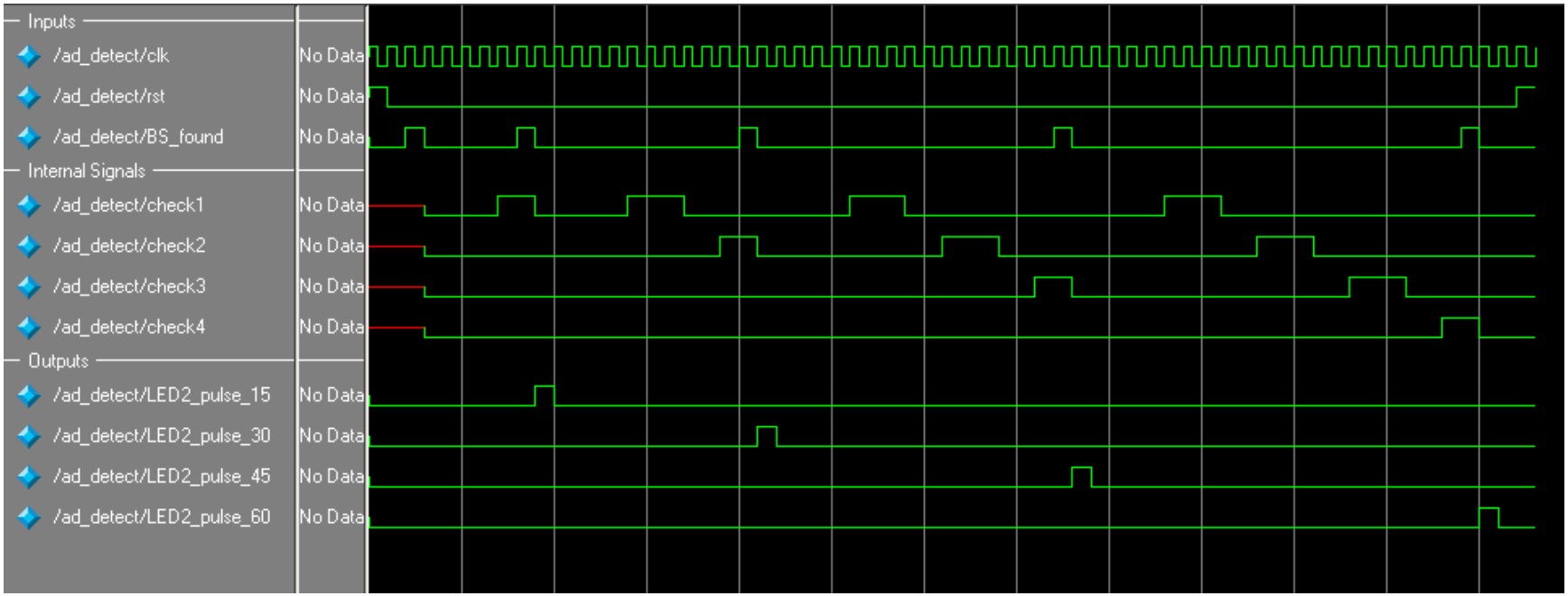

Figure 10 - ad_detect Simulation