Mini Project - VGA Text, Mouse, and LFSR

Maryam Hemmati

Department of Electrical, Computer and Software Engineering
University of Auckland

email: m.hemmati@auckland.ac.nz

COMPSYS 305-Digital Systems Design

17 April 2024

Content

1. VGA Interface Review
2. Text Display on VGA Screen
3. PS/2 Mouse Interface
4. Linear Feedback Shift Register (LFSR)

VGA Interface - Review

Image on VGA screen is displayed by turning the pixels ON and OFF.

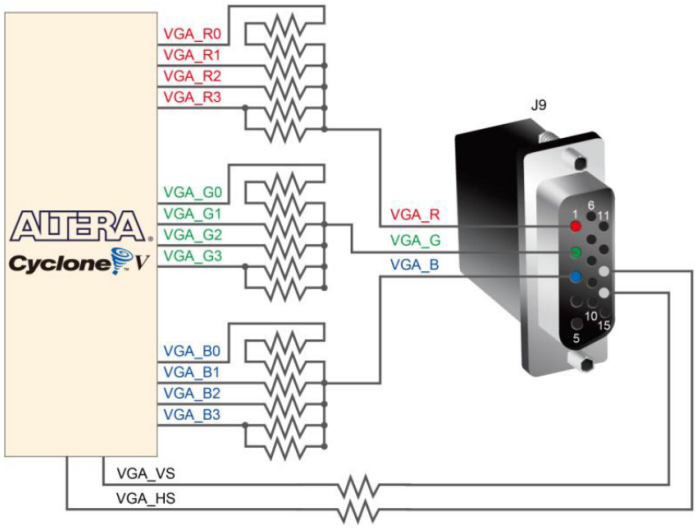
- Video signal must redraw the entire screen 60 times per sec (**60Hz**) to avoid flickers.
  - Human eyes detect flickers at refresh rate less than 30Hz.
- We will use the common VGA display standard at **25MHz** pixel rate with **640x480** resolution.
  - Each pixel takes 40ns at 25MHz pixel rate.

VGA Interface - Review

VGA video standard contains 5 active signals:

- **Horizontal** and **vertical synchronisation** signals.
- Three analog signals for **red**, **green** and **blue** (**RGB**) colours formation.
  - By changing the analog voltage levels of the RGB signals, different colours can be produced.
  - Depending on the number of bits supported by the development board, different amount of colours can be represented.

VGA Interface - DE0-CV Board - Review



VGA Interface - VGA_Sync Component - Review

We need a component to drive the control signals to the display and provide pixel values at the right rate.

- In order to generate the VGA signal at 25 MHz, the clock signal provided by DE0-CV (50MHz) needs to be halved.
- 25 MHz clock signal can be used by counters to generate the horizontal and vertical sync signals.
- The counters also represent row and column address of a pixel, which can be used by other components to retrieve pixel information.

Text Display

If we want to put a text on the screen, we need to know the pattern of characters.

- Based on the character pattern, pixel row, and column information, we decide on RGB values to be sent to the VGA_Sync component.
- The following lines of code can put **H** on the screen:

```
if (((8<row<18) and (col = 8)) or ((8<row<18) and (col = 13))
  or ((row=13) and (8<col<13))) then
    red <= '1';
else
    red <= '0';
end if;
```

We can store the display pattern of characters in a memory and access the memory for writing text on the screen.

Text Display

A group of characters are stored in a memory block in the FPGA.

- This memory is instantiated in the ***char_rom.vhd***
- The memory should be initialized with the information of character patterns.
  - A *.mif file is used to initialize the memory.
  - TCGROM.mif is the memory initialization file that contains the patterns of 64 characters.
  - Each character in a .mif file is described through 8 lines of memory address and is translated to a block of 8x8 pixels.

Address	Font Data
000001000 :	00011000 ;
000001001 :	00111100 ;
000001010 :	01100110 ;
000001011 :	01111110 ;
000001100 :	01100110 ;
000001101 :	01100110 ;
000001110 :	01100110 ;
000001111 :	00000000 ;

8 x 8 Font Pixel Data

0	0	0	1	1	0	0	0
0	0	1	1	1	1	0	0
0	1	1	0	0	1	1	0
0	1	1	1	1	1	1	0
0	1	1	0	0	1	1	0
0	1	1	0	0	1	1	0
0	1	1	0	0	1	1	0
0	0	0	0	0	0	0	0

Text Display

char_rom.vhd gets an instance of ***altsyncram*** component which is a memory IP core.

```
ENTITY char_rom IS
    PORT
    (
        character_address   :   IN STD_LOGIC_VECTOR (5 DOWNTO 0);
        font_row, font_col  :   IN STD_LOGIC_VECTOR (2 DOWNTO 0);
        clock               :   IN STD_LOGIC ;
        rom_mux_output      :   OUT STD_LOGIC
    );
END char_rom;

ARCHITECTURE SYN OF char_rom IS

    SIGNAL rom_data     : STD_LOGIC_VECTOR (7 DOWNTO 0);
    SIGNAL rom_address  : STD_LOGIC_VECTOR (8 DOWNTO 0);

    COMPONENT altsyncram
    GENERIC (
        address_aclr_a          : STRING;
        clock_enable_input_a    : STRING;
        clock_enable_output_a   : STRING;
        init_file               : STRING;
        intended_device_family  : STRING;
        lpm_hint                : STRING;
        lpm_type                : STRING;
        numwords_a              : NATURAL;
        operation_mode          : STRING;
        outdata_aclr_a          : STRING;
        outdata_reg_a           : STRING;
        widthad_a               : NATURAL;
        width_a                 : NATURAL;
        width_byteena_a         : NATURAL
    );
    PORT (
        clock0      : IN STD_LOGIC ;
        address_a   : IN STD_LOGIC_VECTOR (8 DOWNTO 0);
        q_a         : OUT STD_LOGIC_VECTOR (7 DOWNTO 0)
    );
    END COMPONENT;
```


Text Display

We only need to provide ***rom_address*** and extract one bit of ***rom_data*** as an output for each pixel.

```
BEGIN

    altsyncram_component : altsyncram
    GENERIC MAP (
        address_aclr_a => "NONE",
        clock_enable_input_a => "BYPASS",
        clock_enable_output_a => "BYPASS",
        init_file => "tcgrom.mif",
        intended_device_family => "Cyclone III",
        lpm_hint => "ENABLE_RUNTIME_MOD=NO",
        lpm_type => "altsyncram",
        numwords_a => 512,
        operation_mode => "ROM",
        outdata_aclr_a => "NONE",
        outdata_reg_a => "UNREGISTERED",
        widthad_a => 9,
        width_a => 8,
        width_byteena_a => 1
    )
    PORT MAP (
        clock0 => clock,
        address_a => rom_address,
        q_a => rom_data
    );

    rom_address <= character_address & font_row;
    rom_mux_output <= rom_data (CONV_INTEGER(NOT font_col(2 DOWNTO 0)));

END SYN;
```


Text Display

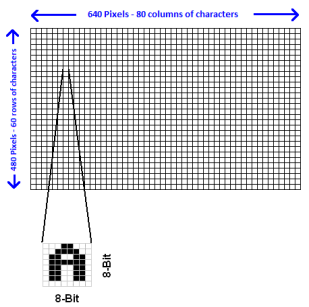
The following table shows the contents of the CharROM which is initialized through TCGROM.mif file.

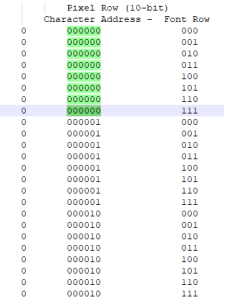
- Memory depth is **512**.
- Memory width is **8**. The content of memory for each address is an 8-bit value.
- Notice that the address is in **Oct** format.

CHAR	ADDRESS	CHAR	ADDRESS	CHAR	ADDRESS	CHAR	ADDRESS
@	00	P	20	Space	40	0	60
A	01	Q	21	!	41	1	61
B	02	R	22	"	42	2	62
C	03	S	23	#	43	3	63
D	04	T	24	$	44	4	64
E	05	U	25	%	45	5	65
F	06	V	26	&	46	6	66
G	07	W	27	'	47	7	67
H	10	X	30	(	50	8	70
I	11	Y	31	)	51	9	71
J	12	Z	32	*	52	A	72
K	13	[	33	+	53	B	73
L	14	Dn Arrow	34	,	54	C	74
M	15	]	35	-	55	D	75
N	16	Up Arrow	36	.	56	E	76
O	17	Lft Arrow	37	/	57	F	77

Text Display

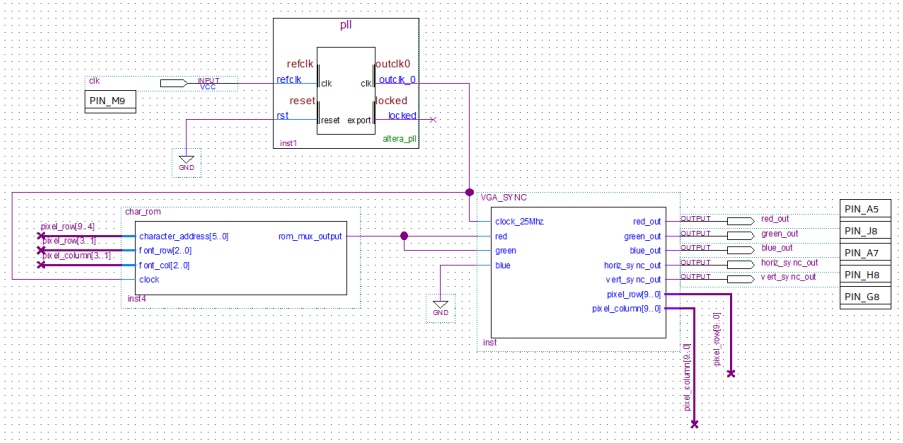
- The way we use part of pixel-row and pixel-column value as the address to the CharROM defines the size of the text.
  - If we use 3 lower bits of the pixel-row address, we will get the text in its original size of 8x8.
- To make characters larger, each dot in the font should map to several pixels.
  - To double the size, each dot should map to a 2x2 pixel block.
  - pixel-row[3 downto 1] and pixel-column[3 downto 1] are used as the font row and font column.





Text Display Example

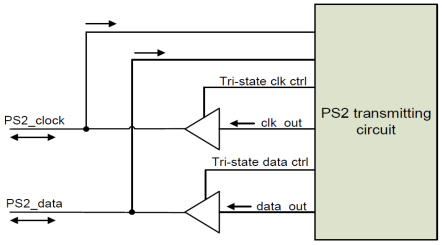
Try this example and see how you can fill the screen with rows of different characters:



PS/2 Mouse Interface

The PS/2 device interface was originally developed by IBM and is used by many mice and keyboards.

- The PS/2 mouse and keyboard implement a **bidirectional synchronous** serial protocol.
- The interface consists of two bidirectional signals:
  - **Clock**
  - **Data**



PS/2 Mouse Interface

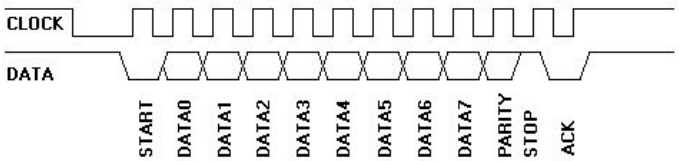
There are three different bus states:

- **Idle**
  - Both *Data* and *Clock* lines are kept **high**.
  - This is the only state where the mouse is allowed to begin transmitting data.
- **Inhibit**
  - The host may inhibit communication at any time by pulling the *Clock* line low.
  - The host then pulls *Data* low and releases *Clock* to enter *Host Request-to-Send*
- **Host Request-to-Send**
  - This state signals the device to start generating clock pulses for the host to device communication.

The clock signal is always generated by the mouse, unless the host wants to inhibit mouse transmission in order to send a command.

PS/2 Mouse Interface

- Data is transmitted **one byte** at a time. Each byte is sent in a sequence consisting of 11-12 bits. These bits are:
  - **1 start bit**: Always '0'
  - **8 data bits**: Least significant bit first.
  - **1 parity bit**: Odd parity
  - **1 stop bit**: Always '1'
  - **1 acknowledge bit**: Host-to-device communication only

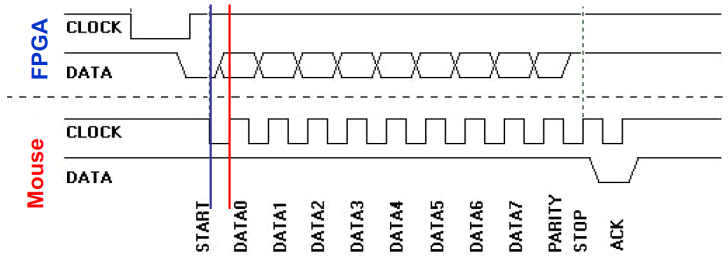


Host (DE0-CV board) drives the data line when **sending commands** to the mouse.

Mouse drives the data line when **sending data** to the DE0-CV board.

PS/2 Mouse Interface - Host to Device Communication

- The clock signal is always generated by the mouse unless in *Inhibit* state.
- Data is shifted out by the host at clock **falling edge** and is read by the mouse at clock **rising edge**.
- An acknowledgment bit is sent from the mouse to the host.



The diagram shows the separated clock and data signals generated by host and mouse side which can be superimposed.

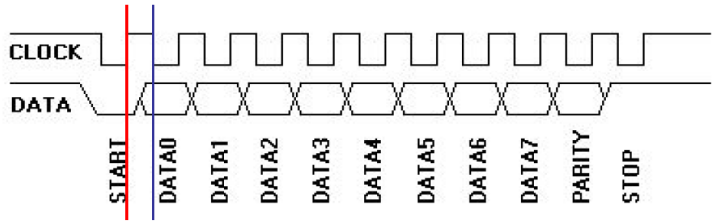
PS/2 Mouse Interface - Host to Device Communication

The steps that host must follow to send data to a PS/2 device:

1. Bring the Clock line low for at least 100 microseconds.
2. Bring the Data line low.
3. Release the Clock line.
4. Wait for the mouse to bring the Clock line low.
5. Set/reset the Data line to send the first data bit.
6. Wait for the mouse to bring Clock high.
7. Wait for the mouse to bring Clock low.
8. Repeat steps 5-7 for the other seven data bits and the parity bit.
9. Release the Data line.
10. Wait for the mouse to bring Data low.
11. Wait for the mouse to bring Clock low.
12. Wait for the mouse to release Data and Clock.

PS/2 Mouse Interface - Device to Host Communication

- The clock signal is generated by the mouse.
- The data bit is written by the mouse at clock **rising edge** and is read by the host at clock **falling edge**.
- No acknowledgment bit is required.



PS/2 Mouse Interface

Mouse Modes

- **Reset Mode**
  - Initialization and self test is done.
- **Stream Mode**
  - Mouse transmits three data packets.
  - Host must transmit **0xF4** command to the mouse to initiate data reporting.
- **Remote Mode**
  - Host requests movement data packets.
- **Wrap Mode**
  - This is diagnostic mode.

PS/2 Mouse Interface - Initialization

- At power up, the mouse will go to **Reset Mode**.
  - It runs a self test and sends the message **0xAA**, which indicates that the test is passed.
  - It then sends **0x00** which is the ID for standard PS/2 mouse.
  - If the PS/2 mouse is connected before the FPGA device is programmed this data can be ignored in the interface design.
- In **Stream Mode**, the streaming should be enabled by a command from host.
  - Command **0xF4** is sent by the host.
  - Acknowledge is sent by the mouse.
  - Mouse continuously transmits three data packets which include the movement information as well as the pushbutton status.
  - Mouse sends data at a default sampling rate of 100 samples per second.

PS/2 Mouse Interface - Data Packet Format

Bits	MSB (7)	6	5	4	3	2	1	LSB (0)
Byte 1	Yo	Xo	Ys	Xs	1	0	R	L
Byte 2	X7	X6	X5	X4	X3	X2	X1	X0
Byte 3	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0

- **X7-X0**: Moving distance of X-axis (horizontal move).
  - moving right = positive values
  - moving left = negative values
- **Y7-Y0**: Moving distance of Y-axis (vertical move).
  - moving up = positive values
  - moving down = negative values
- **Xo**: X data overflow bit (1 = overflow)
- **Yo**: Y data overflow bit (1 = overflow)
- **Xs**: X data sign bit (1 = negative)
- **Ys**: Y data sign bit (1 = negative)
- **L**: Left button status bit (1 = button pressed, 0 = released)
- **R**: Right button status bit (1 = button pressed, 0 = released)

PS/2 Mouse Interface

- Byte 1 of the packet contains button status information and X, Y motion sign, and overflow bits.
  - 8-bit distance value together with the sign bit covers the range of -256 to +255.
  - If the movement is too fast the overflow bits are set.
- Byte 2 and 3 contain the value of relative X, Y movement.
  - These values should be added to the current cursor position.
  - Two 10-bit registers are used to store the cursor position within the screen range of 640x480.
  - The registers should be updated once a new packet of data is received.
- The cursor is usually initialized to the center of a video screen at power up.

PS/2 Mouse Interface - Mouse Component

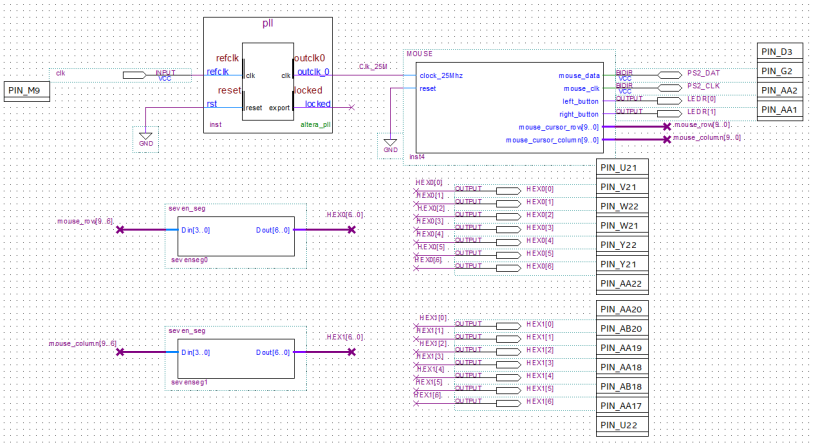
```
9   ENTITY MOUSE IS
10      PORT( clock_25Mhz, reset             : IN std_logic;
11            SIGNAL mouse_data              : INOUT std_logic;
12            SIGNAL mouse_clk               : INOUT std_logic;
13            SIGNAL left_button, right_button: OUT std_logic;
14            SIGNAL mouse_cursor_row        : OUT std_logic_vector(9 DOWNTO 0);
15            SIGNAL mouse_cursor_column     : OUT std_logic_vector(9 DOWNTO 0));
16  END MOUSE;
```

```
46  mouse_cursor_row <= cursor_row;
47  mouse_cursor_column <= cursor_column;
48
49              -- tri_state control logic for mouse data and clock lines
50  MOUSE_DATA <= 'Z' WHEN MOUSE_DATA_DIR = '0' ELSE MOUSE_DATA_BUF;
51  MOUSE_CLK <=  'Z' WHEN MOUSE_CLK_DIR = '0' ELSE MOUSE_CLK_BUF;
```

```
105     WITH mouse_state SELECT
106 -- Mouse Data Tri-state control line: '1' DE0 drives, '0'=Mouse Drives
107         MOUSE_DATA_DIR  <=  '0' WHEN INHIBIT_TRANS,
108                             '0' WHEN LOAD_COMMAND,
109                             '0' WHEN LOAD_COMMAND2,
110                             '1' WHEN WAIT_OUTPUT_READY,
111                             '0' WHEN WAIT_CMD_ACK,
112                             '0' WHEN INPUT_PACKETS;
113 -- Mouse Clock Tri-state control line: '1' DE0 drives, '0'=Mouse Drives
114     WITH mouse_state SELECT
115         MOUSE_CLK_DIR   <=  '1' WHEN INHIBIT_TRANS,
116                             '1' WHEN LOAD_COMMAND,
117                             '1' WHEN LOAD_COMMAND2,
118                             '0' WHEN WAIT_OUTPUT_READY,
119                             '0' WHEN WAIT_CMD_ACK,
120                             '0' WHEN INPUT_PACKETS;
```

PS/2 Mouse Interface Example

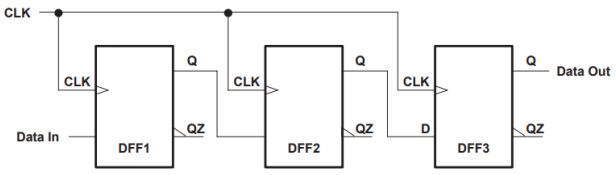
Try this simple example and see how you can see the row and column values of your cursor on seven segment displays.



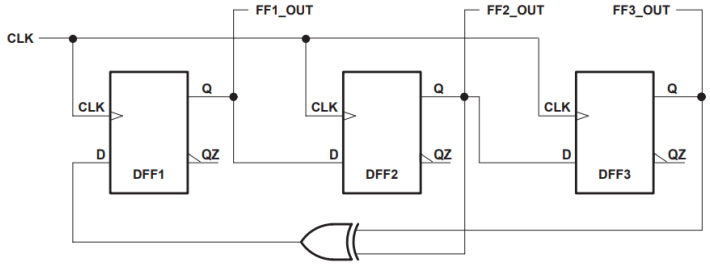
Linear Feedback Shift Register (LFSR)

- A linear feedback shift register **(LFSR)** is a shift register whose input bit is the output of a linear function of two or more bits of its previous states.
- The linear feedback can be formed by performing exclusive-OR on the outputs of two or more of the flip flops together.
  - Alternatively XNOR can be used for the feedback.
- LFSRs can be used in variety of applications such as
  - Pseudo-random number generators
  - Test pattern generation
  - Cyclic Redundancy Check (CRC)
  - Cryptography

Linear Feedback Shift Register (LFSR)



A 3-Bit Shift Register



Linear Feedback Shift Register

Linear Feedback Shift Register (LFSR)

- The points within the register chain, where the feedback comes from are called **taps**.
  - Taps are the bits that influence the output.
  - Two LFSRs with the same seed but different taps generate different sequences.
- The initial value of the LFSR is called the **seed**.
  - It should be a **non-zero** value, otherwise LFSR would be stuck at the seed value.
- An LFSR is of maximal length if it sequence through every possible value.
  - A maximal length **n-bit** LFSR can sequence through $2^n - 1$ values.
  - The state "0000..." (all zeros) is not included in the sequence.

Linear Feedback Shift Register (LFSR)

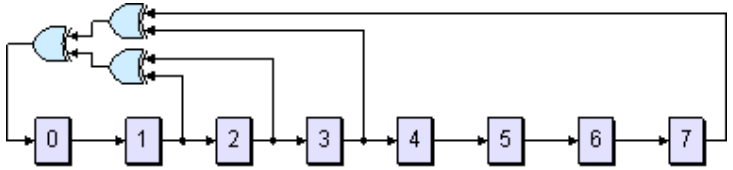
- The choice of taps determines how many values there are in a given sequence before the sequence is repeated.
- Some tap choices for maximal length sequence is provided:

Number of bits	Length of loop	Taps
2	3	0,1
3	7	0,2
4	15	0,3
5	31	1,4
6	63	0,5
7	127	0,6
8	255	1,2,3,7
9	511	3,8
10	1023	2,9
11	2047	1,10

Linear Feedback Shift Register (LFSR)

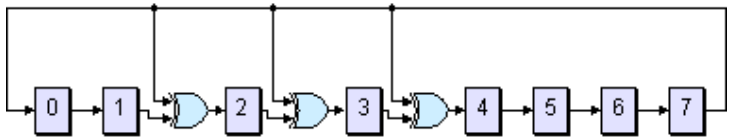
There are two types of LFSRs, depending on how feedback is formed:

- In **Fibonacci LFSR**, the XOR gates (taps), are placed on the feedback path.
- Increasing the levels of logic in the combinational feedback path can **negatively impact** the maximum clocking frequency of the function.



Linear Feedback Shift Register (LFSR)

- In **Galois LFSR**, the XOR gates (taps), are placed between the registers.
- Galois type is more recommended in this project.



Summary

- We looked at VGA interface and discussed how to show text on the VGA screen through several examples.
- We looked at the PS/2 mouse interface and discussed about its communication protocol.
  - Host to device communication for sending the commands.
  - Device to host communication for sending the data.
- We introduced LFSR to be used as a pseudo-random number generator.
  - LFSR can be used in your mini project to generate random values for the gaps in the pipes.

Acknowledgment

- Some figures/notes are taken from or inspired by the
  - CS305 Lecture notes by Muhammad Nadeem, 2019
  - EETimes Tutorial : Linear Feedback Shift Registers by Max Maxfield, 2006
  - PS/2 Mouse/Keyboard Protocol by Adam Chapweske, 1999