CSCE 313: Embedded Systems

Final Project Notes

Instructor: Jason D. Bakos

UNIVERSITY OF SOUTH CAROLINA

Final Project

- Final project: report AND demonstration due by April 29
- Each group must make appointment for 20 minute demonstration of their final project
  - Both members must attend demo
  - Otherwise, 15% penalty
- Objective:
  - Begin with Lab 5
  - Convert all floating-point computations to fixed-point
  - Increase resolution to 1024 x 768

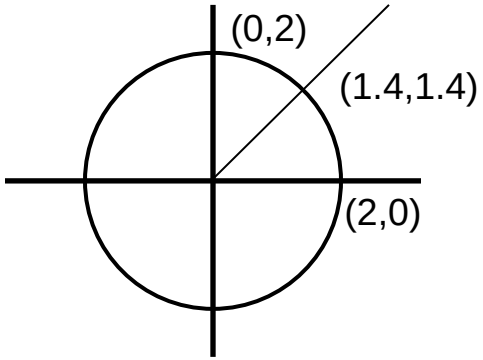
Fixed-Point

- Use a fixed-point representation for:
  - z
  - c
  - min_x, max_x, min_y, max_y
  - x^2+y^2 (for checking for divergence)

Fixed-Point Review

- Recall: fixed-point has fixed range
  - Recall: Range of non-fixed-point n-bit integer:
    - $-2^{n-1} <= val <= 2^{n-1}-1$
  - Range of signed (n,m) value:
    - $-2^{n-m-1} <= val <= 2^{n-m-1} - 2^{-m}$
  - Need to decide where to set decimal point
    - For c, [min|max]_[x|y]:
      - Need to represent values from -2 to 2
      - Use (32,29) representation for [-4,4) (to include +2)
    - z should cover the worst case for a diverged pixel
    - x^2+y^2 should cover the worst case for the r^2 of a diverged pixel

Worst Case Analysis



z	c	P(z) $(z_x^2-z_y^2+c_x, 2z_xz_y+c_y)$	LH bits	x^2+y^2	LH bits
(0,2)	(0,2)	(-4,2)	4	20	6
(0,2)	(2,0)	(-2,0)	3	4	4
(2,0)	(0,2)	(4,2)	4	20	6
(2,0)	(2,0)	(6,0)	4	36	7
(1.4,1.4)	(1.4,1.4)	(1.4,5.3)	4	30	6

Range and Precision

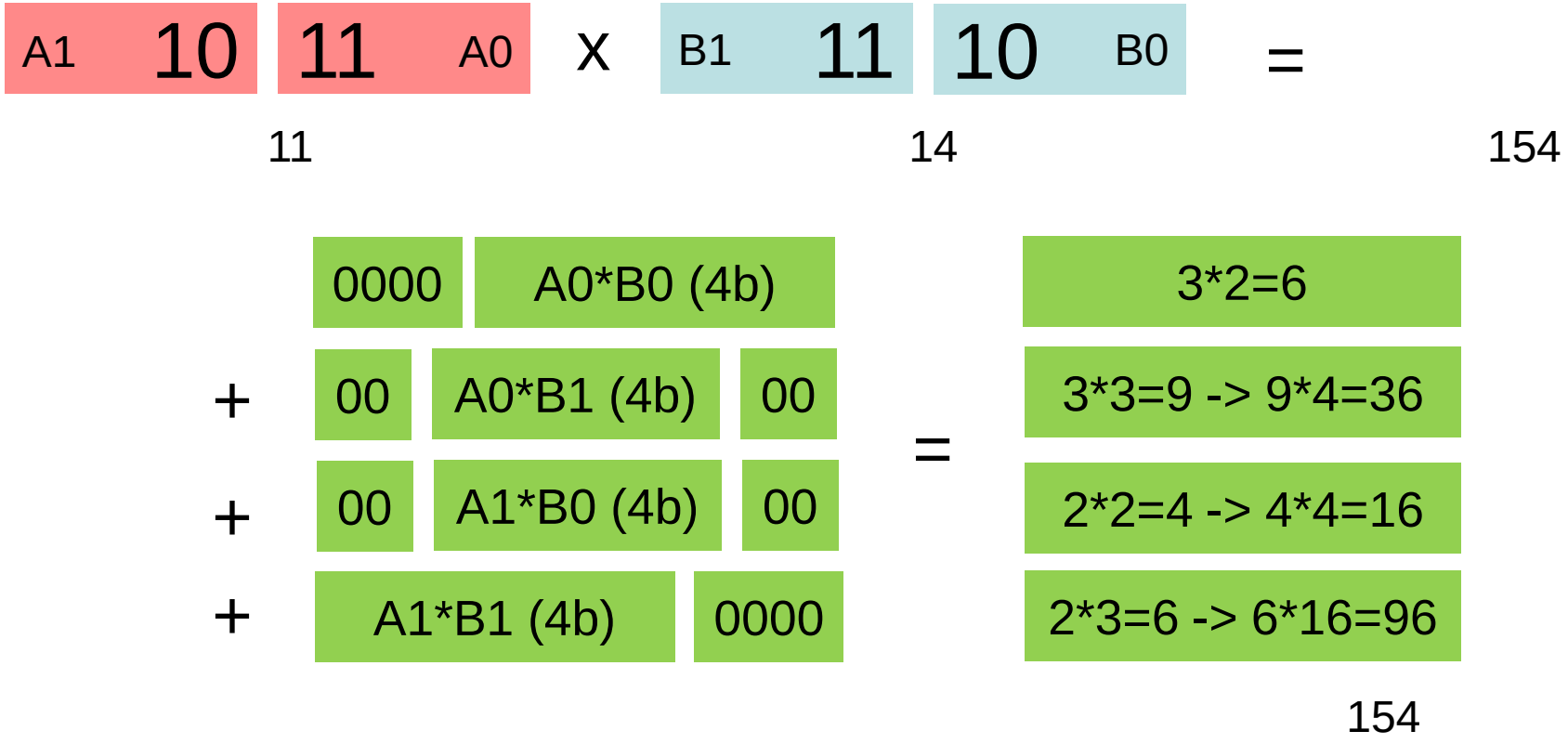
- For c, need 3 bits left of decimal: (32,29)
- For z, need 4 bits left of decimal: (32,28)
- For r^2, need 7 bits the left of decimal: (32,25)
- For row and col, need 0 to the right of the decimal: (32,0)

- Problem:
  - Multiply (32,n) val with (32,m) val => (64,n+m) val

- Need a way to get 64-bit product, or we lose the upper 32 bits

Multiply

- Assume 4-bit registers, need 8 bit product:



Example Fixed Point Multiply

- Multiply A = (32,n) val and B = (32,m) val, need C = (32,o) product:
  - Declare A and B as "long"
  - Declare C as "long long"
  - Cast A and B as "long long",
  - C = (long long)A * (long long)B;
  - Convert C from (64,n+m) to (32,o):
    - Shift C (n+m)-o bits to the right
  - Return C as int

High Resolution

- Goal: Increase resolution from 320x240 to 1024x768

- Problems:
  - Native resolution of VGA Controller is 640x480, so hardware modification is needed
  - SRAM is only 512MB, not large enough to store a higher resolution frame, so we need to move pixel buffer to SDRAM

Hardware Modification

- Remove SRAM interface, connect DMA controller master interface directly to SDRAM
- Remove rescaler (no longer needed)
- Change DMA controller settings to 1024x768
- Each time you generate in SOPC Builder, must make edits to two generated Verilog files

Hardware Modification

- VGA Controller Verilog file, line 78 (after each generation):

```
parameter CW                    = 9;
parameter DW                    = 29;

parameter R_UI                  = 29;
parameter R_LI                  = 20;
parameter G_UI                  = 19;
parameter G_LI                  = 10;
parameter B_UI                  = 9;
parameter B_LI                  = 0;

/* Number of pixels */
parameter H_ACTIVE              = 640;
parameter H_FRONT_PORCH         =  16;
parameter H_SYNC                =  96;
parameter H_BACK_PORCH          =  48;
parameter H_TOTAL               = 800;

/* Number of lines */
parameter V_ACTIVE              = 480;
parameter V_FRONT_PORCH         =  10;
parameter V_SYNC                =   2;
parameter V_BACK_PORCH          =  33;
parameter V_TOTAL               = 525;

parameter NUMBER_OF_BITS_FOR_LINES      = 10;
parameter LINE_COUNTER_INCREMENT        = 10'h001;

parameter NUMBER_OF_BITS_FOR_PIXELS     = 10;
parameter PIXEL_COUNTER_INCREMENT       = 10'h001;
```

```
parameter CW                    = 9;
parameter DW                    = 29;

parameter R_UI                  = 29;
parameter R_LI                  = 20;
parameter G_UI                  = 19;
parameter G_LI                  = 10;
parameter B_UI                  = 9;
parameter B_LI                  = 0;

/* Number of pixels */
parameter H_ACTIVE              = 1024;
parameter H_FRONT_PORCH         =  24;
parameter H_SYNC                =  136;
parameter H_BACK_PORCH          =  160;
parameter H_TOTAL               = 1344;

/* Number of lines */
parameter V_ACTIVE              = 768;
parameter V_FRONT_PORCH         =  3;
parameter V_SYNC                =   6;
parameter V_BACK_PORCH          =  29;
parameter V_TOTAL               = 806;

parameter NUMBER_OF_BITS_FOR_LINES      = 11;
parameter LINE_COUNTER_INCREMENT        = 11'h001;
parameter NUMBER_OF_BITS_FOR_PIXELS     = 11;
parameter PIXEL_COUNTER_INCREMENT       = 11'h001;
```

Hardware Modification

- Clocks Verilog file, line 69 (after each generation):

```
parameter SYS_CLK_MULT          = 1;
```

- Change to:

```
parameter SYS_CLK_MULT          = 2;
```

- Line 174:

```
DE_Clock_Generator_System.clk2_divide_by                = 2,
DE_Clock_Generator_System.clk2_duty_cycle               = 50,
DE_Clock_Generator_System.clk2_multiply_by              = 1,
```

- Change to:

```
DE_Clock_Generator_System.clk2_divide_by                = 7,
DE_Clock_Generator_System.clk2_duty_cycle               = 50,
DE_Clock_Generator_System.clk2_multiply_by              = 9,
```

Software Modification

- New pixel buffer requires 1024x748x2 = 1572864 bytes (1.5MB)
  - Should be OK with 2MB/CPU, depending on code size

- Allocate in SDRAM (as a global variable):

  volatile alt_u16 pixel_buffer_memory[786432]; // 768x1024

- Copy address of only one processor's buffer to the DMA controller for F/B buffer (in main()):

```
alt_up_pixel_buffer_dma_change_back_buffer_address(pixel_buffer,(unsigned
int)pixel_buffer_memory);
alt_up_pixel_buffer_dma_swap_buffers(pixel_buffer);
while (alt_up_pixel_buffer_dma_check_swap_buffers_status(pixel_buffer));
alt_up_pixel_buffer_dma_change_back_buffer_address(pixel_buffer,(unsigned
int)pixel_buffer_memory);
```

Software Modifications

- Subsequent writes to buffer using address stored in DMA controller will write the same buffer:

  IOWR_16DIRECT(*(((alt_u32 *)VIDEO_PIXEL_BUFFER_DMA_0_BASE+1)),offset<<1,pixel_color);

  or

  alt_up_pixel_buffer_dma_draw(**pixel_buffer,color,col,row);**

- This is because pointer is stored in global location

- Note: Don't use divide! Add a fixed-point constant for (1/768) and (1/1024) and multiply these