```c
#include "address_map_arm.h"
#include <stdbool.h>
#include <stdlib.h>
#include <stdio.h>
#include <time.h>

volatile int pixel_buffer_start; // global variable

void plot_pixel(int x, int y, short int line_color)
{
    *(short int *)(pixel_buffer_start + (y << 10) + (x << 1)) = line_color;
}

// Swaps 2 numbers using the XOR operation
void swap(int * x, int * y)
{
    int temp = *x;
    *x = *y;
    *y = temp;
}

void draw_line(int x1, int y1, int x2, int y2, short int colour)
{
    // Check steepness of the line, if it is steep, it's better
    // to move along the y-axis when drawing
    bool is_steep = abs(y2-y1) > abs(x2-x1);
    // If it is steep switch the x and y values
    // the drawing loop will decide how the drawing will occur
    if(is_steep) {
        swap(&x1,&y1);
        swap(&x2,&y2);
    }

    // We are going to increment from x1 to x2 so
    // swap the endpoints if x1 > x2
    if(x1 > x2) {
        swap(&x1,&x2);
        swap(&y1,&y2);
    }

    int deltax = x2-x1;
    int deltay = abs(y2-y1);
    int error = -(deltax/2);
    int x,y,y_step;

    // Figure out how y will be incremented
    if(y1<y2) y_step = 1;
    else y_step = -1;

    for(x=x1,y=y1; x<=x2; x++) {
        // If the line is steep the x and y values are swapped
        if(is_steep) plot_pixel(y,x,colour);
        else plot_pixel(x,y,colour);

        // Check margin of error
        error += deltay;
        if(error>=0) {
            y += y_step; // Increment y val
            error -= deltax; // Reset error
        }
    }
}

// Fill a rectangle with a chosen colour
void fill_rect(int x, int y, int width, int height, short int colour)
{
    int dx,dy;
    for(dx=0; dx<width; dx++) {
        for(dy=0; dy<height; dy++) {
            plot_pixel(x+dx,y+dy,colour);
        }
    }
}

// Draw black to every pixel on the screen
void clear_screen()
{
    int x,y;
    // The screen is 320x240
    for(x=0; x<320; x++) {
        for(y=0; y<240; y++) {
            plot_pixel(x,y,0x0000);
        }
    }
}

// Synchronizes the display with the VGA timing
void wait_for_vsync()
{
    volatile int * pixel_ctrl_ptr = (int *)PIXEL_BUF_CTRL_BASE;
    register int status;

    *pixel_ctrl_ptr = 1; // Start synchronization process

    // Keep waiting until the whole screen ahs been drawn
    do {
        status = *(pixel_ctrl_ptr + 3);
    } while((status & 0x01) != 0);
}

int main(void)
{
    volatile int * pixel_ctrl_ptr = (int *)PIXEL_BUF_CTRL_BASE;

    srand(time(NULL)); // Set up for random number generation

    short int colourBank[8] = {0x001F,0x07E0,0xF800,0xF81F,0x07FF,0xF81F,0xFFE,0xFFFF};
    int numRects = 8; // Have 8 rectangles
    int width[8],height[8],colour[8],x[8],y[8],x_step[8],y_step[8];

    // Set up the rectangles, they will all 2x2 in size
    int i;
    for(i=0; i<numRects; i++) {
        width[i] = 2;
        height[i] = 2;
        colour[i] = colourBank[rand()%8];
        // Avoid spawning the rectangle out of bounds
        x[i] = rand()%(320-width[i]);
        y[i] = rand()%(240-height[i]);
        // Set initial direction
        x_step[i] = rand()%2 * 2 - 1; // +/- 1
        y_step[i] = rand()%2 * 2 - 1; // +/- 1
    }

    /* set front pixel buffer to start of FPGA On-chip memory */
    *(pixel_ctrl_ptr + 1) = FPGA_ONCHIP_BASE; // first store the address in the
                                        // back buffer
    /* now, swap the front/back buffers, to set the front buffer location */
    wait_for_vsync();
    /* initialize a pointer to the pixel buffer, used by drawing functions */
    pixel_buffer_start = *pixel_ctrl_ptr;
    clear_screen(); // pixel_buffer_start points to the pixel buffer
    /* set back pixel buffer to start of SDRAM memory */
    *(pixel_ctrl_ptr + 1) = DDR_BASE;
    pixel_buffer_start = *(pixel_ctrl_ptr + 1); // we draw on the back buffer

    while (1)
    {
        // Erase any boxes and lines that were drawn in the last iteration
        clear_screen();

        // Draw boxes and lines and update locations
        for(i=0; i<numRects; i++) {
            // Draw the line connecting boxes
            if(i==numRects-1) { // Wrap around
                draw_line(x[i],y[i],x[0],y[0],colour[i]);
            } else {
                draw_line(x[i],y[i],x[i+1],y[i+1],colour[i]);
            }

            // Draw the box
            fill_rect(x[i],y[i],width[i],height[i],colour[i]);

            // Update location
            x[i] += x_step[i];
            y[i] += y_step[i];

            // Horizontal bounce
            if(x[i]+width[i]==319) x_step[i] = -1;
            else if(x[i]==0) x_step[i] = 1;

            // Vertical bounce
            if(y[i]+height[i]==239) y_step[i] = -1;
            else if(y[i]==0) y_step[i] = 1;
        }

        wait_for_vsync(); // swap front and back buffers on VGA vertical sync
        pixel_buffer_start = *(pixel_ctrl_ptr + 1); // new back buffer
    }

    return 0;
}
```