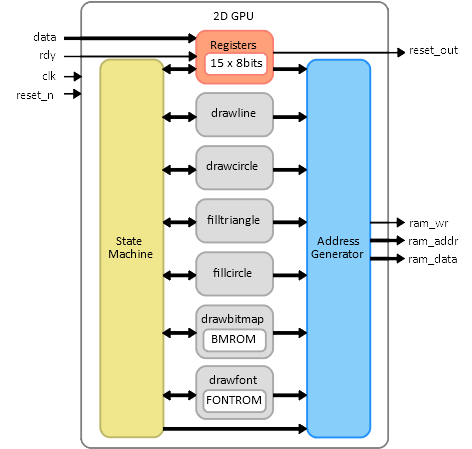
**Gnarly Grey 2D GPU IP**

**7/25/18**

The Gnarly Grey 2D GPU IP supports acceleration functions for drawing line, rectangle, circle, triangle and filling up rectangle, circle and triangle inside a very small Lattice Ultraplus 5K LUT FPGA. There’s also support for mono-color 64x64 pixel bitmap and fonts with variable size. The GPU can be controlled with very simple interface like UART to receive its commands or register configurations with very low bandwidth. Rendered graphics from these settings are passed to a RAM/Display Buffer. Current implementation of supports 12-bit (4096) colors (4-bits for R, G, B each).

IP Block Diagram

The GPU IP has 15 registers to store information related to GPU task such as graphic coordinates, GPU functions to be execute, color bitmap number, font character and size, etc. Information stored in registers is utilized by a state machine, address generator and acceleration sub modules (drawline, drawcircle, filltriangle, etc.) to perform the given acceleration task. A state machine enables acceleration sub modules based on information configured in the register set and the address generator takes data from acceleration sub module to write rendering information on display buffer.

The GPU IP is designed to take register address and data information from the same input bus. It takes the first data received on the bus as the address and second data received as the data.

There’s facility to connect multiple GPU (up to 15) on same bus and controlling all GPU (Display) simultaneously or individually by using broadcast or individual control/status registers as mentioned in Register Map section. A display number for particular instance can be set by a Verilog parameter at synthesis time. There’s also a Verilog parameter for rotation, in case other than 0 degree rotation is required.

Port and Parameter Details:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Size (bits) | Description |
| ROTATION | Parameter | 2 | 2’b00 – 0 degree rotation  2’b01 – 90 degree rotation  2’b10 – 180 degree rotation  2’b11 – 270 degree rotation |
| DISPLAY\_NO | Parameter | 4 | Value from 1 (4’h0001) to 15 (4’h1111) |
| clk | Input | 1 | Clock |
| reset\_n | Input | 1 | Reset – Active Low |
| rdy | Input | 1 | Ready – Active High |
| data | Input | 8 | Data – used for register address and data |
| reset\_out | Output | 1 | Reset Out – Active High |
| ram\_wr | Output | 1 | Memory Write Enable – Active High |
| ram\_addr | Output | 20 | Memory Write Address |
| ram\_data | Output | 8 | Memory Write Data |

Register Map

Following is description of registers available on IP, all registers take 00h as default value upon reset.

Register Address 00h – Control/Status Register [Boardcast]

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| START | BUSY | RESET | NU | FUN[3] | FUN[2] | FUN[1] | FUN[0] |

|  |  |  |
| --- | --- | --- |
|  | FUN[3:0]  0000  0001  0010  0011  0100  0101  0110  0111  1000  1001 | Draw Function  pixel  line  rectangle  triangle  circle  fill rectangle  fill triangle  fill circle  bitmap  font |
|  | NU | Not Used |
|  | RESET  0  1 | Reset GPU and generate reset\_out  No effect  Reset, self clearing |
|  | BUSY  0  1 | GPU busy status  GPU is free to accept new request  GPU is busy |
|  | START  0  1 | Trigger/Start execution of acceleration function  No effect  Start, self clearing |

Register Address 01h – Red Color Register

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| RED[3] | RED[2] | RED[1] | RED[0] | NU | NU | NU | NU |

|  |  |  |
| --- | --- | --- |
|  | NU | Not Used |
|  | RED[3:0] | Red Color Value (0 to 15) |

Register Address 02h – Green Color Register

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| GREEN[3] | GREEN[2] | GREEN[1] | GREEN[0] | NU | NU | NU | NU |

|  |  |  |
| --- | --- | --- |
|  | NU | Not Used |
|  | GREEN[3:0] | Green Color Value (0 to 15) |

Register Address 03h – Blue Color Register

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| BLUE[3] | BLUE[2] | BLUE[1] | BLUE[0] | NU | NU | NU | NU |

|  |  |  |
| --- | --- | --- |
|  | NU | Not Used |
|  | BLUE[3:0] | Blue Color Value (0 to 15) |

Register Address 04h – Point 1, X Co-ordinate Register

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| X1[7] | X1[6] | X1[5] | X1[4] | X1[3] | X1[2] | X1[1] | X1[0] |

|  |  |  |
| --- | --- | --- |
|  | X1[7:0] | Point 1, X Co-ordinate Value (0 to 255) |

Register Address 05h – Point 1, Y Co-ordinate Register

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Y1[7] | Y1[6] | Y1[5] | Y1[4] | Y1[3] | Y1[2] | Y1[1] | Y1[0] |

|  |  |  |
| --- | --- | --- |
|  | Y1[7:0] | Point 1, Y Co-ordinate Value (0 to 255) |

Register Address 06h – Point 2, X Co-ordinate Register

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| X2[7] | X2[6] | X2[5] | X2[4] | X2[3] | X2[2] | X2[1] | X2[0] |

|  |  |  |
| --- | --- | --- |
|  | X2[7:0] | Point 2, X Co-ordinate Value (0 to 255) |

Register Address 07h – Point 2, Y Co-ordinate Register

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Y2[7] | Y2[6] | Y2[5] | Y2[4] | Y2[3] | Y2[2] | Y2[1] | Y2[0] |

|  |  |  |
| --- | --- | --- |
|  | Y2[7:0] | Point 2, Y Co-ordinate Value (0 to 255) |

Register Address 08h – Point 3, X Co-ordinate Register

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| X3[7] | X3[6] | X3[5] | X3[4] | X3[3] | X3[2] | X3[1] | X3[0] |

|  |  |  |
| --- | --- | --- |
|  | X3[7:0] | Point 3, X Co-ordinate Value (0 to 255) |

Register Address 09h – Point 3, Y Co-ordinate Register

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Y3[7] | Y3[6] | Y3[5] | Y3[4] | Y3[3] | Y3[2] | Y3[1] | Y3[0] |

|  |  |  |
| --- | --- | --- |
|  | Y3[7:0] | Point 3, Y Co-ordinate Value (0 to 255) |

Register Address 0Ah – Circle Radius Register

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| RAD[7] | RAD[6] | RAD[5] | RAD[4] | RAD[3] | RAD[2] | RAD[1] | RAD[0] |

|  |  |  |
| --- | --- | --- |
|  | RAD[7:0] | Circle Radius Value (0 to 255) |

Register Address 0Bh – Bitmap Number Register

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| NU | NU | NU | BMN[4] | BMN[3] | BMN[2] | BMN[1] | BMN[0] |

|  |  |  |
| --- | --- | --- |
|  | BMN[4:0] | Bitmap Number Value (0 to 23) |
|  | NU | Not Used |

Register Address 0Ch – Font Character Register

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| FC[7] | FC[6] | FC[5] | FC[4] | FC[3] | FC[2] | FC[1] | FC[0] |

|  |  |  |
| --- | --- | --- |
|  | FC[7:0] | Font Character Value (0 to 255) |

Register Address 0Dh – Font Size Register

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| NU | NU | NU | FS[4] | FS[3] | FS[2] | FS[1] | FS[0] |

|  |  |  |
| --- | --- | --- |
|  | FS[4:0] | Font Size Value (1 to 31) |
|  | NU | Not Used |

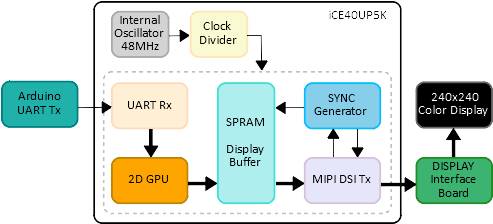
Register Address X0h – Control/Status Register [Individual], X= Display No (1 to 15)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| START | BUSY | RESET | NU | FUN[3] | FUN[2] | FUN[1] | FUN[0] |

|  |  |  |
| --- | --- | --- |
|  | FUN[3:0]  0000  0001  0010  0011  0100  0101  0110  0111  1000  1001 | Draw Function  pixel  line  rectangle  triangle  circle  fill rectangle  fill triangle  fill circle  bitmap  font |
|  | NU | Not Used |
|  | RESET  0  1 | Reset GPU and generate reset\_out  No effect  Reset, self clearing |
|  | BUSY  0  1 | GPU busy status  GPU is free to accept new request  GPU is busy |
|  | START  0  1 | Trigger/Start execution of acceleration function  No effect  Start, self clearing |

Typical System Example:

In this example, a typical system using 2D GPU is showcased. The system consist of a microcontroller such as an Arduino Nano ATMega328 with a UART interface connected to Upduino v1.0 or v2.0 board. A LH154Q01 Display Adapter for UPDuino boards is used to connected a 240x240 pixel MIPI display on one end to the Upduino v1.0 or v2.0 board.



In this typical setup, Arduino Nano microcontroller is the host controller which control’s operation of GPU. Inside the iCE40UP5K FPGA, small UART receiver is used to receive commands from Arduino over UART and provide it to the 2D GPU. The 2D GPU is connected with 128kB Display Buffer (SPRAM) available on FPGA itself. The display buffer is accessed by a MIPI DSI transmitter design to send display buffer data to the 240x240 pixel color display through MIPI display interface board. SYNC Generator controls timing of sending display buffer to MIPI display.

The following are some typical examples of the GPU commands issued by the microcontroller over UART:

1. Draw a line from p1(10,20) to p2(50,60) with color Red.

Serial.write(0x04); //Register address for P1-x

Serial.write(0x0A); //Register data for P1-x

Serial.write(0x05); //Register address for P1-y

Serial.write(0x14); //Register data for P1-y

Serial.write(0x06); //Register address for P2-x

Serial.write(0x32); //Register data for P2-x

Serial.write(0x07); //Register address for P2-y

Serial.write(0x3C); //Register data for P2-y

Serial.write(0x01); //Register address for Red color

Serial.write(0xFF); //Register data for Red color

Serial.write(0x02); //Register address for Green color

Serial.write(0x00); //Register data for Green color

Serial.write(0x03); //Register address for Blue Color

Serial.write(0x00); //Register data for Blue Color

Serial.write(0x00); //Register address for Control/Status Register

Serial.write(0x81); //Register data for Control/Status Register to draw line

2. Draw a Yellow color filled triangle with p1(10,20), p2(50, 60), p3(10, 60)

Serial.write(0x04); //Register address for P1-x

Serial.write(0x0A); //Register data for P1-x

Serial.write(0x05); //Register address for P1-y

Serial.write(0x14); //Register data for P1-y

Serial.write(0x06); //Register address for P2-x

Serial.write(0x32); //Register data for P2-x

Serial.write(0x07); //Register address for P2-y

Serial.write(0x3C); //Register data for P2-y

Serial.write(0x08); //Register address for P3-x

Serial.write(0x0A); //Register data for P3-x

Serial.write(0x09); //Register address for P3-y

Serial.write(0x3C); //Register data for P3-y

Serial.write(0x01); //Register address for Red color

Serial.write(0xFF); //Register data for Red color

Serial.write(0x02); //Register address for Green color

Serial.write(0xFF); //Register data for Green color

Serial.write(0x03); //Register address for Blue Color

Serial.write(0x00); //Register data for Blue Color

Serial.write(0x00); //Register address for Control/Status Register

Serial.write(0x86); //Register data for Control/Status Register to fill triangle

3. Draw a bitmap number 5 at p1(10,20) with SkyBlue color

Serial.write(0x04); //Register address for P1-x

Serial.write(0x0A); //Register data for P1-x

Serial.write(0x05); //Register address for P1-y

Serial.write(0x14); //Register data for P1-y

Serial.write(0x01); //Register address for Red color

Serial.write(0x00); //Register data for Red color

Serial.write(0x02); //Register address for Green color

Serial.write(0xFF); //Register data for Green color

Serial.write(0x03); //Register address for Blue Color

Serial.write(0xFF); //Register data for Blue Color

Serial.write(0x0B); //Register address for Bitmap Number

Serial.write(0x05); //Register data for Bitmap Number

Serial.write(0x00); //Register address for Control/Status Register

Serial.write(0x88); //Register data for Control/Status Register to draw bitmap