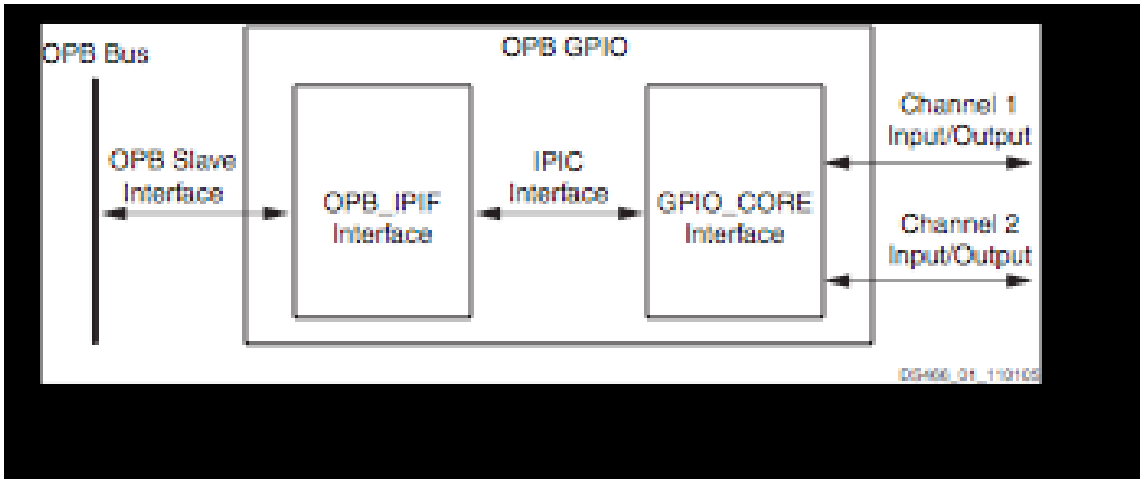
EECS 388
Computer Systems and Assembly Language
General Purpose I/O

GPIO_OPB

- GPIO := General Purpose Input/Output Core
  - Core provides all signals/connections to OPB bus
    - OPB On Chip Peripheral Bus
  - Can Have 1 or 2 Channels of 32 bits each
  - Each bit can be configured as input/output or tri-state



Schematic (Hardware Perspective)

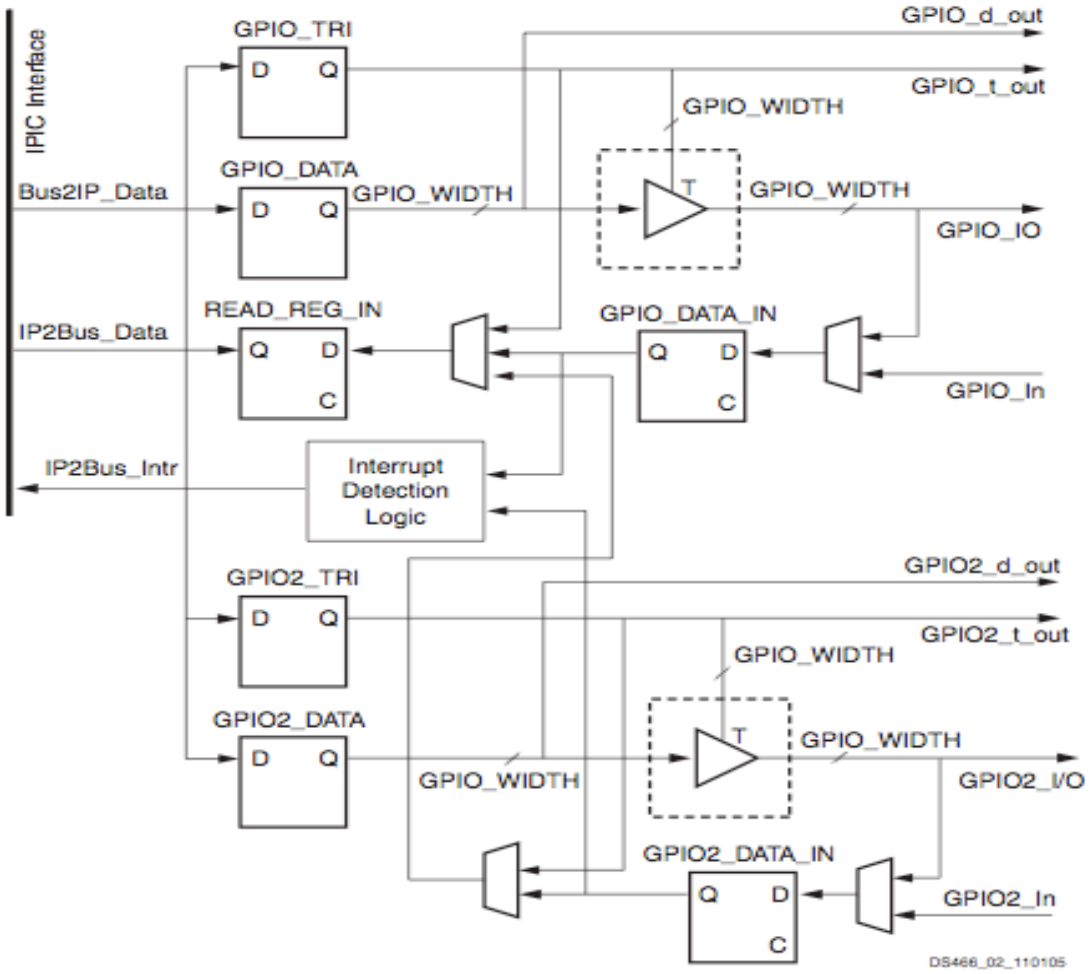


Figure 2: **GPIO_CORE Dual Channel Implementation**

Registers (Programmers Perspective)

- GPIO_TRI := sets up direction and use of Tri-State
  - 0 := write (output) (also turns on tristate connections)
  - 1 := read (input) (disables tristate connections)
    - Tri-state or dedicated input/output pins set during system build

Table 4: **OPB GPIO Registers**

Register Name	Description	OPB Address	Access
GPIO_DATA	Channel 1 OPB GPIO Data Register	C_BASEADDR + 0x00	Read/Write
GPIO_TRI	Channel 1 OPB GPIO 3-state Register	C_BASEADDR + 0x04	Read/Write
GPIO2_DATA	Channel 2 OPB GPIO Data register	C_BASEADDR + 0x08	Read/Write
GPIO2_TRI	Channel 2 OPB GPIO 3-state Register	C_BASEADDR + 0x0C	Read/Write

Data Port

- GPIOx_Data := Port for Data
  - If a bit configured as Output:
    - Writing to it will output the data
    - Reading from it will return last value written
  - If a bit configured as Input
    - Reading will bring in value
    - Writing to it won’t do anything

Example Code Use

```
/* Push buttons are used to control the on-board LEDs. */
// Direction Masks
#define outputDir               0x00000000    // All output bits
#define inputDir                0x0000001F    // 5-input bits

int main()
{
  // Pointer defintions for Button GPIO
  //     ** NOTE - integer definition causes
  //       offsets to be automatically be multiplied by 4!!
  volatile int *base_buttonGPIO = (int*)(0x40040000);
  volatile int *data_buttonGPIO = (int*)(base_buttonGPIO + 0x0);
  volatile int *tri_buttonGPIO    = (int*)(base_buttonGPIO + 0x1);

  // Pointer defintions for LED GPIO
  //     ** NOTE - integer definition causes
  //         offsets to be automatically  be multiplied by 4!!
  volatile int *base_ledGPIO  = (int*)(0x40000000);
  volatile int *data_ledGPIO  = (int*)(base_ledGPIO + 0x0);
  volatile int *tri_ledGPIO     = (int*)(base_ledGPIO + 0x1);

// Variable used to store the state of the buttons
  int data = 0;

// Init. the LED peripheral to outputs
  print("Init. LED GPIO Data Direction...\r\n");
  *tri_ledGPIO = outputDir;

  // Init. the Button peripheral to inputs
  print("Init. Button GPIO Data Direction...\r\n");
  *tri_buttonGPIO = inputDir;

  // Infinitely Loop...
  while(1)
    {  // Read the current state of the push buttons
        data = *data_buttonGPIO;
        xil_printf("buttonState = %d\r\n",data);

        // Set the state of the LEDs
        *data_ledGPIO = data; }
  return 0;
}
```