CSCE 4114
I/O, I/O....its off the chip I go!

Serial and Parallel I/O

David Andrews

dandrews@uark.edu

Serial/Parallel I/O

How do we interface external signals/data into Computer ?

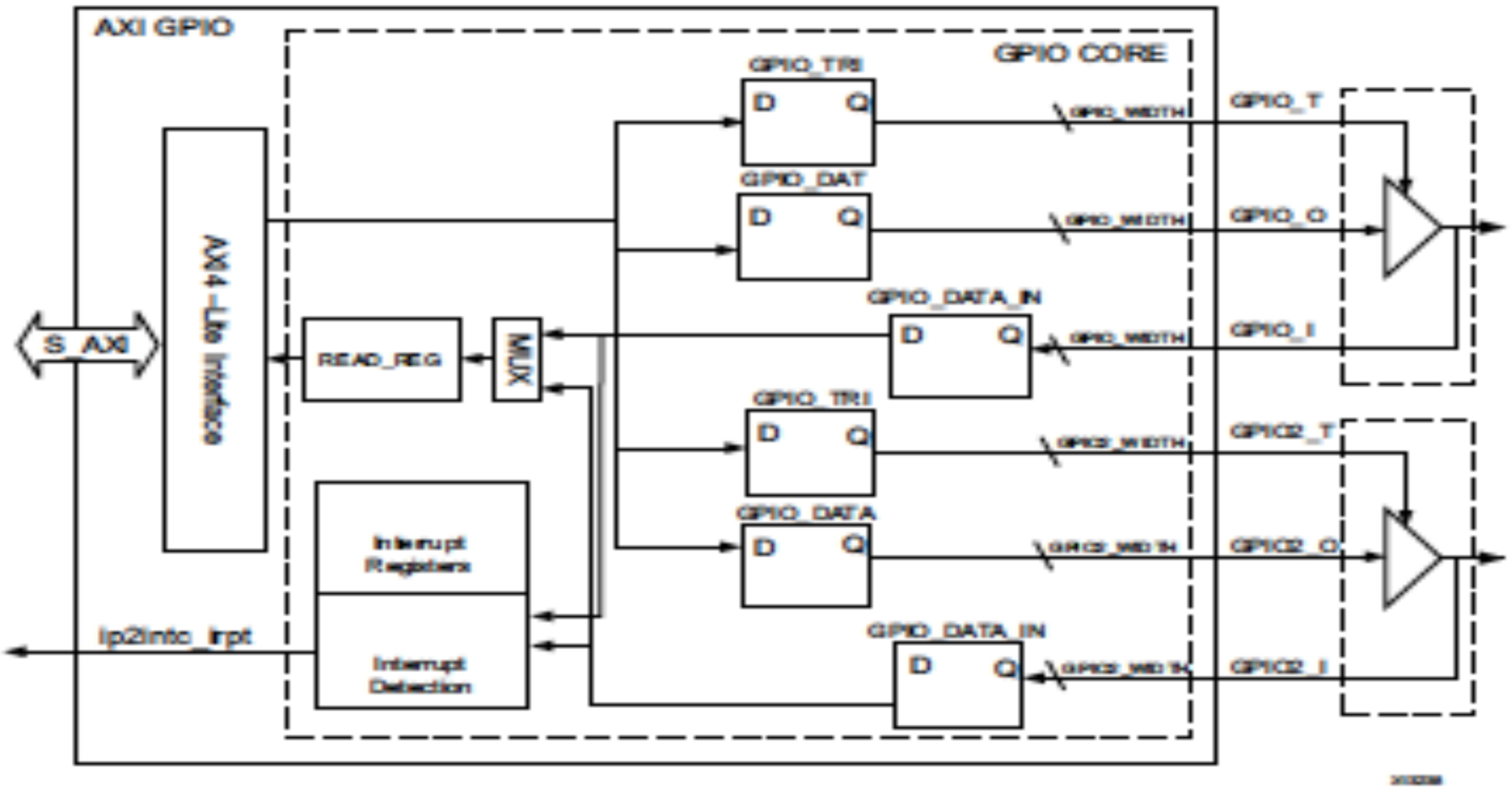
Intel 8255 Programmable Peripheral Interface (PPI)
Motorola 6820 Peripheral Interface Adapter (PIA)

Standard chips that provided general-purpose I/O lines and control lines for handshaking to external devices. Programmability allows different numbers of I/O lines to be set as either inputs or outputs depending on needs.

GPIO: General Purpose Input/Output Core

- Provides all signals/connections to AXI bus
  - AXI (A)dvanced e(X)tensible (I)nterface
  - Part of ARM Advanced Microcontroller Bus Arch (AMBA)
- Can Have 1 or 2 Channels of 32 bits each
- Each bit can be configured as input/output or tri-state

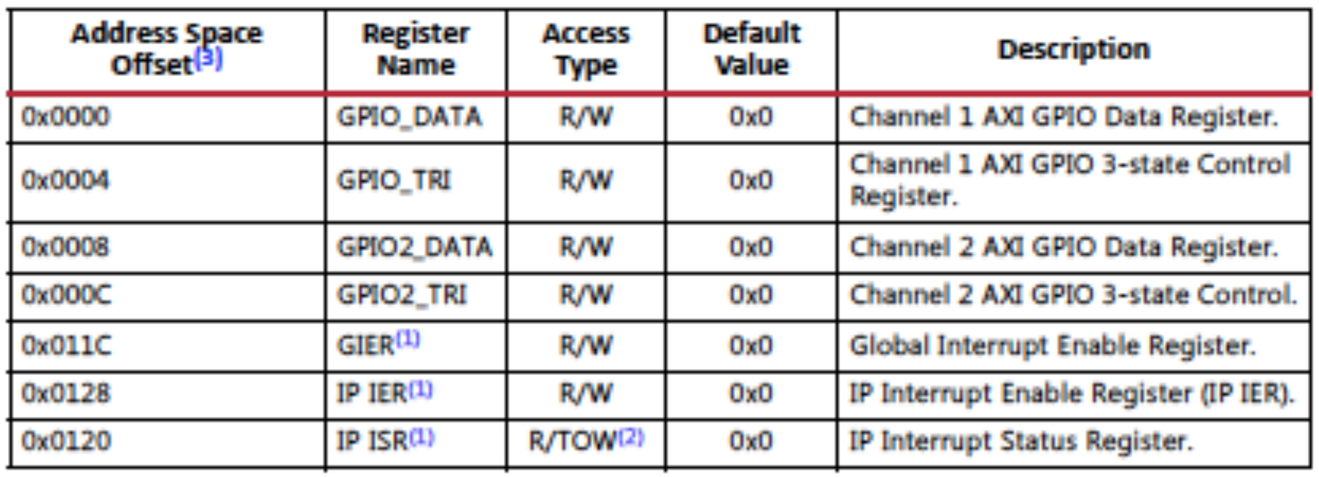
Schematic (Hardware Perspective)



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 2-4: **Registers**



Address Space Offset[3]	Register Name	Access Type	Default Value	Description
0x0000	GPIO_DATA	R/W	0x0	Channel 1 AXI GPIO Data Register.
0x0004	GPIO_TRI	R/W	0x0	Channel 1 AXI GPIO 3-state Control Register.
0x0008	GPIO2_DATA	R/W	0x0	Channel 2 AXI GPIO Data Register.
0x000C	GPIO2_TRI	R/W	0x0	Channel 2 AXI GPIO 3-state Control.
0x011C	GIER[1]	R/W	0x0	Global Interrupt Enable Register.
0x0128	IP IER[1]	R/W	0x0	IP Interrupt Enable Register (IP IER).
0x0120	IP ISR[1]	R/TOW[2]	0x0	IP Interrupt Status Register.

Data Port

- GPIOx_Data := Port for Data
  - If a bit configured as Output:
    - Writing to it will output the data
    - Bit cannot be read
  - 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;
}
```