DISEÑO DE UN PROTOTIPO DE CONTROLADOR PID EN DRONES CON PROCESADOR A LA MEDIDA EN ARQUITECTURA RISC-V PARA IMPLEMENTACIÓN EN FPGA

DESCRIPCION DE BLOQUES, ENTRADAS Y SALIDAS

Autores:

Iván Ricardo Diaz Gamarra
Omar Steck Espinel Santamaria
Magda Daniela Latorre Ortiz

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERIA
DEPARTAMENTO DE ELECTRÓNICA
BOGOTÁ, D.C.

Descripción de bloques y señales

A continuación, se muestra la descripción de los bloques mostrados en la Figura 1, así como la descripción de las señales de interconexión, en caso de que los bloques contengan bloques internos, se mencionan estos bloques junto a una descripción.

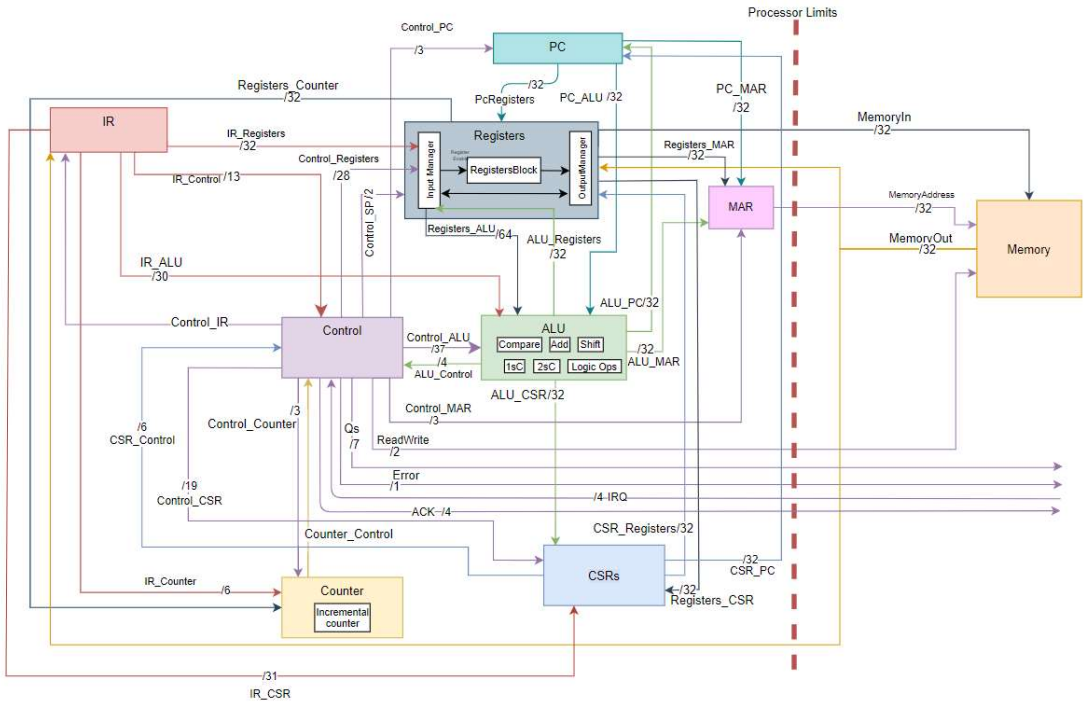

Figura. 1 Diagrama de bloques

ProcessorRV:

Bloque del procesador, contiene las entradas y salidas del sistema. En el diagrama para facilitar la notación, se muestra como la línea punteada que define los límites del procesador.

- Entradas
  - Reset 1 Bit. Señal para reiniciar el sistema
  - Reloj: 1 Bit. Señal de reloj para el funcionamiento de los bloques
  - MemoryOut: 32 bits. Datos provenientes de la memoria
  - IRQ: 4 bits. Bits de interrupciones para periféricos.
- Salidas
  - MemoryAddress: 32 Bits. señal que envía la dirección almacenada en MAR a la memoria.

- Relojs: 1 Bit. Salida de la señal de reloj para análisis, se añade una letra s de salida al final del nombre de la señal para diferenciarla del reloj de entrada.
- error: 1 Bit. Señal de error en caso de no estar en ningún estado.
- Qs: 6 Bits. Q de los estados para comprobación codificado en binario.
- ACK:4 bits, bits de respuesta de interrupciones.
- MemoryIn: 32 bits. Datos provenientes de la memoria

IR:

Instruction Register, registro especial encargado de almacenar las instrucciones provenientes de la memoria y de enviar diferentes partes de la instrucción a los demás bloques para llevar a cabo las acciones necesarias.

- Entradas:
  - Reset 1 Bit. Señal para reiniciar el sistema
  - Reloj: 1 Bit. Señal de reloj para el funcionamiento de los bloques
  - MemoryOut: 32 bits. Salida de datos de la memoria, en este caso, contiene la instrucción
  - Control_IR: 1 bit. Señal del control que indica al IR reemplazar su valor con el valor de Memoryout
- Salidas:
  - IR_Registers: 32 bits envían la dirección de los registros rd, rs1 o rs2 de las instrucciones para acceder a los registros necesarios. La cantidad de registros usados en la instrucción depende de cada instrucción, adicionalmente enviar datos inmediatos para almacenar.
  - IR_Control: 13 bits. Envían al control los opcodes de la instrucción para poder determinar los saltos de las instrucciones.
    - IR [25]|IR [30]|IR[15:12]|IR[6:0]
  - IR_ALU: 30 bits. Envia los datos que deben ser operados en la ALU, tales como datos inmediatos en las instrucciones.
    - IR_ALU (24:0): IR (31: 7)
    - IR_ALU (30:25) IR (6:2)
  - IR_CSRs: 32 bits. Envia las direcciones de los CSR donde se realiza la instrucción. O el IR completo para almacenarlo en las excepciones.
  - IR_Counter: 6 bits. Se usa para cargar directamente un valor al Counter
    - IR_Counter(5:0): (25:20)

PC:

Program Counter, registro especial encargado de almacenar y aumentar la dirección de la memoria donde se recuperan las instrucciones, permite la carga de datos para hacer saltos, al iniciar empieza en la dirección 0x0000.

En esencia el bloque PC, se comporta como un contador unitario, ya que las instrucciones almacenadas en la memoria están sucesivas una de la otra, este contador solo se incrementa.

- Entradas
  - Reset 1 Bit. Señal para reiniciar el sistema
  - Reloj: 1 Bit. Señal de reloj para el funcionamiento de los bloques
  - Control_PC: 3 bits.
    - Control_PC[0]: Señal de control que indica al PC que se auto incremente, estos auto incrementos aumentan 4 al PC actual.
    - Control_PC[1]: Señal de control que indica al PC que reemplace su valor actual, con el valor de ALU_PC.
    - Control_PC[2]: Señal de control que indica al PC que reemplace su valor actual, con el valor de CSR.
  - ALU_PC: 32 bits, contiene el nuevo PC para ser almacenado.
  - CSR_PC: 32 bits, contiene el nuevo valor para PC.
- Salidas
  - PC_MAR: 32 bits. Señal que envía la dirección desde el PC al MAR para obtener la instrucción de memoria.
  - PC_ALU: 32 bits. Señal que envía el contenido del PC a la ALU.

Registers:

Bloque de registros, contiene el selector y 32 registros, como se ve en la Figura. 2

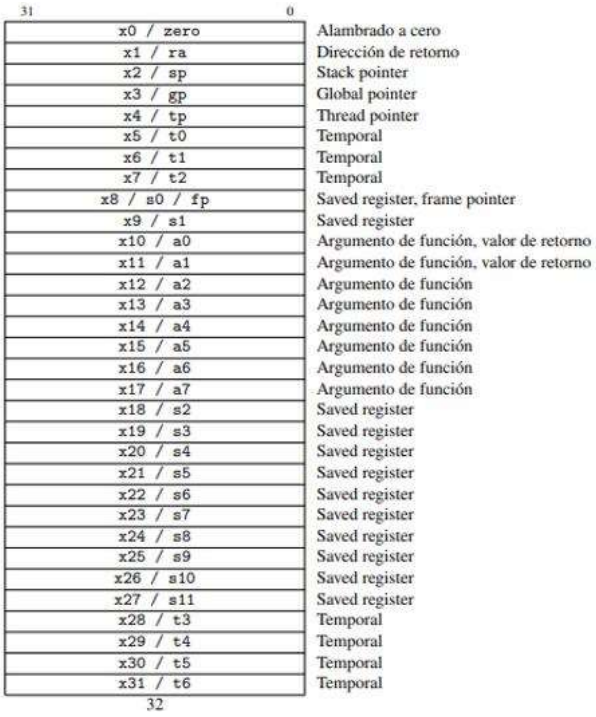

Figura. 2 Distribución de registros

El bloque registers posee un selector que realiza el manejo organiza cuales de los registros son requeridos para esta instrucción, tanto para ser leídos o para almacenar datos dentro de ellos.

- Entradas
  - Reset 1 Bit. Señal para reiniciar el sistema
  - Reloj: 1 Bit. Señal de reloj para el funcionamiento de los bloques
  - IR_Registers: 26 bits. Dirección de los registros a usar en la instrucción y en algunos casos datos *immediate* a guardar.
    - Bits 5:0 = IR (11:7)
    - Bits 10:6 = IR (19:15)
    - Bits 15:11 = IR (24:20)
    - Bits 20:16 = IR (6 : 2)
  - Control_Registers: 25 Bits señal de control a los registros para indicar que datos se asignan a la salida y que datos se guardan, así como la ubicación de estos, se detalla en los decoders.
  - ALU_Registers: 32 bits. Señal de salida de la ALU, envía a los registros el resultado de una operación.
  - MemoryOut: 32 bits. Señal que lleva los datos de memoria a ser almacenados en los registros

- CSR_Registers: 32 bits Señal de los CSR a los registros que contiene el valor a guardar

- Salidas
  - Registers_ALU: 64 bits. Señal que transmite los datos de los registros a la ALU para ser operados.
    - Registers_ALU(63:32) segundo dato 32 bits
    - Registers_ALU(31:0) primer dato 32 bits
  - Registers_Counter: 32 bits. Señal que contiene la cantidad a contar.
  - Memory_In: 32 bits, Señal que contiene el valor a guardar en memoria.
  - Registers_CSR: 32 bits. Señal que contiene un dato a guardar en los CSR.
  - Registers_MAR: 32 bits, señal del banco de registros al MAR para una nueva dirección.

Bloques internos registers

El bloque de registros, mostrado en la Figura. 3 posee los siguientes bloques internos:

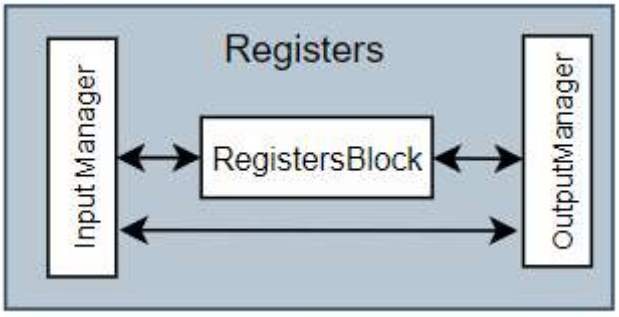

Figura. 3 Bloque de registros

- InputManager, que decodifica la dirección desde el IR para guardar los datos o desde que registros sacar los datos.
- Un banco de registros donde se almacenan los datos, llamado RegisterBlock.
- SP un registro especial con dirección 2, más detalle en bloque SP, incluido dentro del RegistersBlock.
- OutputManager, asigna los datos de salida según los estados del control.

Decoder Registers

El *decoder* de los registros funciona mediante una señal de control que ha sido diseñada para agrupar los estados con las mismas operaciones sobre los registros y datos de entrada y/o salida, al encontrarse en alguno de esos estados agrupados se escribe un 1 en una de las señales correspondientes, esta señal

llega al *decoder* de los registros que realiza la operación con los datos de entrada y salida ya configurados para los estados de esa señal correspondiente.

MAR:

Memory address register. Registro de dirección para almacenar o leer datos de la memoria. Se compone de un registro para una palabra de 32 bits, donde se almacena la dirección de memoria, esta dirección se recibe de ALU, PC o Registers y mediante una señal de control, se selecciona por cuál de las señales de los otros bloques se reemplaza.

- Entradas:
  - Reset: 1 Bit. Señal para reiniciar el sistema
  - Reloj: 1 Bit. Señal de reloj para el funcionamiento de los bloques
  - Control_MAR: 3 bits. Señal que maneja el comportamiento del MAR de acuerdo con los estados:
    - Bit 0: bit para que reciba Pc
    - Bit 1: bit para que reciba ALU
    - Bit 2: bit para que reciba registers
  - ALU_MAR: 32 bits. Señal que envía el resultado de una operación de la ALU al MAR
  - PC_MAR: 32 bits. Señal que envía la dirección desde el PC al MAR para obtener la instrucción de memoria.
  - Registers_MAR: 32 bits, señal del banco de registros al MAR para una nueva dirección.
- Salidas:
  - MemoryAddress: 32 bits. Señal que envía la dirección almacenada en MAR a la memoria.

Control:

El bloque control, contiene la máquina de estados del procesador, esta decide el estado actual del procesador basado en las instrucciones, que se transmiten mediante la señal del IR y las entradas de control, y envía señales a los demás bloques ejecutar las diferentes operaciones que componen cada instrucción.

- Entradas
  - Reset 1 Bit. Señal para reiniciar el sistema
  - Reloj: 1 Bit. Señal de reloj para el funcionamiento de los bloques

- IR_Control: 13 bits. Envía al control los opcodes de la instrucción para poder determinar los saltos de las instrucciones.
  - IR[25]|IR[30]|IR[15:12]|IR[6:0]
- ALU_Control: 4 bits
  - ALU_Control[0] Bandera de fin de operación de la ALU
  - ALU_Control[1] bandera de cero, se pone en uno si el resultado de una operación de ALU es cero
  - ALU_Control[2] bit de signo resultado de ALU
  - ALU_Control[3] bit de carry resultado de ALU
- Counter_Control:1 bit Señal que notifica el fin del conteo.
- IRQ: 4 bits. Bits de interrupciones para periféricos.
- CSR_Control: 6 bits Señal que envía los datos de banderas importantes para cambios de estados y manejo de interrupciones al control.

- Salidas
  - Control_MAR: 3 bits. Señal que maneja el comportamiento del MAR de acuerdo con los estados:
    - Bit 0: bit para que reciba Pc
    - Bit 1: bit para que reciba ALU
    - Bit 2: bit para que reciba registers
  - Control_PC: 3 bits.
    - Control_PC[0]: Señal de control que indica al PC que se auto incremente, estos auto incrementos aumentan 4 al PC actual.
    - Control_PC[1]: Señal de control que indica al PC que reemplace su valor alctual, con el valor de ALU_PC.
    - Control_PC[2]: Señal de control que indica al PC que reemplace su valor alctual, con el valor CSR_PC.
  - Control_Registers: 28 Bits señal de control a los registros para indicar que datos se asignan a la salida y que datos se guardan, así como la ubicación de estos, se detalla en los decoders.
  - Control_IR: una señal de control para que reemplace su valor por el valor del cable Memoryout
  - Control_ALU: 37 bits. Señal del control a la ALU, que dependiendo del estado indica cual operación, con que datos se lleva acabo y en donde se almacena. Para más detalle se ve en el decoder.

- Control_Counter: 3 bits. Señal de control para el contador.
  - Control_Counter[0]: Señal de control hacia el contador para iniciar la cuenta.
  - Control_Counter[1]: Señal de control hacia el contador para reemplazar el valor de la cuenta por IR.
  - Control_Counter[2]: Señal de control hacia el contador para reemplazar el valor de la cuenta por Registers.
- Control_SP: 2 bits señal para el comportamiento del SP.
  - Bit 0: Aumenta el conteo por 4.
  - Bit 1: Disminuye el conteo por cuatro.
- Qs: 85 bits. Señal de los Q de la máquina de estados del control para verificar
- Read: Señal para la memoria de escritura o lectura
  - Read = 1 se activa la lectura
  - Read = 0 se activa la escritura
- Error: 1 bit Señal de error de la máquina de estados
- Control_CSR: 18 bits. Señal de activación de los CSR,
- ACK:4 bits, bits de respuesta de interrupciones

ALU:

Aritmetic logic unit. Bloque que realiza las operaciones aritméticas y lógicas del procesador, generando a partir de una señal de control el resultado de una operación específica a los datos ingresados.

- Entradas:
  - Reset 1 Bit. Señal para reiniciar el sistema
  - Reloj: 1 Bit. Señal de reloj para el funcionamiento de los bloques
  - Registers_ALU: 64 bits. Señal que transmite los datos de los registros a la ALU para ser operados.
  - IR_ALU: 30 bits. Envía los datos que deben ser operados en la ALU, tales como datos inmediatos en las instrucciones.
    - IR_ALU (24:0): IR (31 : 7)
    - IR_ALU (30:25) IR(6:2)
  - Control_ALU: 37 bits. Señal del control a la ALU, que dependiendo del estado indica cual operación, con que datos se lleva acabo y en donde se almacena.
  - PC_ALU: 32 bits. Señal del PC que es operada para los saltos del PC.
- Salidas:

- ALU_Registers: 32 bits. Señal de salida de la ALU, envía a los registros el resultado de una operación.
- ALU_PC: 32 bits, contiene el nuevo PC para ser almacenado.
- ALU_CSR: 32 bits. Señal de salida de la ALU, envía a los CSR el resultado de una operación.
- ALU_MAR: 32 bits señal que envía el resultado de una operación de la ALU al MAR
- ALU_Control: 4 bits
  - ALU_Control[0] Bandera de fin de operación de la ALU
  - ALU_Control[1] bandera de cero, se pone en uno si el resultado de una operación de ALU es cero.
  - ALU_Control[2] bit de signo resultado de ALU
  - ALU_Control[3] bit de carry resultado de ALU

Bloques internos ALU

A continuación se realiza una descripción funcional sobre los bloques internos de la ALU, en algunos casos se indica la cantidad de ciclos de reloj que requiere ese bloque para su operación, para los bloques que no tengan indicada la duración de su operación, esta es de 1 ciclo de reloj, excepto para los bloques de Shift, donde la duración de la operación depende de la cantidad de bits que se desea desplazar, con un tiempo de un ciclo de reloj por cada desplazamiento.

Decoder:

Debido a que los datos que ingresan a la ALU para ser operados están organizados de muchas formas, debido a los varios modos de direccionamiento, es necesario un *decoder* para encargarse de:

- Cuál de las entradas se debe operar.
- De qué forma están organizados los datos a operar.
- Cuál de las operaciones se debe llevar a cabo.
- De cuál de los bloques se genera la salida de la ALU.
- A cuál de las señales de asigna la salida del bloque.

El *decoder* de la ALU funciona mediante una señal de control que ha sido diseñada para agrupar los estados con las mismas operaciones y datos de entrada y salida, al encontrarse en alguno de esos estados agrupados se escribe un 1 en una de las señales correspondientes, esta señal llega al *decoder* de la ALU que realiza la operación con los datos de entrada y salida ya configurados para los estados de esa señal correspondiente.

Sumador:

El sumador realiza la suma o resta de dos datos de 32 bits, en el caso de la resta se realiza haciendo el complemento a 2 del segundo número, el resultado de la suma es un dato de 32 bits y el carry de salida del ultimo sumador, esta operación se realiza en un ciclo de reloj, el sumador esta construido usando una arquitectura Carry Ripple Adder

LogicalShiftRight:

El *Logical Shift Right* es un registro especial, que almacena 32 bits y de acuerdo con una señal de control, desplaza el contenido a la derecha un bit en un ciclo de reloj, reemplazando su bit más significativo, por un cero.

LeftShift

El *Left Logical Shift*, desplaza el contenido del registro hacia la izquierda una vez por ciclo de reloj reemplazando el bit menos significativo por cero.

ArithmeticShiftRight

El *Arithmetic Shift Right*, desplaza el contenido del registro hacia la derecha, replicando el valor del bit más significativo.

And, Or y Xor

El diseño de los bloques lógicos AND, OR y XOR se compone de 32 compuertas del respectivo tipo, AND, OR o XOR operando uno a uno los bits de los dos operandos de entrada y su resultado son los 32 bits de salida del bloque.

Multiplier32Bits

El bloque de multiplicación realiza la operación de multiplicación con signo entre dos datos con signo de 32 bits y a la salida entrega los 32 bits menos significativos de la respuesta, de acuerdo con la operación MUL, descrita en la sección 4. Esta multiplicación se realiza en un ciclo de reloj usando el algoritmo de Booth modificado.

CSR:

El Control and Status Registers, es un bloque de registros propio del estándar RISCV, este bloque, está compuesto de un conjunto de registros útiles al sistema para desarrollar las instrucciones, como registros para interrupciones y variables de estado útiles al control.

- Entradas
    - Reset 1 Bit. Señal para reiniciar el sistema
    - Reloj: 1 Bit. Señal de reloj para el funcionamiento de los bloques

    - ALU_CSR: 32 bits. Señal de salida de la ALU, envía a los CSR el resultado de una operación.
    - IR_CSRs: 12 bits. Envía las direcciones de los CSR donde se realiza la instrucción.
        - IR_CSRs(11:0) : IR(31:20)
    - Registers_CSR : 32 bits. Envía los datos para escribir en los CSR
    - Control_CSR: 19 bits. Señal de activación de los CSR
- Salidas
    - CSR_PC: 32 bits, contiene el nuevo valor para PC.
    - CSR_Registers : 32 bits. Envía a los registros datos
    - CSR_Control: 6 bits Señal que envía los datos de banderas importantes para cambios de estados y manejo de interrupciones al control.

Counter:

Bloque de conteo, se compone de un contador descendente activado por el control, que recibe de forma paralela un valor a contar, mediante una señal de control disminuye de forma unitaria por cada ciclo de reloj, al alcanzar el cero, levanta una bandera hacia control.

- Entradas
    - Reset 1 Bit. Señal para reiniciar el sistema
    - Reloj: 1 Bit. Señal de reloj para el funcionamiento de los bloques
    - Registers_Counter: 32 bits. Señal que contiene la cantidad a contar.
    - IR_Counter : 6 bits. Se usa para cargar directamente un valor al Counter
        - IR_Counter(5:0): (25:20)
    - Control_Counter: 3 bits. Señal de control para el contador.
        - Control_Counter[0]: Señal de control hacia el contador para iniciar la cuenta.
        - Control_Counter[1]: Señal de control hacia el contador para reemplazar el valor de la cuenta por IR.
        - Control_Counter[2]: Señal de control hacia el contador para reemplazar el valor de la cuenta por Registers.
- Salidas
    - Counter_Control: 1 Bit. Señal que notifica el fin del conteo.

SP:

Stack Pointer, es un registro especial que almacena la dirección de memoria correspondiente al Stack, un espacio especial de la memoria, donde se almacenan los datos para hacer llamados a subrutinas.

Entradas:

- Control_SP: 2 bits señal para el comportamiento del SP.
  - Bit 0: Aumenta el conteo por 4.
  - Bit 1: Disminuye el conteo por cuatro.

Salidas:

- SPRegister: salida de 32 bits que contiene el valor actual de SP

Aunque el *stack pointer* es un registro especial, de acuerdo con el estándar, se ubica en el bloque de registros, para escribir la información de este o usarlo para operaciones se usan las operac9iones sobre el bloque de registros con la dirección 0 T 010.