Arquitectura de Computadoras

Informe de Trabajo Final

Repositorio

CHAGAY Adriel, Matrícula: 19021538
SEGURA Gaspar, Matrícula: 40416697

Índice

Índice **2**
Introducción **3**
Diseño y desarrollo **5**
Diagrama de bloques 6
Etapa de Instruction Fetch (IF) **7**
Módulos 8
Etapa de Instruction Decode (ID) **12**
Módulos 13
Etapa Execution (EX) **19**
Módulos 20
Etapa de Memory Access (MA) **24**
Módulos 25
Etapa de Writeback (WB) **27**
Módulos 28
Características Comunes de Todos los Registros Pipeline 30
Comunicación Serial **31**
UART_tick_gen 31
UART_tx 32
UART_rx 33
UART_TOP (Módulo Principal) 34
Integración de Etapas en el Pipeline **36**

Introducción

El trabajo consiste en el diseño e implementación de un procesador MIPS en una placa FPGA. El mismo debe contar con las siguientes características:

- Segmentación del procesador en las siguientes etapas:
  - IF (Instruction Fetch): Búsqueda de la instrucción en la memoria de programa.
  - ID (Instruction Decode): Decodificación de la instrucción y lectura de registros.
  - EX (Execute): Ejecución de la instrucción propiamente dicha.
  - MA (Memory Access): Lectura o escritura desde/hacia la memoria de datos.
  - WB (Write back): Escritura de resultados en los registros.
- Implementación de las siguientes operaciones:
  - R-type: SLL, SRL, SRA, SLLV, SRLV, SRAV, ADDU, SUBU, AND, OR, XOR, NOR, SLT.
  - I-Type: LB, LH, LW, LWU, LBU, LHU, SB, SH, SW, ADDI, ANDI, ORI, XORI, LUI, SLTI, BEQ, BNE, J, JAL.
  - J-Type: JR, JALR.
- Contar con soporte para los siguientes tipos de riesgo:
  - Estructurales: Se producen cuando dos instrucciones tratan de utilizar el mismo recurso en el mismo ciclo.
  - De datos: Se intenta utilizar un dato antes de que esté preparado. Mantenimiento del orden estricto de lecturas y escrituras.
  - De control: Intentar tomar una decisión sobre una condición todavía no evaluada.
- Implementación de las siguientes unidades:
  - Unidad de cortocircuitos.
  - Unidad de detección de riesgos.
  - Unidad de Debug.
- Cumplir con los siguientes requerimientos:
  - El programa a ejecutar debe ser cargado en la memoria del programa mediante un archivo ensamblado.
  - Debe implementarse un programa ensamblador. Debe transmitirse ese programa mediante interfaz UART antes de comenzar a ejecutar.
  - Se debe incluir una unidad de Debug que envíe información hacia y desde la PC mediante la UART. La misma debe poder enviar mediante
  - UART:
    - PC y contador de ciclos.
    - Contenido de los registros.
    - Contenido de la memoria de datos.
- Contar con los siguientes modos de operación:
  - Antes de estar disponible para ejecutar, el procesador está a la espera para recibir un programa mediante la Debug Unit.
  - Una vez cargado el programa, debe permitir dos modos de operación:
    - Continuo, se envía un comando a la FPGA por la UART y esta inicia la ejecución del programa hasta llegar al final del mismo (Instrucción HALT). Llegado ese punto se muestran todos los valores indicados en pantalla.
    - Paso a paso: Enviando un comando por la UART se ejecuta un ciclo de Clock. Se debe mostrar a cada paso los valores indicados.

Diseño y desarrollo

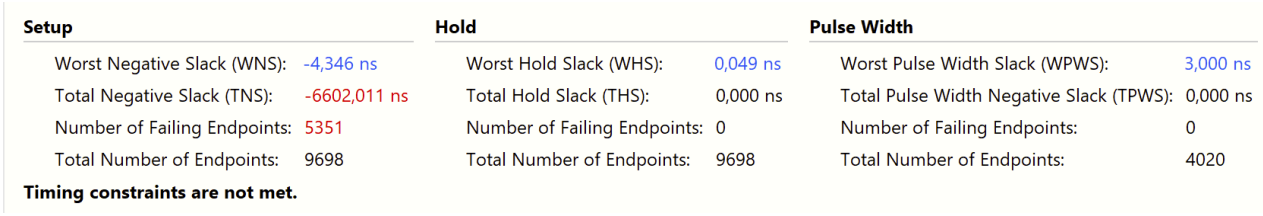
Debido a la complejidad y el tamaño del diseño, se decidió dividirlo en dos etapas. Durante la primera etapa, se hizo enfoque en conseguir un procesador segmentado, centrándonos en el modelado de las etapas y los módulos que las componen, definiendo las operaciones y algunos de los elementos. Se hicieron a un lado las unidades de debug, cortocircuito y detección de riesgos, por agregar una complejidad que podría dificultar conseguir un procesador funcional. Al final de esta etapa, se consiguió un procesador segmentado que puede ejecutar cualquier operación del set, pero de manera aislada (por no contar con herramientas para sortear los riesgos).

Durante la segunda etapa y habiendo comprobado el funcionamiento base del procesador, se modelaron y diseñaron las unidades excluidas durante la primera. Se adaptaron las partes necesarias y también se agregó el módulo necesario para la comunicación UART con el dispositivo.

A continuación, se detalla la composición final del sistema. En cada apartado se describen los módulos, así como las decisiones de diseño que se tomaron para cada uno.

Análisis de Tiempo

Primero con un clock de 100Mhz:



Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	-4,346 ns	Worst Hold Slack (WHS):	0,049 ns	Worst Pulse Width Slack (WPWS):	3,000 ns
Total Negative Slack (TNS):	-6602,011 ns	Total Hold Slack (THS):	0,000 ns	Total Pulse Width Negative Slack (TPWS):	0,000 ns
Number of Failing Endpoints:	5351	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	9698	Total Number of Endpoints:	9698	Total Number of Endpoints:	4020

Timing constraints are not met.

- Setup
  - Worst Negative Slack (WNS): -4.346 ns. Esto indica que la ruta crítica más lenta del diseño tarda 4.346 ns más de lo permitido por el período del reloj. En otras palabras, la señal llega tarde a su flanco de reloj, lo que puede causar errores de temporización.
  - Total Negative Slack (TNS): -6602.011 ns. Es la suma de todos los desfases negativos en las rutas que no cumplen el setup. Es un indicador global del “grado de violación” del timing. Un TNS muy grande sugiere que muchas rutas tienen problemas, no solo la peor.
  - Number of Failing Endpoints: 5351. Número de flancos de reloj / registros donde las condiciones de setup no se cumplen. Esto confirma que el problema es generalizado, no aislado.
- Hold Timing
  - Worst Hold Slack: 0.046 ns. Esto indica que la ruta más corta cumple por 46 ps, apenas positiva.
  - Total Hold Slack: 0 ns. No hay rutas con hold negativo.
  - Failing Endpoints: 0. Todas las rutas cumplen la condición de hold.

El camino crítico:

Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Cl...	Clock Uncertainty
Path 1	-4.346	11	93	ID_EX/o_rt_reg[2]/C	IF/u_PC/o_pc_reg[3]_rep/D	9.201	2.521	6.680	5.000	clk_out1_clk_w...	clk_out1_clk_w...	0.074

En resumen, el diseño no cumple con el timing de setup en muchas rutas. Esto puede provocar errores de datos o comportamiento incorrecto en el hardware. La ruta más crítica falla por ~4.3 ns, lo que es bastante significativo.

Con un clock de 75Mhz:

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	-2,545 ns	Worst Hold Slack (WHS):	0,065 ns	Worst Pulse Width Slack (WPWS):	3,000 ns
Total Negative Slack (TNS):	-626,167 ns	Total Hold Slack (THS):	0,000 ns	Total Pulse Width Negative Slack (TPWS):	0,000 ns
Number of Failing Endpoints:	1124	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	9674	Total Number of Endpoints:	9674	Total Number of Endpoints:	4007

Timing constraints are not met.

Los límites de tiempo aún se sobrepasan pero por menor margen que en 100Mhz.

Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Cl...	Clock Uncertainty
Path 1	-2.545	12	135	ID_EX/o_rs_reg[2]/C	IF/u_PC/o_pc_reg[9]/D	9.092	2.635	6.457	6.667	clk_out1_clk_w...	clk_out1_clk_w...	0.119

Con un clock de 50Mhz:

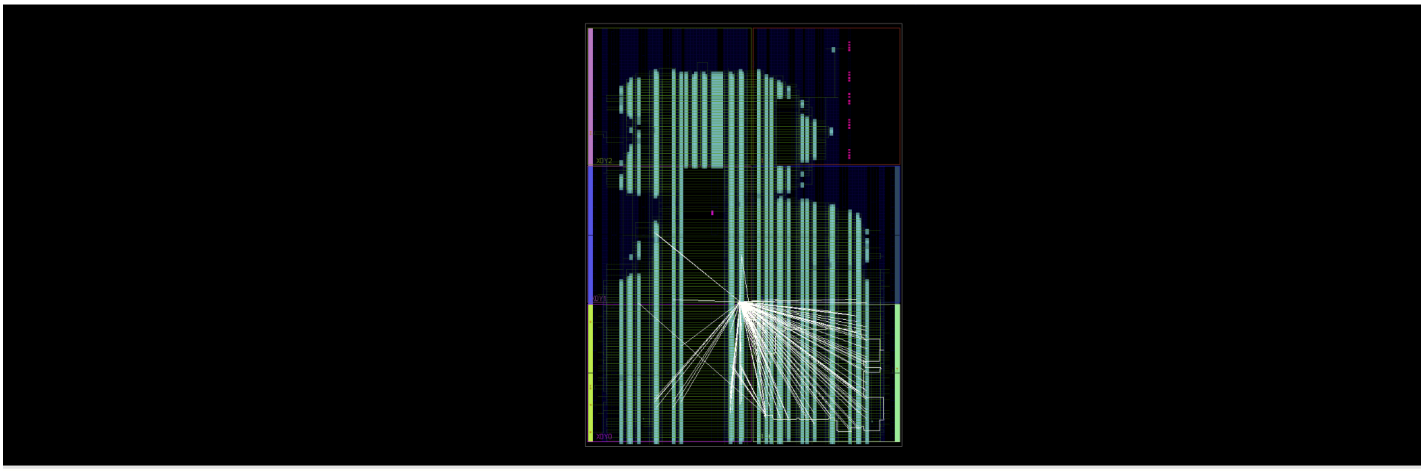
Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0,269 ns	Worst Hold Slack (WHS):	0,021 ns	Worst Pulse Width Slack (WPWS):	3,000 ns
Total Negative Slack (TNS):	0,000 ns	Total Hold Slack (THS):	0,000 ns	Total Pulse Width Negative Slack (TPWS):	0,000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	12924	Total Number of Endpoints:	12908	Total Number of Endpoints:	5777

All user specified timing constraints are met.

- Setup
  - WNS (Worst Negative Slack): 0.269 ns. El camino más crítico del diseño tiene 0.269 ns de margen positivo. Como no es negativo, significa que el diseño cumple con la restricción de tiempo de setup.
  - TNS (Total Negative Slack): 0 ns. No hay violaciones de tiempo en todo el diseño.
  - Número de Failing Endpoints: 0. Ningún registro está incumpliendo el tiempo.

- Hold
  - WHS (Worst Hold Slack): 0.021 ns. El peor camino de hold también cumple, aunque con un margen muy pequeño.
- Pulse Width
  - WPWS (Worst Pulse Width Slack): 3.000 ns. También está bien, con un margen amplio.

El diseño cumple todas las restricciones de tiempo. El path crítico es el del WNS = 0.269 ns, y ese valor representa qué tan cerca está el circuito de fallar en su frecuencia objetivo



Design Runs | Power | DRC | Methodology | Timing

Intra-Clock Paths - clk_out1_clk_wiz_0 - Setup

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Cl...	Clock Uncertainty
Path 1	0.269	1	1	125	EX_MA/o_flg_mem_size_reg[1]_rep__8/C	MA/u_Data_Me...g[229][1]/D	9.568	0.580	8.988	10.0	clk_out1_clk_w...	clk_out...k_wiz_0	0.084

Al clickear en WNS se puede ver el camino crítico realizado durante el reporte.

A 50 MHz el período es 20.0 ns. Con WNS = +0.269 ns, el retardo del peor camino es ≈ 19.731 ns; por tanto la Fmax ≈ 1 / 19.731 ns ≈ 50.68 MHz.
Es decir, se podría aumentar apenas ~0.7 % (50.5–50.7 MHz) antes de estar al límite, y eso sin contar variación de PVT (temperatura/voltaje), incertidumbre de reloj o cambios de ruteo.

Además, el WHS = +0.021 ns en hold es minúsculo: cualquier “optimización” o cambio de ruteo podría romper hold aunque el setup mejore.

Con 50.76Mhz:

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	-0,060 ns	Worst Hold Slack (WHS):	0,056 ns	Worst Pulse Width Slack (WPWS):	3,000 ns
Total Negative Slack (TNS):	-0,060 ns	Total Hold Slack (THS):	0,000 ns	Total Pulse Width Negative Slack (TPWS):	0,000 ns
Number of Failing Endpoints:	1	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	9651	Total Number of Endpoints:	9651	Total Number of Endpoints:	3992

Timing constraints are not met.

No se respetan los límites de tiempo necesarios para funcionar correctamente pero se obtiene 1 solo camino crítico que ha fallado:

Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Cl...	Clock Uncertainty
Path 1	-0.060	11	93	ID_EX/o_rt_reg[0]/C	IF/u_PC/o_pc_reg[1]_rep__5/D	9.584	2.500	7.084	9.9	clk_out1_clk_w...	clk_out...k_wiz_0	0.247
Path 2	0.059	11	93	ID_EX/o_rt_reg[0]/C	IF/u_PC/o_pc_reg[4]/D	9.461	2.500	6.961	9.9	clk_out1_clk_w...	clk_out...k_wiz_0	0.247
Path 3	0.117	11	93	ID_EX/o_rt_reg[0]/C	IF/u_PC/o_pc_reg[1]_rep__15/D	9.421	2.500	6.921	9.9	clk_out1_clk_w...	clk_out...k_wiz_0	0.247
Path 4	0.143	11	93	ID_EX/o_rt_reg[0]/C	IF/u_PC/o_pc_reg[1]_rep__6/D	9.385	2.500	6.885	9.9	clk_out1_clk_w...	clk_out...k_wiz_0	0.247
Path 5	0.150	11	93	ID_EX/o_rt_reg[0]/C	IF/u_PC/o_pc_reg[2]_rep__1/D	9.382	2.500	6.882	9.9	clk_out1_clk_w...	clk_out...k_wiz_0	0.247
Path 6	0.159	11	93	ID_EX/o_rt_reg[0]/C	IF/u_PC/o_pc_reg[1]_rep__16/D	9.384	2.500	6.884	9.9	clk_out1_clk_w...	clk_out...k_wiz_0	0.247
Path 7	0.163	11	93	ID_EX/o_rt_reg[0]/C	IF/u_PC/o_pc_reg[2]_rep__5/D	9.365	2.500	6.865	9.9	clk_out1_clk_w...	clk_out...k_wiz_0	0.247
Path 8	0.164	11	93	ID_EX/o_rt_reg[0]/C	IF/u_PC/o_pc_reg[2]_rep__0/D	9.385	2.500	6.885	9.9	clk_out1_clk_w...	clk_out...k_wiz_0	0.247
Path 9	0.174	11	93	ID_EX/o_rt_reg[0]/C	IF/u_PC/o_pc_reg[3]_rep/D	9.341	2.500	6.841	9.9	clk_out1_clk_w...	clk_out...k_wiz_0	0.247
Path 10	0.185	11	93	ID_EX/o_rt_reg[0]/C	IF/u_PC/o_pc_reg[2]_rep/D	9.355	2.500	6.855	9.9	clk_out1_clk_w...	clk_out...k_wiz_0	0.247

Se intenta con 50.75Mhz hasta 50.72Mhz, siendo esta última frecuencia la máxima frecuencia alcanzada donde los límites de tiempo son respetados.

Con 50.72Mhz:

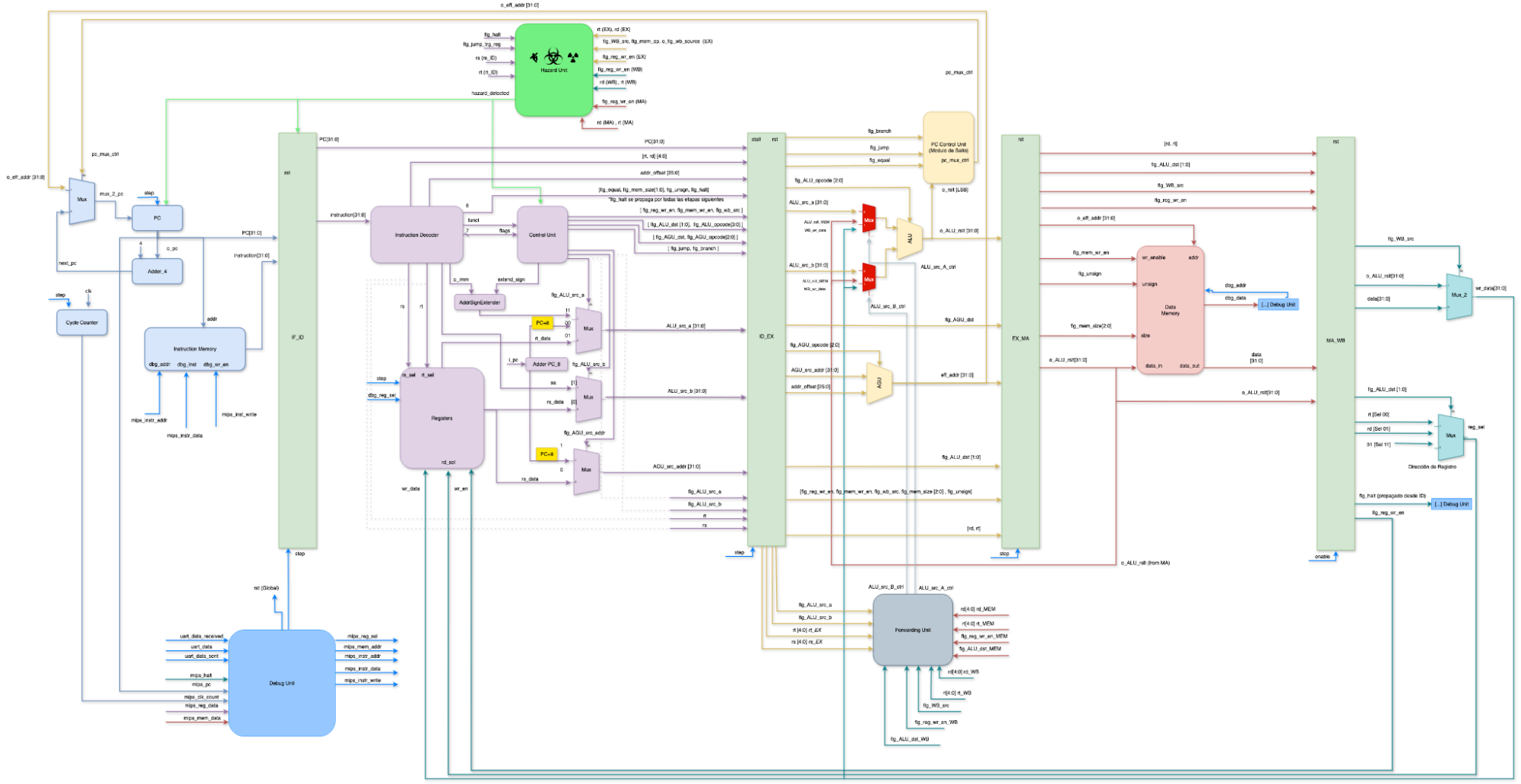
Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0,063 ns	Worst Hold Slack (WHS):	0,037 ns	Worst Pulse Width Slack (WPWS):	3,000 ns
Total Negative Slack (TNS):	0,000 ns	Total Hold Slack (THS):	0,000 ns	Total Pulse Width Negative Slack (TPWS):	0,000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	9657	Total Number of Endpoints:	9657	Total Number of Endpoints:	3994

All user specified timing constraints are met.

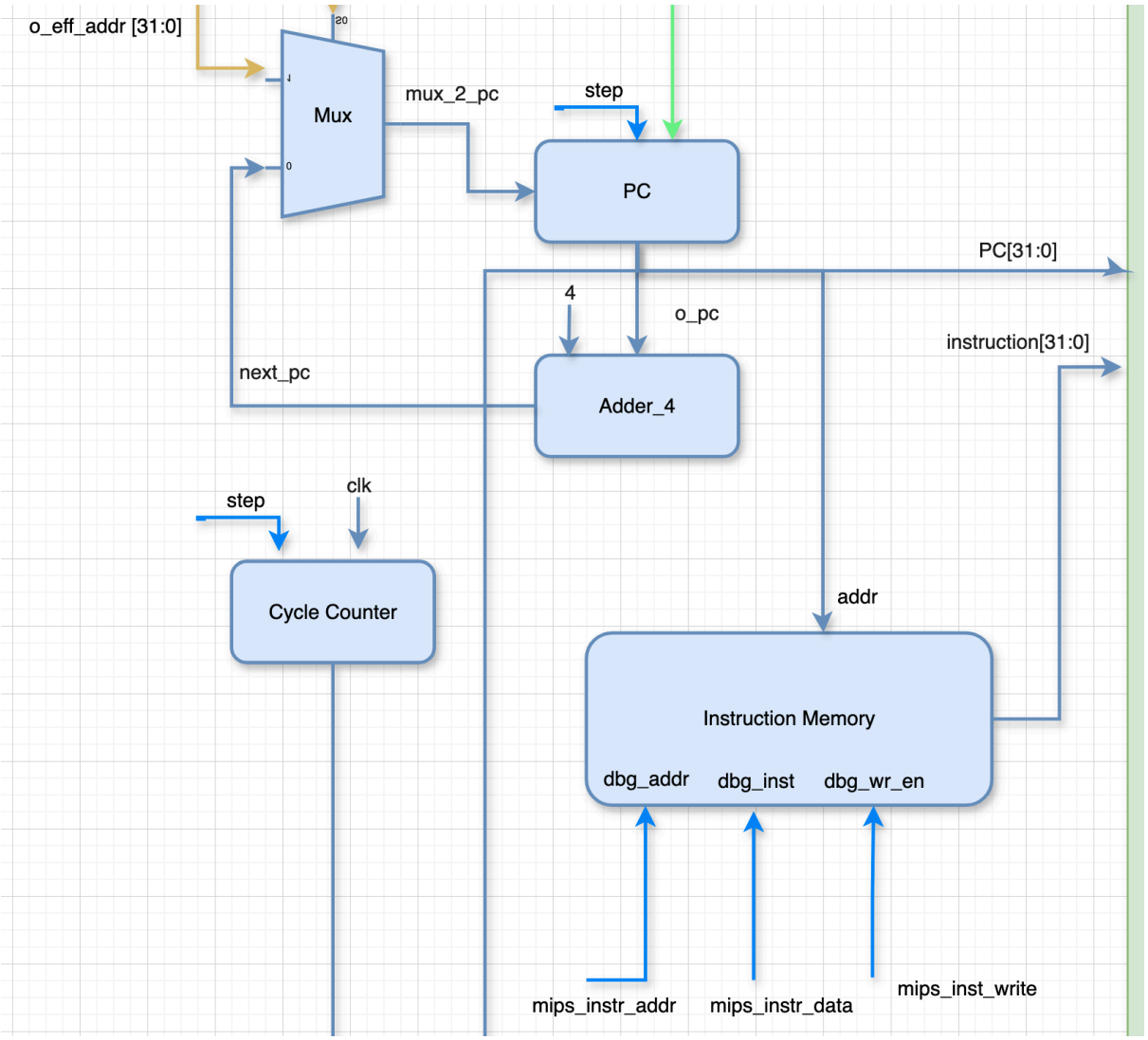
Name	Slack ^1	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Cl...	Clock Uncertainty
Path 1	0.063	13	93	EX_MA/o_rt_reg[2]/C	IF/u_PC/o_pc_reg[2]_rep__0/D	9.524	2.377	7.147	9.858	clk_out1_clk_w...	clk_out1_clk_w...	0.173
Path 2	0.068	13	93	EX_MA/o_rt_reg[2]/C	IF/u_PC/o_pc_reg[1]_rep__14/D	9.561	2.377	7.184	9.858	clk_out1_clk_w...	clk_out1_clk_w...	0.173
Path 3	0.089	13	93	EX_MA/o_rt_reg[2]/C	IF/u_PC/o_pc_reg[2]_rep__4/D	9.484	2.377	7.107	9.858	clk_out1_clk_w...	clk_out1_clk_w...	0.173
Path 4	0.183	13	93	EX_MA/o_rt_reg[2]/C	IF/u_PC/o_pc_reg[2]/D	9.417	2.377	7.040	9.858	clk_out1_clk_w...	clk_out1_clk_w...	0.173
Path 5	0.204	13	93	EX_MA/o_rt_reg[2]/C	IF/u_PC/o_pc_reg[2]_rep__3/D	9.352	2.377	6.975	9.858	clk_out1_clk_w...	clk_out1_clk_w...	0.173
Path 6	0.222	13	93	EX_MA/o_rt_reg[2]/C	IF/u_PC/o_pc_reg[4]/D	9.358	2.377	6.981	9.858	clk_out1_clk_w...	clk_out1_clk_w...	0.173
Path 7	0.228	13	93	EX_MA/o_rt_reg[2]/C	IF/u_PC/o_pc_reg[3]_rep__0/D	9.410	2.377	7.033	9.858	clk_out1_clk_w...	clk_out1_clk_w...	0.173
Path 8	0.230	13	93	EX_MA/o_rt_reg[2]/C	IF/u_PC/o_pc_reg[3]_rep/D	9.350	2.377	6.973	9.858	clk_out1_clk_w...	clk_out1_clk_w...	0.173
Path 9	0.235	13	93	EX_MA/o_rt_reg[2]/C	IF/u_PC/o_pc_reg[1]_rep__15/D	9.360	2.377	6.983	9.858	clk_out1_clk_w...	clk_out1_clk_w...	0.173
Path 10	0.240	13	93	EX_MA/o_rt_reg[2]/C	IF/u_PC/o_pc_reg[1]_rep__11/D	9.332	2.377	6.955	9.858	clk_out1_clk_w...	clk_out1_clk_w...	0.173

Diagrama de bloques



Etapa de Instruction Fetch (IF)

Este apartado describe los módulos utilizados en la etapa de búsqueda de instrucciones (IF) del procesador MIPS implementado en el módulo `Top_Instruction_Fetch`.



Módulos

Program Counter (PC)

Entradas

- **i_clk**
- **i_rst**
- **i_next_pc** [32] (mux_2_pc)
- **i_step**
- **hazard_detected** (en verde, viene de la unidad de hazard)

Salidas

- **o_pc** [32]: Conectado al módulo Adder_4, a memoria de instrucciones y a salida de la etapa.

Sincronismo

- Negedge

Descripción

Como su nombre lo indica, este módulo se encarga de indicar en qué posición del programa nos encontramos. Su salida apunta a la dirección de la memoria de instrucciones donde se aloja la próxima instrucción a ejecutar.

Durante el flanco indicado en el cuadro, pone en su salida el valor que se encuentra en la entrada, siempre que no se bloquee la actualización de su salida (por hazard o step). Si a la señal del clock se encuentra activado el reset, entonces pone su salida a cero.

El PC funciona por flanco de bajada para darle tiempo a los circuitos de cambio de contador de programa (manejo de saltos) a actualizarse y estar en valores estables. Esto evita ingresar una burbuja en caso de, por ejemplo, un jump (ya que si bien hay una demora en reconocer la instrucción, la actualización del pc se realiza en el segundo semiciclo).

PC_Mux

Entradas

- i_sel_jump: Señal de selección para salto (1: salto, 0: secuencial)
- i_next_pc [31:0]: Dirección del próximo PC secuencial (PC + 4)
- i_jump_pc [31:0]: Dirección de destino del salto

Salidas

- o_pc [31:0]: Dirección del PC seleccionada

Sincronismo

Combinacional (sin clock)

Descripción

Este multiplexor de 2 entradas determina la próxima dirección del Program Counter. Selecciona entre la ejecución secuencial normal (PC + 4) o una dirección de salto proporcionada por las etapas posteriores del pipeline. La señal de control proviene de la lógica de control de saltos y branches, permitiendo implementar tanto saltos incondicionales como condicionales en el procesador MIPS.

Adder (PC+4)

Entradas

- i_rst: Señal de reset
- i_operand_1 [31:0]: Primer operando (constante 4)
- i_operand_2 [31:0]: Segundo operando (PC actual)

Salidas

- o_result [31:0]: Resultado de la suma (PC + 4)

Sincronismo

Combinacional (sin clock)

Descripción

Este sumador calcula PC + 4, que representa la dirección de la próxima instrucción en ejecución secuencial. El valor 4 corresponde al tamaño de una instrucción en bytes en la arquitectura MIPS (todas las instrucciones son de 32 bits = 4 bytes). Este valor es fundamental para el funcionamiento normal del pipeline, ya que permite que el procesador avance secuencialmente a través del programa. Si hay un reset activo, la salida se pone en cero.

Instruction_Memory

Entradas

- i_clk: Señal de reloj
- i_rst: Señal de reset
- i_step: Señal de step para ejecución paso a paso
- i_addr [31:0]: Dirección de lectura de instrucción (desde PC)
- i_dbg_addr [31:0]: Dirección para debug/escritura
- i_dbg_inst [31:0]: Instrucción a escribir en modo debug
- i_dbg_wr_en: Habilitación de escritura en modo debug

Salidas

- o_data [31:0]: Instrucción leída desde la memoria
- o_dbg_data [31:0]: Datos leídos en modo debug

Sincronismo

Lectura en flanco de bajada (negedge), escritura debug en flanco de subida de i_dbg_wr_en, lectura debug combinacional

Descripción

Este módulo implementa la memoria de instrucciones del procesador MIPS. Almacena las instrucciones del programa en un arreglo de memoria organizado por bytes. La lectura normal de instrucciones se realiza usando la dirección proporcionada por el PC, devolviendo una instrucción completa de 32 bits formada por la concatenación de 4 bytes consecutivos. Incluye funcionalidad de debug que permite escribir instrucciones en la memoria y leer desde direcciones arbitrarias, facilitando la carga de programas y el debugging del procesador. La memoria está parametrizada para permitir diferentes tamaños según las necesidades del sistema.

Cycle_Counter

Entradas

- i_clk: Señal de reloj
- i_rst: Señal de reset
- i_step: Señal de step para ejecución paso a paso

Salidas

- o_count [31:0]: Contador de ciclos transcurridos

Sincronismo

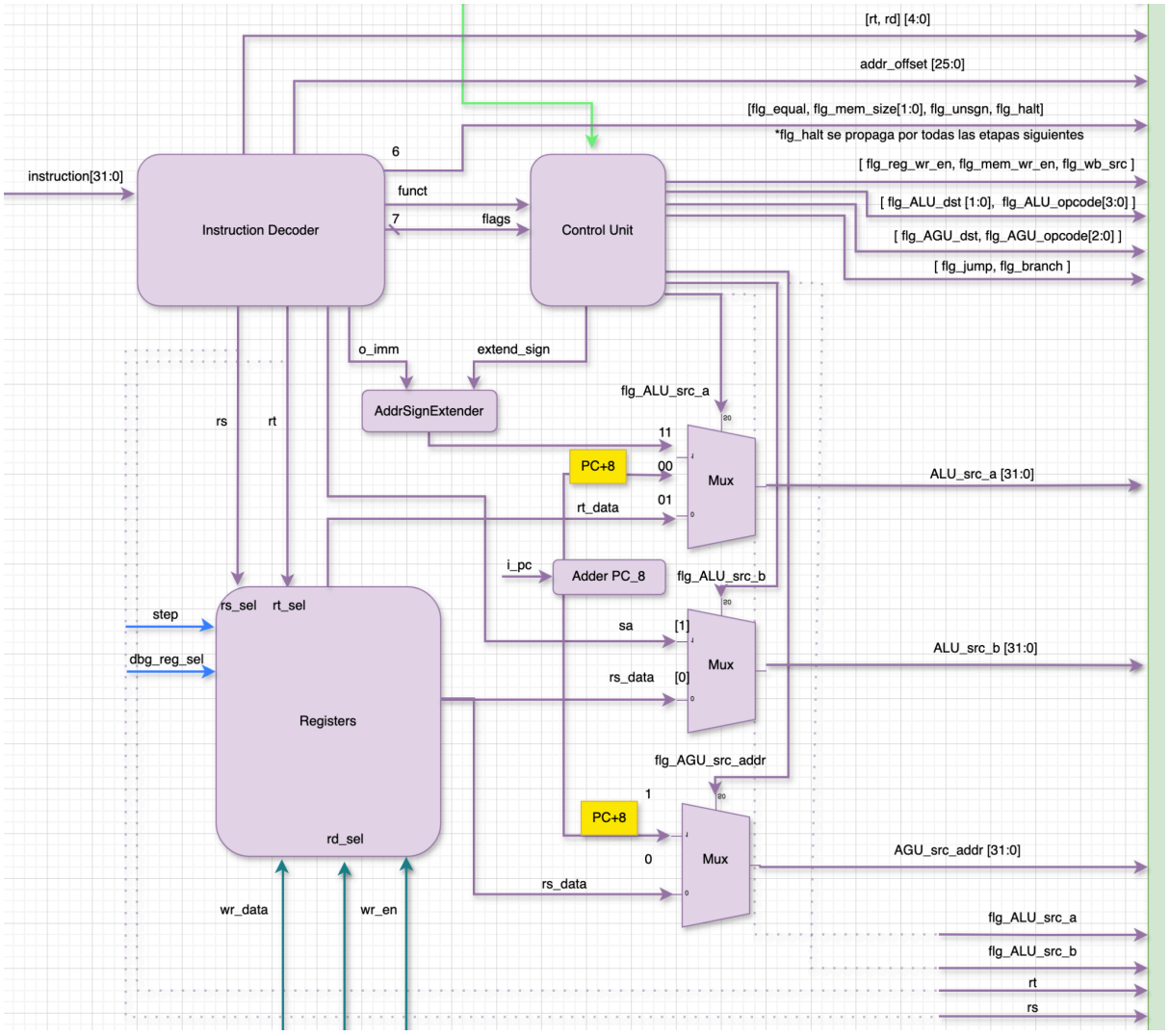
Flanco de subida (posedge)

Descripción

Este módulo implementa un contador de ciclos para estadísticas y propósitos de debug. Incrementa su valor en cada ciclo de reloj cuando está habilitado el step y no hay reset activo. Proporciona información valiosa sobre el rendimiento del procesador y permite rastrear cuántos ciclos han transcurrido durante la ejecución de un programa. Es especialmente útil para análisis de performance y debugging, ya que permite correlacionar eventos específicos con ciclos de reloj particulares. El contador se reinicia a cero cuando se activa la señal de reset.

Etapa de Instruction Decode (ID)

Este apartado describe los módulos utilizados en la etapa de decodificación de instrucciones (ID) del procesador MIPS implementado en el módulo `Top_Instruction_Decode`.



Módulos

Instruction_Decoder

Entradas

- i_instr [31:0]: Instrucción de 32 bits a decodificar

Salidas

- o_funct [5:0]: Código de función para instrucciones tipo R o bits inferiores del opcode para instrucciones tipo I
- o_rs [4:0]: Campo RS de la instrucción (registro fuente)
- o_rt [4:0]: Campo RT de la instrucción (registro destino/fuente)
- o_rd [4:0]: Campo RD de la instrucción (registro destino)
- o_sa [31:0]: Campo SA de la instrucción (shift amount)
- o_imm [15:0]: Campo inmediato de la instrucción
- o_addr_offset [25:0]: Offset de dirección para instrucciones de salto
- o_flg_pc_modify: Bandera que indica si la instrucción modifica el PC (jump/branch)
- o_flg_link_ret: Bandera que indica si se debe guardar la dirección de retorno
- o_flg_addr_type [1:0]: Tipo de dirección (00: registro, 01: offset de 26 bits, 10: offset de 16 bits)
- o_flg_equal: Bandera para comparaciones de igualdad
- o_flg_inmediate: Bandera que indica si es una instrucción tipo I
- o_flg_mem_op: Bandera que indica si es una operación de memoria
- o_flg_mem_type: Tipo de operación de memoria (0: load, 1: store)
- o_flg_mem_size [1:0]: Tamaño de datos (00: byte, 01: halfword, 11: word)
- o_flg_unsign: Bandera para operaciones sin signo
- o_flg_halt: Bandera de instrucción de parada

Sincronismo

Combinacional (sin clock)

Descripción

Este módulo se encarga de decodificar la instrucción de 32 bits de entrada y extraer todos los campos relevantes según el tipo de instrucción (R, I, o J). Además, genera las señales de control necesarias que indican el tipo de operación a realizar. El decodificador analiza el opcode y los campos de función para determinar las características de la instrucción y configurar apropiadamente las señales de control para las siguientes etapas del pipeline.

Control_Unit

Entradas

- i_funct [5:0]: Código de función de la instrucción
- i_flg_pc_modify: Bandera de modificación del PC
- i_flg_link_ret: Bandera de guardar dirección de retorno
- i_flg_addr_type [1:0]: Tipo de dirección (cómo debe obtenerse)
- i_flg_inmediate: Bandera de instrucción inmediata
- i_flg_mem_op: Bandera de operación de memoria (load/store)
- i_flg_mem_type: Tipo de operación de memoria
- i_hazard_detected: Señal de detección de riesgo
- i_flg_halt: Bandera de parada

Salidas

- o_flg_ALU_src_a [1:0]: Selector de fuente A de la ALU (00: PC+8, 01: rt_data, 11: sign_ext)
- o_flg_ALU_src_b: Selector de fuente B de la ALU (0: rs_data, 1: sa)
- o_flg_ALU_dst [1:0]: Selector de destino de la ALU (00: RT, 01: RD, 11: $ra)
- o_ALU_opcode [3:0]: Código de operación para la ALU
- o_flg_AGU_src_addr: Selector de dirección para AGU (0: rs_data, 1: PC+8)
- o_flg_AGU_opcode [2:0]: Código de operación para AGU
- o_flg_jump: Bandera de instrucción de salto
- o_flg_branch: Bandera de instrucción de salto condicional
- o_flg_reg_wr_en: Habilitación de escritura en registros
- o_flg_mem_wr_en: Habilitación de escritura en memoria
- o_flg_wb_src: Selector de fuente para writeback (0: memoria, 1: ALU)
- o_flg_jmp_trg_reg: Bandera de salto a través de registro
- o_extend_sign [1:0]: Modo de extensión de signo (00: con signo, 01: ceros arriba, 10: ceros abajo)

Sincronismo

Combinacional (sin clock)

Descripción

La unidad de control genera todas las señales de control necesarias para las etapas EX, MEM y WB basándose en el tipo de instrucción decodificada. Determina la configuración de los multiplexores, las operaciones de la ALU y AGU, y las habilitaciones de escritura.

Registers

Entradas

- i_clk: Señal de reloj
- i_rst: Señal de reset
- i_rs_sel [4:0]: Selector del registro RS
- i_rt_sel [4:0]: Selector del registro RT
- i_rd_sel [4:0]: Selector del registro RD (para escritura)
- i_wr_en: Habilitación de escritura
- i_wr_data [31:0]: Datos a escribir
- i_dbg_reg_sel [4:0]: Selector de registro para debug
- i_step: Señal de step para debug

Salidas

- o_rs_data [31:0]: Datos del registro RS
- o_rt_data [31:0]: Datos del registro RT
- o_dbg_reg_data [31:0]: Datos del registro seleccionado para debug

Sincronismo

Escritura en flanco de bajada (negedge), lectura combinacional

Descripción

Este módulo implementa el banco de 32 registros de propósito general del procesador MIPS. Incluye el registro $zero (siempre en 0) y 31 registros adicionales. La escritura se realiza en el flanco de bajada del reloj para evitar conflictos con la lectura, mientras que la lectura es asíncrona. El módulo también incluye funcionalidad de debug que permite leer cualquier registro independientemente de las operaciones normales del procesador.

Sign_Extender

Entradas

- i_inmediate [15:0]: Valor inmediato de 16 bits
- i_mode [1:0]: Modo de extensión (00: con signo, 01: ceros arriba, 10: ceros abajo)

Salidas

- o_result [31:0]: Resultado extendido a 32 bits

Sincronismo

Combinacional (sin clock)

Descripción

Este módulo extiende valores inmediatos de 16 bits a 32 bits según el modo especificado. Puede realizar extensión de signo en complemento a 2 (replicando el bit más significativo), extensión con ceros en la parte superior (para operaciones lógicas), o colocar el inmediato en la parte superior con ceros en la inferior (para la instrucción LUI). Es esencial para el manejo correcto de constantes en instrucciones tipo I.

Adder (PC_8)

Entradas

- i_rst: Señal de reset
- i_operand_1 [31:0]: Primer operando (PC actual)
- i_operand_2 [31:0]: Segundo operando (constante 8)

Salidas

- o_result [31:0]: Resultado de la suma (PC + 8)

Sincronismo

Combinacional (sin clock)

Descripción

Este sumador calcula PC + 8, que representa la dirección de la instrucción siguiente a la siguiente (dos instrucciones adelante). Este valor es necesario para instrucciones de salto con enlace (JAL, JALR) donde se debe guardar la dirección de retorno. El valor 8 corresponde a dos direcciones de instrucción (cada instrucción ocupa 4 bytes).

Mux_4 (ALU_SRC_A)

Entradas

- i_sel [1:0]: Señal de selección
- i_a [31:0]: Entrada A (PC + 8)
- i_b [31:0]: Entrada B (rt_data)
- i_c [31:0]: Entrada C (no conectado)
- i_d [31:0]: Entrada D (sign_ext_result)

Salidas

- o_result [31:0]: Salida seleccionada

Sincronismo

Combinacional (sin clock)

Descripción

Multiplexor de 4 entradas que selecciona la fuente A para la ALU. Puede elegir entre PC+8, datos del registro RT (para ciertos shifts), o el resultado del extensor de signo (para operaciones inmediatas). La selección se basa en el tipo de instrucción y es controlada por la unidad de control.

Mux_2 (ALU_SRC_B)

Entradas

- i_sel: Señal de selección
- i_a [31:0]: Entrada A (rs_data)
- i_b [31:0]: Entrada B (sa - shift amount)

Salidas

- o_result [31:0]: Salida seleccionada

Sincronismo

Combinacional (sin clock)

Descripción

Multiplexor de 2 entradas que selecciona la fuente B para la ALU. Puede elegir entre los datos del registro RS (para la mayoría de operaciones) o el campo SA (shift amount) de la instrucción (para operaciones de desplazamiento con cantidad fija). La selección depende del tipo de operación aritmética o de desplazamiento.

Mux_2 (AGU_SRC_ADDR)

Entradas

- i_sel: Señal de selección
- i_a [31:0]: Entrada A (rs_data)
- i_b [31:0]: Entrada B (PC + 8)

Salidas

- o_result [31:0]: Salida seleccionada

Sincronismo

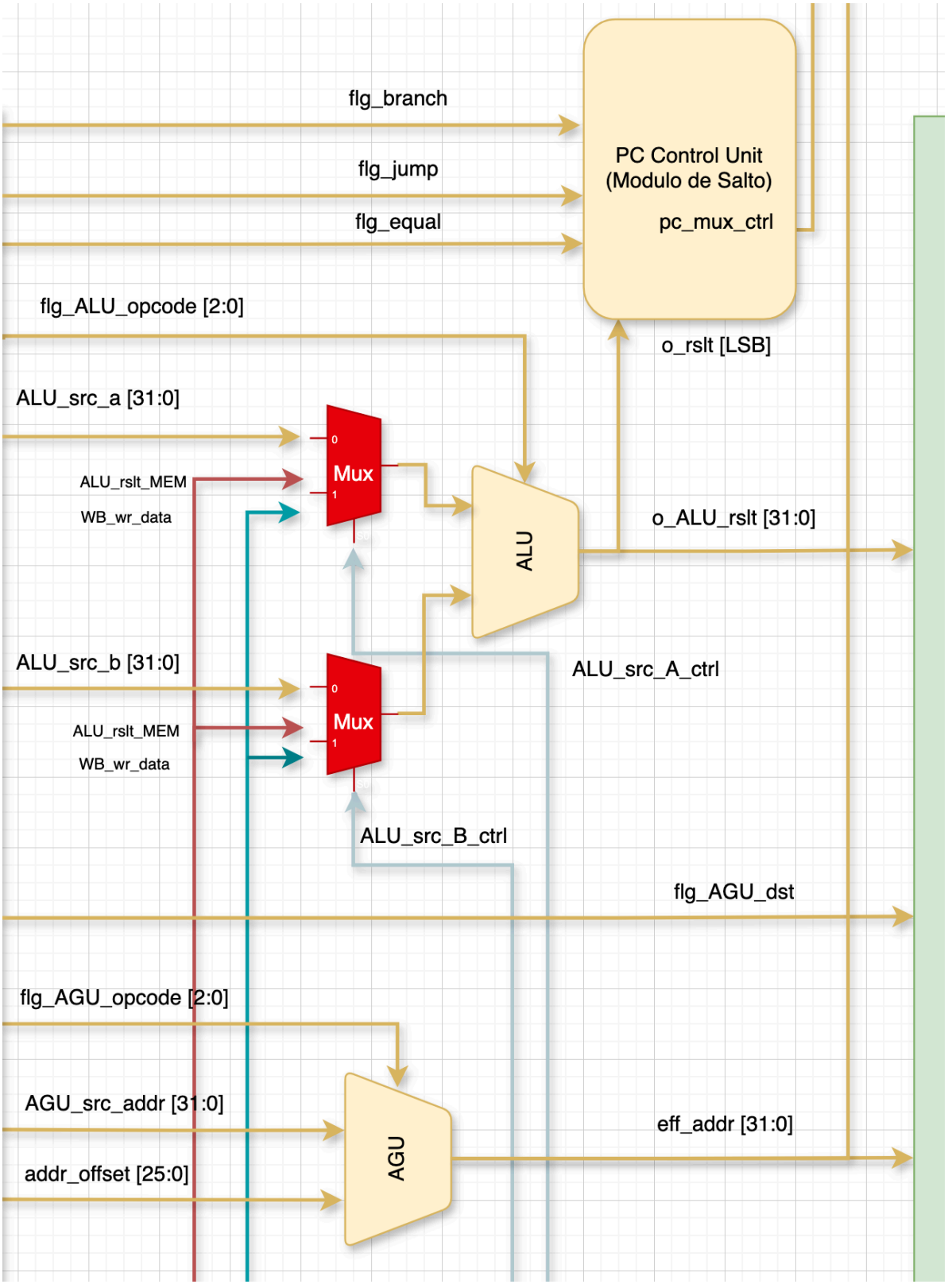
Combinacional (sin clock)

Descripción

Multiplexor que selecciona la dirección base para la Unidad de Generación de Direcciones (AGU). Puede elegir entre el contenido del registro RS (para direcciones base de memoria) o PC+8 (para saltos relativos al PC). Esta selección permite diferentes modos de direccionamiento para operaciones de memoria y saltos.

Etapa Execution (EX)

Este apartado describe los módulos utilizados en la etapa de ejecución (EX) del procesador MIPS implementado en el módulo `Top_Execute`.



Módulos

Mux_4 (ALU_mux_A)

Entradas

- i_sel [1:0]: Señal de selección del forwarding unit
- i_a [31:0]: Entrada A (dato original desde ID)
- i_b [31:0]: Entrada B (resultado ALU desde MEM)
- i_c [31:0]: Entrada C (dato desde WB)
- i_d [31:0]: Entrada D (no conectado)

Salidas

- o_result [31:0]: Salida seleccionada para entrada A de la ALU

Sincronismo

Combinacional (sin clock)

Descripción

Multiplexor de 4 entradas que implementa el forwarding para el operando A de la ALU. Selecciona entre el dato original proveniente de la etapa ID, el resultado de la ALU desde la etapa MEM (para resolver hazards RAW), o el dato desde la etapa WB. Esta funcionalidad es esencial para resolver dependencias de datos en el pipeline sin necesidad de insertar stalls, mejorando significativamente el rendimiento del procesador.

Mux_4 (ALU_mux_B)

Entradas

- i_sel [1:0]: Señal de selección del forwarding unit
- i_a [31:0]: Entrada A (dato original desde ID)
- i_b [31:0]: Entrada B (resultado ALU desde MEM)
- i_c [31:0]: Entrada C (dato desde WB)
- i_d [31:0]: Entrada D (no conectado)

Salidas

- o_result [31:0]: Salida seleccionada para entrada B de la ALU

Sincronismo

Combinacional (sin clock)

Descripción

Multiplexor de 4 entradas que implementa el forwarding para el operando B de la ALU. Funciona de manera similar al ALU_mux_A, pero para el segundo operando de la ALU. Permite resolver hazards de datos seleccionando el valor más actualizado disponible en el pipeline, ya sea desde la etapa MEM o WB, evitando la necesidad de stalls y manteniendo el throughput del procesador.

ALU

Entradas

- i_opcode [3:0]: Código de operación ALU
- i_op_A [31:0]: Operando A
- i_op_B [31:0]: Operando B

Salidas

- o_rslt [31:0]: Resultado de la operación
- o_zero: Bandera de resultado cero
- o_carry: Bandera de acarreo
- o_ovfl_exception: Bandera de overflow

Sincronismo

Combinacional (sin clock)

Descripción

La Unidad Aritmético-Lógica (ALU) es el corazón computacional del procesador. Implementa todas las operaciones aritméticas y lógicas del conjunto de instrucciones MIPS. Las operaciones incluyen:

- **Desplazamientos**: LSR (0000), LSL (0001), ASR (0010)
- **Aritmética**: UADD (0100), USUB (0101), ADD (1100) con detección de overflow
- **Lógicas**: AND (0110), OR (0111), XOR (1000), NOR (1001)
- **Comparaciones**: GT (1010), CMP (1011)
- **Passthrough**: PASS (0011) para transferencias directas

La ALU genera banderas de estado que son utilizadas por instrucciones de branch condicional y para la detección de excepciones. El resultado es usado tanto para operaciones de memoria como para escritura en registros.

AGU (Address Generation Unit)

Entradas

- i_opcode [2:0]: Código de operación AGU
- i_addr [31:0]: Dirección base
- i_offset [25:0]: Offset para cálculo de dirección

Salidas

- o_eff_addr [31:0]: Dirección efectiva calculada
- o_addr_exception [1:0]: Banderas de excepción de alineamiento

Sincronismo

Combinacional (sin clock)

Descripción

La Unidad de Generación de Direcciones (AGU) calcula las direcciones efectivas para operaciones de memoria y saltos. Implementa diferentes modos de direccionamiento:

- **Directo** (000): Dirección desde registro RS
- **Base + Offset** (001): Dirección base + offset de 16 bits con signo
- **PC-relativo con shift** (010): Offset de 16 bits con signo, desplazado 2 bits, sumado a la base
- **Directo con reemplazo** (011): Reemplaza los 28 bits inferiores del PC con offset de 26 bits desplazado

También detecta excepciones de alineamiento verificando que las direcciones sean múltiplos de 4 para accesos a word. Es fundamental para la correcta implementación de instrucciones de memoria (load/store) y saltos.

PC_Control_Unit

Entradas

- i_flg_jump: Bandera de instrucción de salto
- i_flg_branch: Bandera de instrucción de branch
- i_flg_equal: Bandera de tipo de comparación (1: igual, 0: desigual)
- i_rslt_lsb: Bit menos significativo del resultado ALU

Salidas

- o_pc_mux_ctrl: Señal de control para el multiplexor del PC

Sincronismo

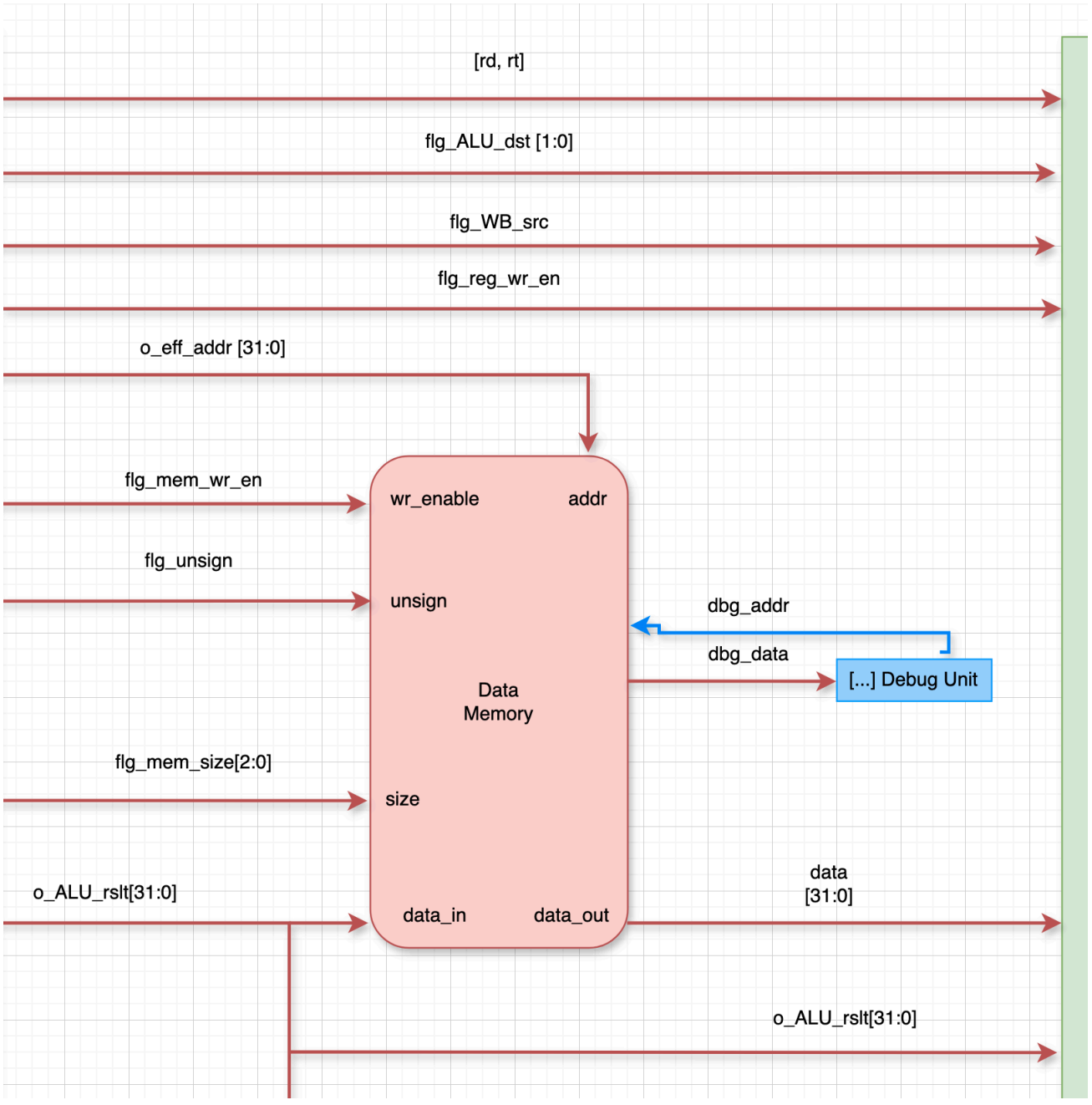
Combinacional (sin clock)

Descripción

Esta unidad determina si se debe realizar un salto o continuar con la ejecución secuencial. Implementa la lógica de decisión para branches condicionales y saltos incondicionales. Para branches, evalúa la condición comparando el tipo esperado (igual/desigual) con el resultado real de la comparación (indicado por el LSB del resultado ALU). La salida controla el multiplexor en la etapa IF que selecciona entre PC+4 (secuencial) o la dirección de salto calculada por la AGU. Es crítica para el control de flujo del programa y la implementación correcta de estructuras de control como loops e if-statements.

Etapa de Memory Access (MA)

Este documento describe los módulos utilizados en la etapa de acceso a memoria (MEM) del procesador MIPS implementado en el módulo `Top_Memory_Access`.



Módulos

Data_Memory

Entradas

- i_clk: Señal de reloj
- i_rst: Señal de reset
- i_step: Señal de step para ejecución paso a paso
- i_write_en: Habilitación de escritura en memoria
- i_size [1:0]: Tamaño del dato (00: byte, 01: halfword, 11: word)
- i_unsigned: Bandera para operaciones sin signo
- i_addr [31:0]: Dirección de memoria (dirección efectiva desde AGU)
- i_data [31:0]: Datos a escribir (resultado ALU para stores)
- i_dbg_addr [31:0]: Dirección para debug/lectura externa

Salidas

- o_data [31:0]: Datos leídos desde la memoria
- o_dbg_data [31:0]: Datos leídos en modo debug

Sincronismo

Lectura/escritura en flanco de bajada (negedge), lectura debug combinacional

Descripción

Este módulo implementa la memoria de datos del procesador MIPS, donde se almacenan las variables y estructuras de datos del programa. La memoria está organizada por bytes (8 bits por celda) y soporta accesos de diferentes tamaños:

- **Byte** (00): Acceso a 8 bits con extensión de signo opcional
- **Halfword** (01): Acceso a 16 bits (2 bytes consecutivos) con extensión de signo opcional
- **Word** (11): Acceso a 32 bits (4 bytes consecutivos) sin extensión de signo

Operaciones de escritura: Se realizan cuando está habilitada la señal i_write_en y i_step está activo. Los datos provenientes del resultado de la ALU se escriben en la dirección calculada por la AGU según el tamaño especificado.

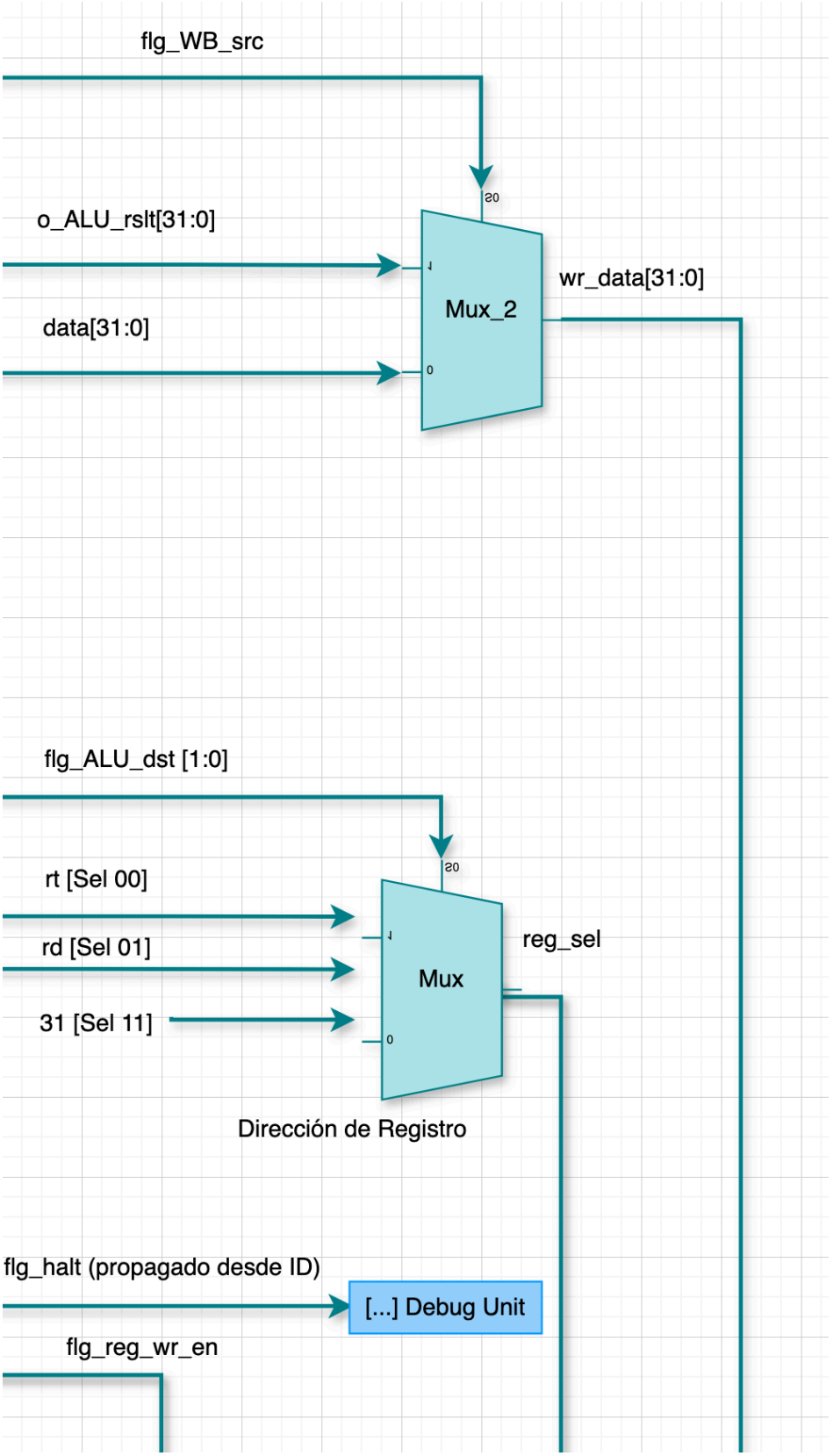
Operaciones de lectura: Se realizan cuando no hay escritura activa. Los datos se leen desde la dirección especificada y se extienden según el tamaño y el tipo (con o sin signo). Para bytes y halfwords, la extensión de signo se aplica según la bandera i_unsigned.

Funcionalidad de debug: Permite lectura asíncrona desde cualquier dirección para propósitos de debugging y monitoreo del estado de la memoria, independientemente de las operaciones normales del procesador.

La memoria utiliza el flanco de bajada del reloj para mantener consistencia temporal con el resto del pipeline y evitar condiciones de carrera. Es fundamental para la implementación de instrucciones load/store del conjunto MIPS.

Etapa de Writeback (WB)

Este apartado describe los módulos utilizados en la etapa de escritura (WB) del procesador MIPS implementado en el módulo `Top_Writeback`.



Módulos

Mux_2 (WB_Data_Source)

Entradas

- i_sel: Señal de selección de fuente de datos (i_flg_wb_src)
- i_a [31:0]: Entrada A (datos desde memoria - i_data)
- i_b [31:0]: Entrada B (resultado ALU - i_ALU_rslt)

Salidas

- o_result [31:0]: Datos seleccionados para escritura en registro

Sincronismo

Combinacional (sin clock)

Descripción

Este multiplexor de 2 entradas selecciona la fuente de los datos que se escribirán en el registro destino. Puede elegir entre:

- **Datos de memoria (0)**: Para instrucciones load (LW, LH, LB, etc.) donde el valor a escribir proviene de la memoria de datos
- **Resultado ALU (1)**: Para instrucciones aritméticas, lógicas y de comparación donde el resultado de la ALU debe escribirse en el registro

Esta selección es fundamental para distinguir entre instrucciones que modifican registros con resultados computacionales versus aquellas que cargan datos desde memoria. La señal de control i_flg_wb_src es generada por la unidad de control en la etapa ID.

Mux_4 (Register_Destination_Selector)

Entradas

- i_sel [1:0]: Señal de selección de registro destino (i_flg_ALU_dst)
- i_a [4:0]: Entrada A (registro RT - i_rt)
- i_b [4:0]: Entrada B (registro RD - i_rd)
- i_c [4:0]: Entrada C (no conectado - 5'b0)
- i_d [4:0]: Entrada D (registro $ra - 5'd31)

Salidas

- o_result [4:0]: Dirección del registro destino seleccionado

Sincronismo

Combinacional (sin clock)

Descripción

Este multiplexor de 4 entradas determina cuál registro será el destino de la operación de escritura. Las opciones son:

- **RT (00)**: Para instrucciones tipo I (inmediatas) como ADDI, ANDI, ORI, LW, etc., donde el registro destino es RT
- **RD (01):** Para instrucciones tipo R donde el registro destino es RD (ADD, SUB, AND, OR, etc.)
- **No usado (10)**: Entrada no conectada para posibles extensiones futuras
- **$ra (11)**: Para instrucciones de salto con enlace (JAL, JALR) donde se debe guardar la dirección de retorno en el registro $ra (registro 31)

Esta funcionalidad es esencial para el correcto direccionamiento de la escritura en el banco de registros, asegurando que cada tipo de instrucción escriba en el registro apropiado según las convenciones de la arquitectura MIPS. La señal de control i_flg_ALU_dst es generada por la unidad de control basándose en el tipo de instrucción decodificada.

Características Comunes de Todos los Registros Pipeline

Sincronismo

- Clock: Todos operan en flanco de subida (posedge)
- Reset: Reset síncrono que limpia todas las salidas
- Step Control: Actualizan solo cuando `i_step` está habilitado

Control de Pipeline

- Stall Support: Mantienen valores cuando no deben actualizar
- Bubble Insertion: Pueden insertar NOPs durante hazards
- Temporal Isolation: Aíslan temporalmente las etapas del pipeline

Flujo de Información

La información fluye secuencialmente:

IF → Reg_IF_ID → ID → Reg_ID_EX → EX → Reg_EX_MA → MEM → Reg_MA_WB → WB

Además, existen rutas de retroalimentación:

- WB → ID: Para escritura en banco de registros
- EX → IF: Para control de saltos
- MEM/WB → EX: Para forwarding

Esta arquitectura pipeline permite la ejecución superpuesta de hasta 5 instrucciones simultáneamente, mejorando significativamente el throughput del procesador MIPS.

Comunicación Serial

Este apartado describe los módulos utilizados en la interfaz UART implementada en el módulo `UART_TOP`.

UART_tick_gen

Entradas

- i_clk: Señal de reloj del sistema (50 MHz)
- i_rst: Señal de reset

Salidas

- o_tick: Pulso de tick para sincronización UART (señal con pulsos para sobremuestrear 16 veces la tasa de envío de símbolos por UART de 9600bps)

Sincronismo

Flanco de subida (posedge)

Descripción

Generador de ticks para la sincronización del protocolo UART. Divide la frecuencia del reloj del sistema para generar pulsos a una frecuencia 16 veces mayor que el baud rate configurado (típicamente 9600 bps). Esta frecuencia permite el sobremuestreo necesario para la recepción confiable de datos seriales.

Parámetros:

- CLOCK: Frecuencia del reloj del sistema (50,000,000 Hz)
- BAUD_RATE: Velocidad de transmisión (9,600 bps)
- TICK_RATE: Divisor calculado = CLOCK/(BAUD_RATE×16)

El contador interno se incrementa en cada ciclo de reloj y genera un tick cuando alcanza TICK_RATE-1, luego se reinicia. Este tick sincroniza tanto la transmisión como la recepción de datos UART.

UART_tx

Entradas

- i_clk: Señal de reloj del sistema
- i_rst: Señal de reset
- i_ready: Señal para iniciar transmisión
- i_data [7:0]: Datos a transmitir (8 bits)

Salidas

- o_tx: Línea serial de transmisión
- o_done: Bandera que indica transmisión completada

Sincronismo

Flanco de subida (posedge) con divisor de frecuencia interno

Descripción

Módulo transmisor UART que implementa el protocolo serie estándar. Convierte datos paralelos de 8 bits en transmisión serie con el formato:

- 1 bit de start (0 lógico)
- 8 bits de datos (LSB primero)
- 1 bit de stop (1 lógico)

Estados

- IDLE: Estado de reposo, línea TX en alto, esperando señal i_ready
- TRANSMIT: Transmitiendo bits secuencialmente según el protocolo

Funcionamiento

1. En estado IDLE, cuando i_ready se activa, carga los datos en un registro de desplazamiento
2. Usa un divisor de frecuencia interno para generar el baud rate correcto
3. Transmite bit de start, seguido de los 8 bits de datos y bit de stop
4. Genera señal o_done al completar la transmisión
5. Retorna a estado IDLE

Parámetros configurables

- CLK_FREQ: Frecuencia del reloj
- BAUD_RATE: Velocidad de transmisión
- PAYLOAD_SIZE: Tamaño de datos (8 bits)

UART_rx

Entradas

- i_clk: Señal de reloj del sistema
- i_rst: Señal de reset
- i_rx: Línea serial de recepción
- i_tick: Pulso de sincronización desde UART_tick_gen

Salidas

- o_ready: Bandera que indica dato recibido
- o_data [7:0]: Datos recibidos (8 bits)

Sincronismo

Flanco de subida (posedge) sincronizado con i_tick

Descripción

Módulo receptor UART que decodifica la transmisión serie en datos paralelos. Implementa sobremuestreo (16x) para detectar y recibir datos de manera confiable.

Estados

- IDLE: Esperando detección de bit de start
- START: Verificando bit de start en el punto medio del periodo
- DATA: Recibiendo los 8 bits de datos
- STOP: Verificando bit de stop

Funcionamiento

1. En IDLE, monitorea línea RX esperando transición de 1 a 0 (bit de start)
2. Usa sobremuestreo 16x para localizar el centro de cada bit transmitido
3. Muestrea cada bit en el punto medio (muestra 7 de 16) para máxima confiabilidad
4. Recibe los 8 bits de datos secuencialmente (LSB primero)
5. Verifica bit de stop y genera señal o_ready con datos válidos
6. Retorna a IDLE para próxima recepción

Características

- Sobremuestreo 16x para sincronización robusta
- Detección automática de frame de datos
- Tolerancia a variaciones de timing
- Manejo de errores de frame implícito

Parámetros

- PAYLOAD_SIZE: Tamaño de datos (8 bits)
- OVERSAMPLING: Factor de sobremuestreo (16x)

UART_TOP (Módulo Principal)

Entradas

- i_clk: Señal de reloj del sistema
- i_rst_rx: Reset específico para receptor
- i_rst_tx: Reset específico para transmisor
- i_rx: Línea serial de entrada
- i_data [7:0]: Datos a transmitir
- i_send_data: Comando para iniciar transmisión

Salidas

- o_data [7:0]: Datos recibidos
- o_tx: Línea serial de salida
- o_flg_data_recieved: Bandera de dato recibido
- o_flg_data_sent: Bandera de dato transmitido

Sincronismo

Según submódulos componentes

Descripción

Módulo top-level que integra todos los componentes UART en una interfaz completa de comunicación serial bidireccional. Coordina el funcionamiento de:

- Generador de ticks: Proporciona sincronización base
- Transmisor: Maneja envío de datos

- Receptor: Maneja recepción de datos

Características del sistema

- Comunicación full-duplex (simultánea TX/RX)
- Baud rate configurable (9600 bps por defecto)
- Protocolo estándar: 8 bits datos, 1 start, 1 stop, sin paridad
- Resets independientes para TX y RX
- Señales de estado para handshaking

Aplicaciones en el sistema

- Interfaz de debug con PC/terminal
- Carga de programas en memoria de instrucciones
- Monitoreo de registros y memoria
- Control de ejecución (step, run, reset)
- Comunicación con Debug_Unit para control del procesador MIPS

Este módulo es fundamental para la interacción usuario-sistema, permitiendo control completo del procesador desde una interfaz externa.

Integración de Etapas en el Pipeline

Este apartado describe los módulos utilizados en el sistema completo del procesador MIPS implementado en el módulo `TOP_TOP`.

Generador de Clock (clk_wiz_0)

Entradas

- clk_in1: Señal de reloj de entrada del sistema
- reset: Señal de reset (conectado a 0)

Salidas

- clk_out1: Señal de reloj generada para el sistema
- locked: Señal que indica estabilidad del clock

Sincronismo

IP Core de Vivado

Descripción

Generador de reloj basado en IP Core de Vivado que convierte la frecuencia de entrada del sistema (típicamente 100MHz) a la frecuencia de trabajo del procesador (50MHz). Proporciona un reloj estable y sincronizado para todo el sistema. Es fundamental para el correcto funcionamiento temporal de todo el pipeline del procesador.

Top_Instruction_Fetch (IF)

Entradas

- i_clk: Señal de reloj del sistema
- i_rst: Reset desde debug unit
- i_pc_mux_ctrl: Control de multiplexor PC desde EX
- i_eff_addr: Dirección efectiva desde EX
- i_inst_mem_wr_en: Habilitación escritura instrucciones desde debug
- i_inst_mem_addr: Dirección para escribir instrucciones desde debug
- i_inst_mem_data: Datos de instrucción desde debug
- i_hazard_detected: Señal de hazard desde HU
- i_step: Señal de step desde debug unit

Salidas

- o_pc: Program Counter actual
- o_instr: Instrucción obtenida
- o_cycle_count: Contador de ciclos
- o_dbg_data: Datos para debug

Sincronismo

Según componentes internos

Descripción

Etapa completa de búsqueda de instrucciones que incluye PC, memoria de instrucciones, sumador PC+4 y multiplexor de PC. Se encarga de obtener la próxima instrucción a ejecutar y calcular la siguiente dirección del PC.

Reg_IF_ID

Entradas

- i_clk: Señal de reloj
- i_rst: Señal de reset
- i_step: Señal de step
- i_pc [31:0]: Program Counter desde IF
- i_instruction [31:0]: Instrucción desde IF
- i_hazard_detected: Señal de hazard detection

Salidas

- o_pc [31:0]: Program Counter registrado
- o_instruction [31:0]: Instrucción registrada

Sincronismo

Flanco de subida (posedge)

Descripción

Registro pipeline entre las etapas IF e ID. Almacena el PC y la instrucción obtenida en IF para pasarla a la etapa ID en el siguiente ciclo. Incluye lógica de stall para freezar el pipeline cuando se detectan hazards, manteniendo los mismos valores cuando hay detección de riesgos.

Top_Instruction_Decode (ID)

Entradas

- Señales de control, PC, instrucción, señales de writeback desde WB

Salidas

- Señales de control para EX, datos de registros, señales de debug

Sincronismo

Según componentes internos

Descripción

Etapa completa de decodificación que incluye decodificador de instrucciones, unidad de control, banco de registros, extensor de signo y multiplexores. Decodifica la instrucción y genera todas las señales de control necesarias para las etapas posteriores.

Reg_ID_EX

Entradas

- Todas las señales de control y datos desde ID

Salidas

- Todas las señales de control y datos registradas para EX

Sincronismo

Flanco de subida (posedge)

Descripción

Registro pipeline entre ID y EX. Es el registro más complejo del pipeline ya que debe transferir todas las señales de control generadas en ID (opcode ALU, opcode AGU, señales de writeback, datos de registros, etc.) hacia la etapa EX. Incluye lógica de stall para insertar NOPs cuando se detectan hazards.

Forwarding_Unit (FU)

Entradas

- i_rt_EX, i_rs_EX [4:0]: Registros fuente en EX
- i_flg_ALU_src_A [1:0], i_flg_ALU_src_B: Señales de tipo de fuente ALU
- i_rt_MEM, i_rd_MEM [4:0]: Registros destino en MEM
- i_flg_reg_wr_en_MEM: Habilitación escritura en MEM
- i_flg_ALU_dst_MEM [1:0]: Tipo destino ALU en MEM
- i_flg_reg_wr_en_WB: Habilitación escritura en WB
- i_rt_WB, i_rd_WB [4:0]: Registros destino en WB
- i_flg_ALU_dst_WB [1:0]: Tipo destino ALU en WB

Salidas

- o_ALU_src_a_ctrl [1:0]: Control forwarding fuente A ALU
- o_ALU_src_b_ctrl [1:0]: Control forwarding fuente B ALU

Sincronismo

Combinacional (sin clock)

Descripción

Unidad de forwarding que detecta dependencias RAW (Read After Write) y genera las señales de control para los multiplexores de forwarding en la etapa EX. Compara los registros fuente de la instrucción en EX con los registros destino de las instrucciones en MEM y WB. Si detecta una dependencia, habilitar el forwarding del valor más reciente, evitando stalls innecesarios y mejorando el rendimiento del pipeline.

Hazard_Unit (HU)

Entradas

- i_rd_EX, i_rt_EX [4:0]: Registros destino en EX
- i_rd_MA, i_rt_MA [4:0]: Registros destino en MA
- i_rd_WB, i_rt_WB [4:0]: Registros destino en WB
- i_reg_wr_EX, i_reg_wr_MA, i_reg_wr_WB: Habilitaciones de escritura
- i_flg_WB_src_EX: Fuente de writeback en EX
- i_flg_mem_op_EX: Operación de memoria en EX
- i_rs_ID, i_rt_ID [4:0]: Registros fuente en ID
- i_flg_jmp_trg_reg: Bandera de salto a registro
- i_flg_halt: Bandera de halt

Salidas

- o_hazard_detected: Señal de detección de hazard

Sincronismo

Combinacional (sin clock)

Descripción

Unidad de detección de riesgos que identifica situaciones donde el forwarding no puede resolver los riesgos de datos y control. Detecta principalmente:

- **Load-use hazards**: Cuando una instrucción load es seguida inmediatamente por una instrucción que usa el registro destino
- **Jump register hazards**: Cuando una instrucción JR/JALR depende de un registro que está siendo modificado
- **Halt condition**: Cuando se ejecuta una instrucción de parada

Cuando detecta un hazard, genera una señal que causa stalls en el pipeline hasta que la dependencia se resuelva.

Top_Execute (EX)

Entradas

- Señales de control desde ID_EX, datos forwarded

Salidas

- Resultado ALU, dirección efectiva, control PC

Sincronismo

Según componentes internos

Descripción

Etapa completa de ejecución que incluye ALU, AGU, multiplexores de forwarding y unidad de control de PC. Realiza las operaciones aritméticas/lógicas y calcula direcciones efectivas para memoria y saltos.

Reg_EX_MA y Reg_MA_WB

Descripción

Registros pipeline que transfieren señales de control y datos entre las etapas EX-MEM y MEM-WB respectivamente. Similar a los otros registros pipeline pero con conjuntos específicos de señales para cada etapa.

Top_Memory_Access (MA)

Descripción

Etapa de acceso a memoria que contiene la memoria de datos y maneja operaciones load/store.

Top_Writeback (WB)

Descripción

Etapa final que selecciona datos y registro destino para escribir en el banco de registros.

UART_TOP

Entradas

- i_clk: Señal de reloj
- i_rst_rx, i_rst_tx: Señales de reset para RX y TX
- i_rx: Línea serial de recepción
- i_data [7:0]: Datos a transmitir
- i_send_data: Señal para iniciar transmisión

Salidas

- o_data [7:0]: Datos recibidos
- o_tx: Línea serial de transmisión
- o_flg_data_recieved: Bandera de dato recibido
- o_flg_data_sent: Bandera de dato enviado

Sincronismo

Según protocolo UART

Descripción

Interfaz UART completa que incluye transmisor, receptor y generador de baudrate. Permite comunicación serial con dispositivos externos para carga de programas, debugging y monitoreo del procesador. Fundamental para la interfaz de usuario del sistema.

Debug_Unit (DU)

Entradas

- Señales UART, estado del procesador MIPS

Salidas

- Control del procesador, datos UART, estado debug

Sincronismo

Máquina de estados síncrona

Descripción

Unidad de debug que implementa una interfaz de control para el procesador. Permite:

- **Ejecución paso a paso**: Control fino del pipeline
- **Carga de programas**: Escribir instrucciones en memoria
- **Monitoreo**: Leer registros, memoria y estado del procesador
- **Control de reset**: Reinicializar el sistema

Actúa como puente entre la interfaz UART y el procesador, implementando un protocolo de comunicación que permite el debugging completo del sistema desde un terminal externo.