Universitat Politècnica de Catalunya

Arquitectura de computadors d'altes prestacions

Práctica 5

Procesador: segmentación lineal, camino de datos y control

Carlota Catot
Miguel Antunez

Grupo 6

Quatrimestre primavera 2020-2021

Índice

1. Pregunta 1

Diseñe el módulo LDD utilizando el menor número posible de puertas lógicas y comparadores. Justifique el diseño de forma sucinta y sistemática.

1.1. Módulo LDD

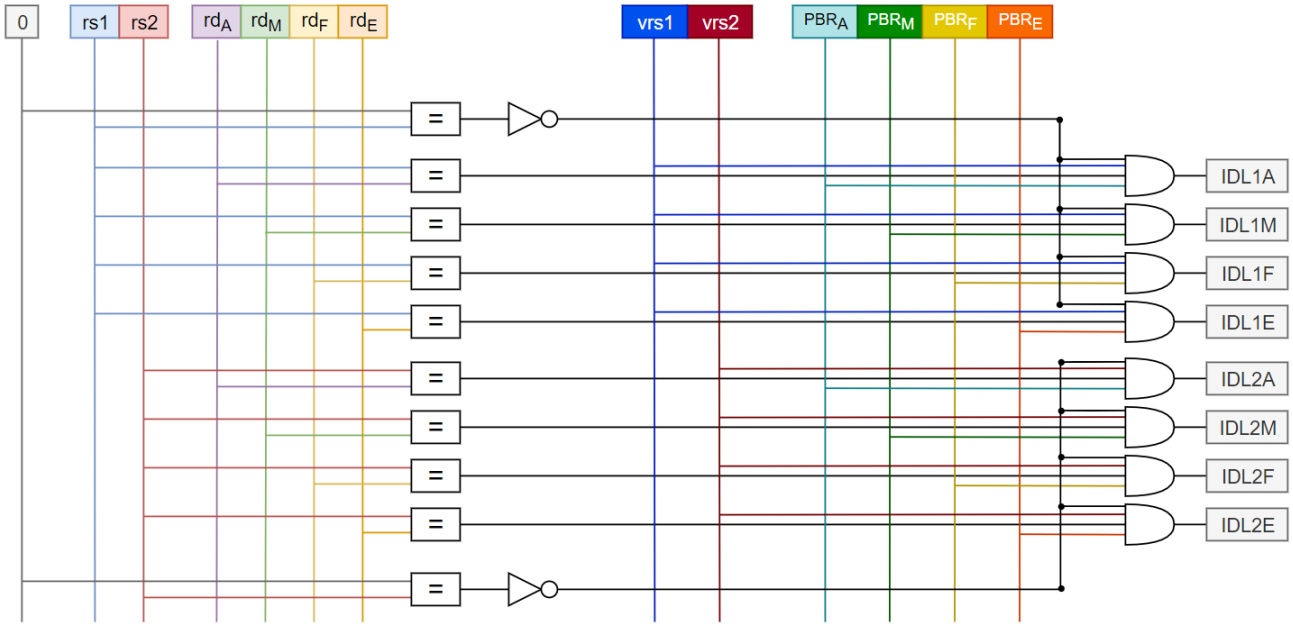


Figura 1: Esquema logico modulo LDD

1.2. Justificación

En este módulo nos encargaremos de la lógica de detección de dependencia de datos debido a registros (RS1 i RS2 son los dos registros a tener en cuenta), el resto de valores que tendremos en cuenta para ello son RD (identificadores de registro destino de las instrucciones, nos indican la etapa), vrs (para indicar la validez de los indentificadores rs) i por último PBR (indica si el registro tiene permiso de escritura en el banco de registros). Todo ello esta explicado más extensamente en la pagina 370 del enunciado.

Se debe hacer lo mismo para RS1 i para RS2 para poder llegar a los pines IDL1X i ILD2X respectivamente.

En primer lugar se comprueba el valor de RS con RD i se mira si son iguales. Seguidamente se cojera el valor de vrs (vrs1 cuando trabajamos con rs1 y vrs2 cuando trabajamos con rs2) y el valor de PBR (PBRa cuando se coje RDa, PBRm cuando se coje RDm, PBRf cuando se coje RDf y PBRe cuando se coje RDe), y todo ello se junta en una puerta logica AND para sacar el valor de IDLXY (el valor X dependen de si se ha cogido RS1 o RS2 y el valor Y depende de que RD se ha cogido).

Por otro lado se ha de comparar el valor de 0 con los registros de RS y se debera poner una puerta NOT ya que se tiene que comprovar en todas las puertas AND que RS no es el registro RS ya que si es el registro 0 no existe riesgo.

2. Pregunta 2

Diseñe el módulo LDRD utilizando el menor número posible de puertas lógicas, limitando el número de entradas a 2. Justifique el diseño de forma sucinta y sistemática.

2.1. Módulo LDRD

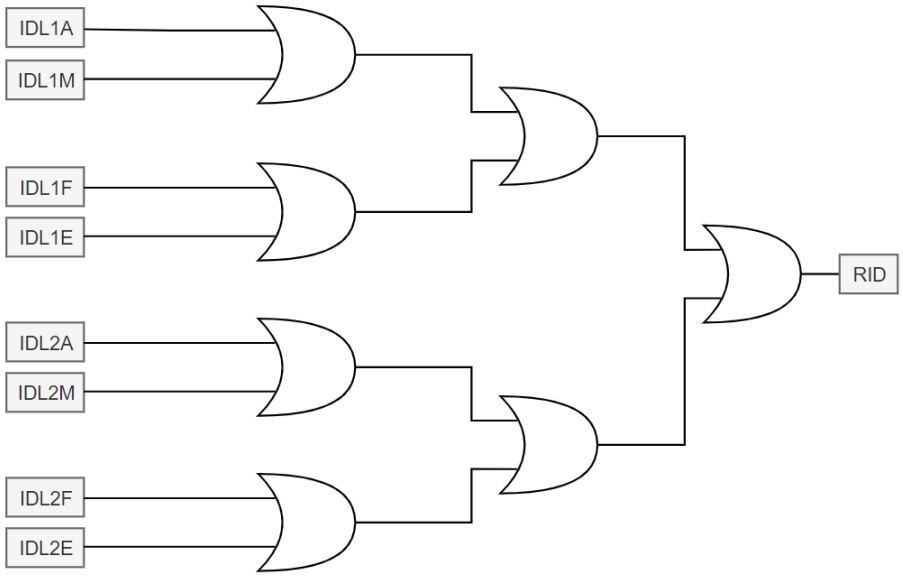


Figura 2: Esquema logico modulo LDRD

2.2. Justificación

En este módulo nos encargaremos de la lógica de detección de riesgos de datos. Por tanto únicamente se deberà mirar si el modulo LDD (de la pregunta anterior) ha encontrado dependencia de datos en alguno de los registros.

Para este modulo se trabaja con puertas OR ya que necesitamos tener una dependencia en el caso de que cualquiera de las señales generadas en el modulo LDD sea 1, por eso motivo se comprueba con puertas OR, en el caso de que cualquier valor de IDL sera 1 RID ya serà 1, ya que no nos importa en que etapa se detecte, simplemente se ha de saber que se han de inyectar NOPs hasta que deje de haber dependencia de datos.

El esquema se ha hecho de esta manera ya que el enunciado ha limitado el numero de entradas a 2 pero se podrian usar menos puertas logicas.

3. Pregunta 3

Diseñe el módulo LDRS utilizando el menor número posible de puertas lógicas, limitando el número de entradas a 2. Justifique el diseño de forma sucinta y sistemática.

3.1. Módulo LDRS

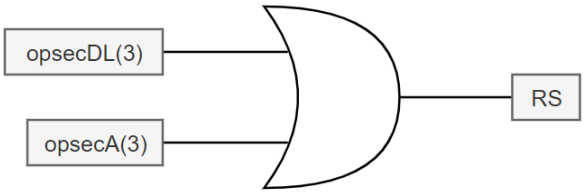


Figura 3: Esquema logico modulo LDRS

3.2. Justificación

En este módulo nos encargaremos de la lógica de detección de riesgo de secuenciamiento.

Como se puede ver aqui solo disponemos de dos pines de entrada ya que solo nos interesa el valor de la tepa DL y ALU, ya que hasta la etapa DL no podemos saber si la instruccion que estabamos ejecutando es de secuenciamiento y en la etapa ALU se escribirá en CP, por lo que en la siguiente etapa ya se podra empezar a ejecutar la instrucción correcta. Se mide con una OR ya que con que una de las dos entradas sea 1 ya se produce riesgo de secuenciamiento.

4. Pregunta 4

Diseñe el módulo LGR utilizando el menor número posible de puertas lógicas, limitando el número de entradas a 2. Justifque el diseño de forma sucinta y sistemática.

4.1. Módulo LGR

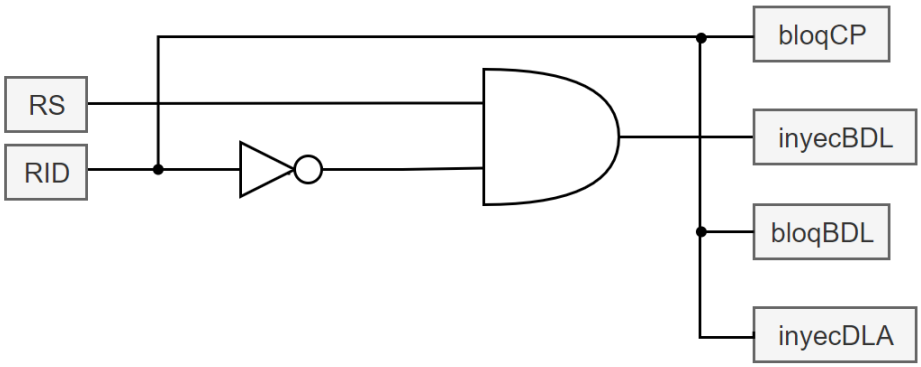


Figura 4: Esquema logico modulo LGR

4.2. Justificación

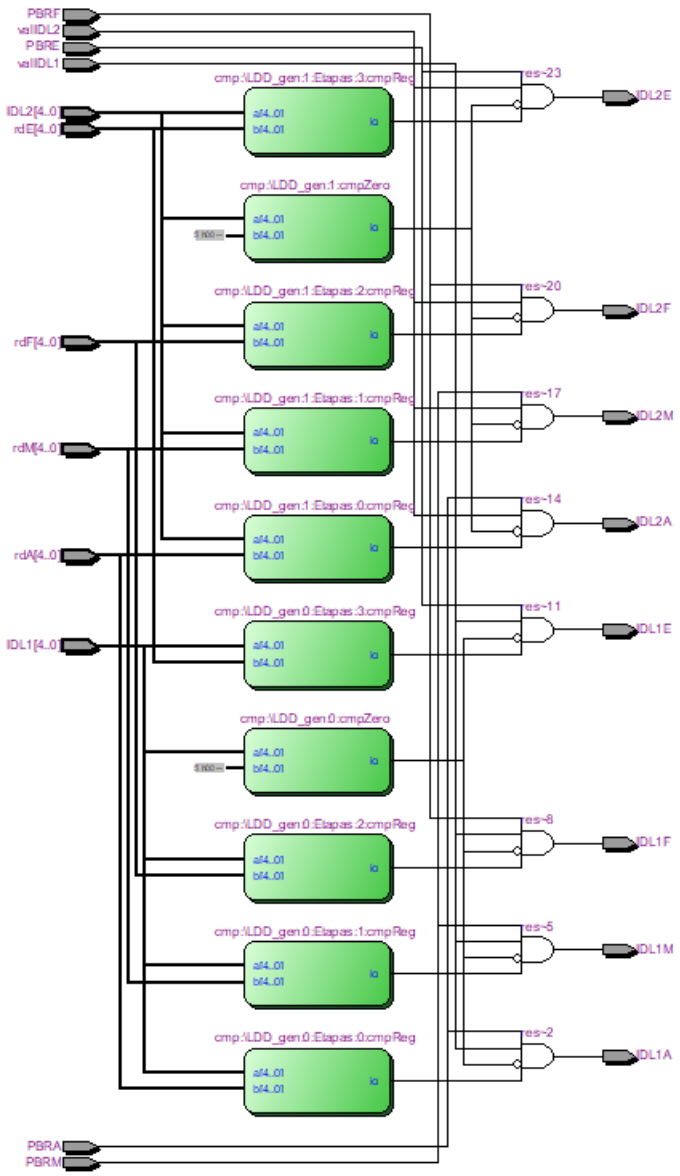
En este módulo, a partir de lo que se ha analizado en los modulos anteriores, riesgo de datos (LDRD) y riesgo de secuenciamiento (LDRS) se generarán las señales de control para los registros de desacoplo.

Como podemos ver tenemos dos entradas, que son RS (salida del modulo LDRS donde se comprueba el riesgo de secuenciamiento) y RID (salida del modulo LDRD donde se comprueba el riesgo de datos), en el caso de que haya riesgo de datos (RID = 1) se pone a 1 el valor de bloqCP y bloqDL para bloquear el registro CP y el registro entre la etapa BUS y la etapa DL, por otro lado también se le da valor 1 inyecDLA para inyectar nops desde la etapa DL a la siguiente etapa. Por otro lado el valor de inyecBDL que sirve para inyectar nops desde la etapa BUS a las siguientes etapas solo será 1 en el caso de que haya riesgo de secuenciamiento y no haya riesgo de datos, de esta manera se explican las puertas AND y NOT del esquema logico.

5. Pregunta 5

En el subdirectorio LIB (Apéndice 5.2) se encuentran los ficheros asociados al diseño de la Lógica de InterBloqueos (proyecto quartus LIB.qpf). Describa en VHDL los 4 módulos anteriores (ficheros LDD.vhd, LDRD.vhd, LDRS.vhd y LGR.vhd), utilizando un modelo estructural. Los ficheros contienen la declaración de la interface. El módulo LDD tiene una estructura regular. Utlice sentencias generate en la descripción VHDL. Entregue el esquema RTL del módulo LDD elaborado por quartus. Compruebe conjuntamente el diseño de los 4 módulos anteriores. El programa de prueba suministrado (prueba_LIB.vhd) compara en cada ciclo las salidas de los módulos diseñados con los respectivos modelos de referencia correctos.

5.1. Diseño RTL



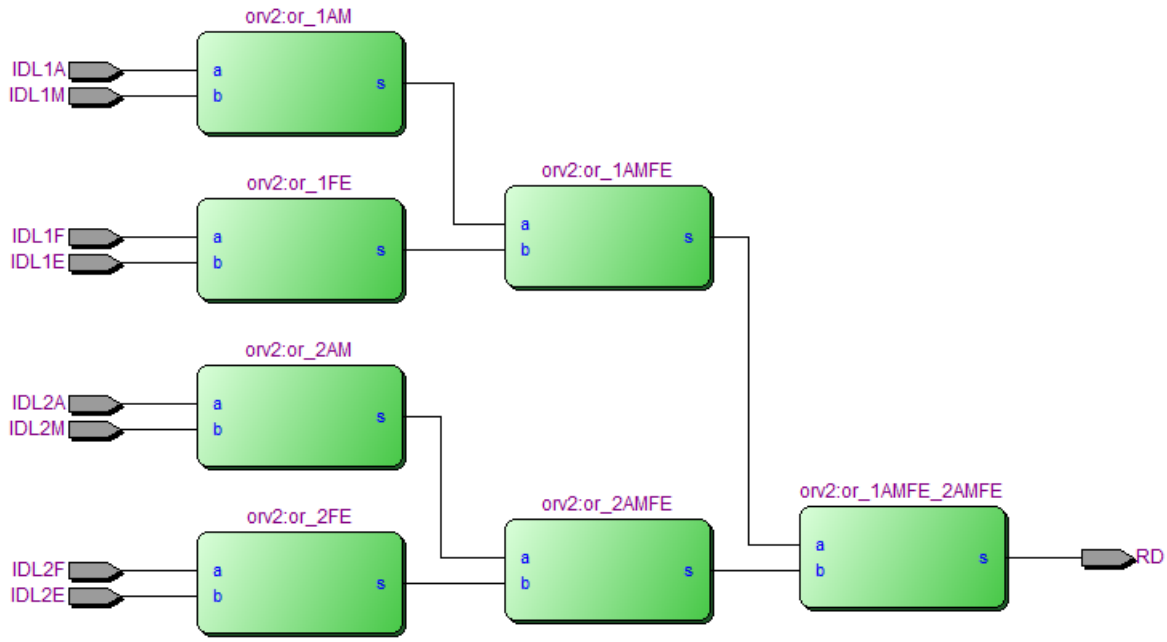
5.2. Codigos

5.2.1. LDD.vhd

```
architecture estructural of LDD is

--SIGNALS
type t_regF is array (0 to 1) of dir_reg;
signal regF: t_regf;
type t_regDest is array (0 to 3) of dir_reg;
signal regDest: t_regDest;

type vec_b4 is array (0 to 3) of std_logic;
signal PBR: vec_b4;

type vec_b2 is array (0 to 1) of std_logic;
signal c_zeros: vec_b2; -- IDL1 y IDL2
signal Val: vec_b2; -- VALIDL1 y VALIDL2

type tip_reg is array (0 to 7) of std_logic;

signal regIguales: tip_reg;
signal res: tip_reg;

begin
regF <= (0 => IDL1, 1 => IDL2);
Val <= (0 => VALIDL1, 1=> VALIDL2);
PBR <= (0 => PBRA, 1 => PBRM, 2 => PBRF, 3 => PBRE);
regDest <= (0 => rdA, 1 => rdM, 2 => rdF, 3 => rdE);


LDD_gen: for i in 0 to 1 generate
        --Comprobamos antes si algun IDLx vale 0
        cmpZero: cmp port map(a => regF(i), b => (others => '0'), ig => c_zeros(i));

        Etapas: for j in 0 to 3 generate
            cmpReg: cmp port map (a=>regF(i), b=>regDest(j), ig => regIguales((4*i)+j));
            res((4*i)+j) <= ((not c_zeros(i)) and regIguales((4*i)+j) and Val(i) and PBR(j));
        end generate Etapas;

end generate LDD_gen;

--RESULTADOS
IDL1A <= res(0) after retLDD;
IDL1M <= res(1) after retLDD;
IDL1F <= res(2) after retLDD;
IDL1E <= res(3) after retLDD;
IDL2A <= res(4) after retLDD;
IDL2M <= res(5) after retLDD;
IDL2F <= res(6) after retLDD;
IDL2E <= res(7) after retLDD;

end estructural;
```

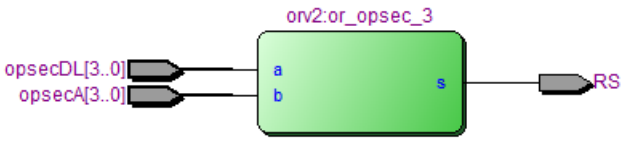
5.2.2. LDRD.vhd

```
architecture comportamiento of LDRD is

signal or_1A_1M,or_1F_1E,or_2A_2M,or_2F_2E: std_logic;
signal or1, or2: std_logic;
signal s_rd: std_logic;

component orv2 is
        port(a, b: in std_logic;
        s: out std_logic);
end component;

begin

or_1AM: orv2 port map (a => IDL1A, b=> IDL1M, s=> or_1A_1M);
or_1FE: orv2 port map (a => IDL1F, b=> IDL1E, s=> or_1F_1E);
or_2AM: orv2 port map (a => IDL2A, b=> IDL2M, s=> or_2A_2M);
or_2FE: orv2 port map (a => IDL2F, b=> IDL2E, s=> or_2F_2E);

or_1AMFE: orv2 port map (a => or_1A_1M, b=> or_1F_1E, s=> or1);
or_2AMFE: orv2 port map (a => or_2A_2M, b=> or_2F_2E, s=> or2);

or_1AMFE_2AMFE: orv2 port map (a => or1, b=> or2, s=> s_rd);

RD <= s_rd after retLDRD;

end comportamiento;
```



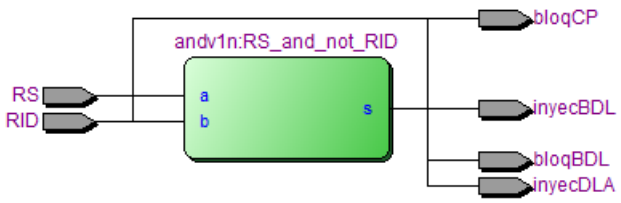
5.2.3. LDRS.vhd

```
architecture comportamiento of LDRS is

signal salida_rs: std_logic;

component orv2 is
        port(a, b: in std_logic;
        s: out std_logic);
end component;

begin
    or_opsec_3: orv2 port map (a => opsecDL(3), b=> opsecA(3), s=> salida_rs);
    RS <= salida_rs after retLDRS;
end comportamiento;
```



5.2.4. LGR.vhd

```
architecture comportamiento of LGR is

signal s_inyec_BDL: std_logic;

component andv1n is
        port(a, b: in std_logic;
                s: out std_logic);
end component;

begin
RS_and_not_RID: andv1n port map (a => RS, b => RID, s => s_inyec_BDL);

bloqCP <= RID after retLGR;
bloqBDL <= RID after retLGR;

inyecDLA <= RID after retLGR;
inyecBDL <= s_inyec_BDL after retLGR;

end comportamiento;
```



6. Pregunta 6

Utilice el programa euclides para comprobar el funcionamiento del procesador segmentado. Añada un proceso al programa de prueba (ENSAMBLADO/PRUEBAS/prueba_Rproc_MD_MI.vhd) para obtener las métricas indicadas en la tabla. Calcule la Ganancia respecto del procesador serie.

Instrucciones ejecutadas	104	Instrucciones de secuenciamiento	32
Dependencias de datos 4 ciclos de bloqueo	3	Dependencias de datos 3 ciclos de bloqueo	9
Dependencias de datos 2 ciclos de bloqueo	1	Dependencias de datos 1 ciclos de bloqueo	0
Ciclos perdidos por riesgos de datos	41	Ciclos perdidos por riesgo de secuenciamiento	64
CPI	2.009	Ganancia	2.177

Para encontrar estos valores hemos generado un código (usando el codigo de la práctica anterior) y usando las señales RID i RS, que como hemos visto en preguntas anteriores son las señales que comprueban si hay riesgo de secuenciamiento o riesgo de datos. Posteriormente lo hemos calculado manualmente para asegurar que todo este bien analizando el fichero de *resultados* que se genera ejecutando el fichero de pruebas.

Para calcular el CPI se ha efectuado de la siguiente manera:

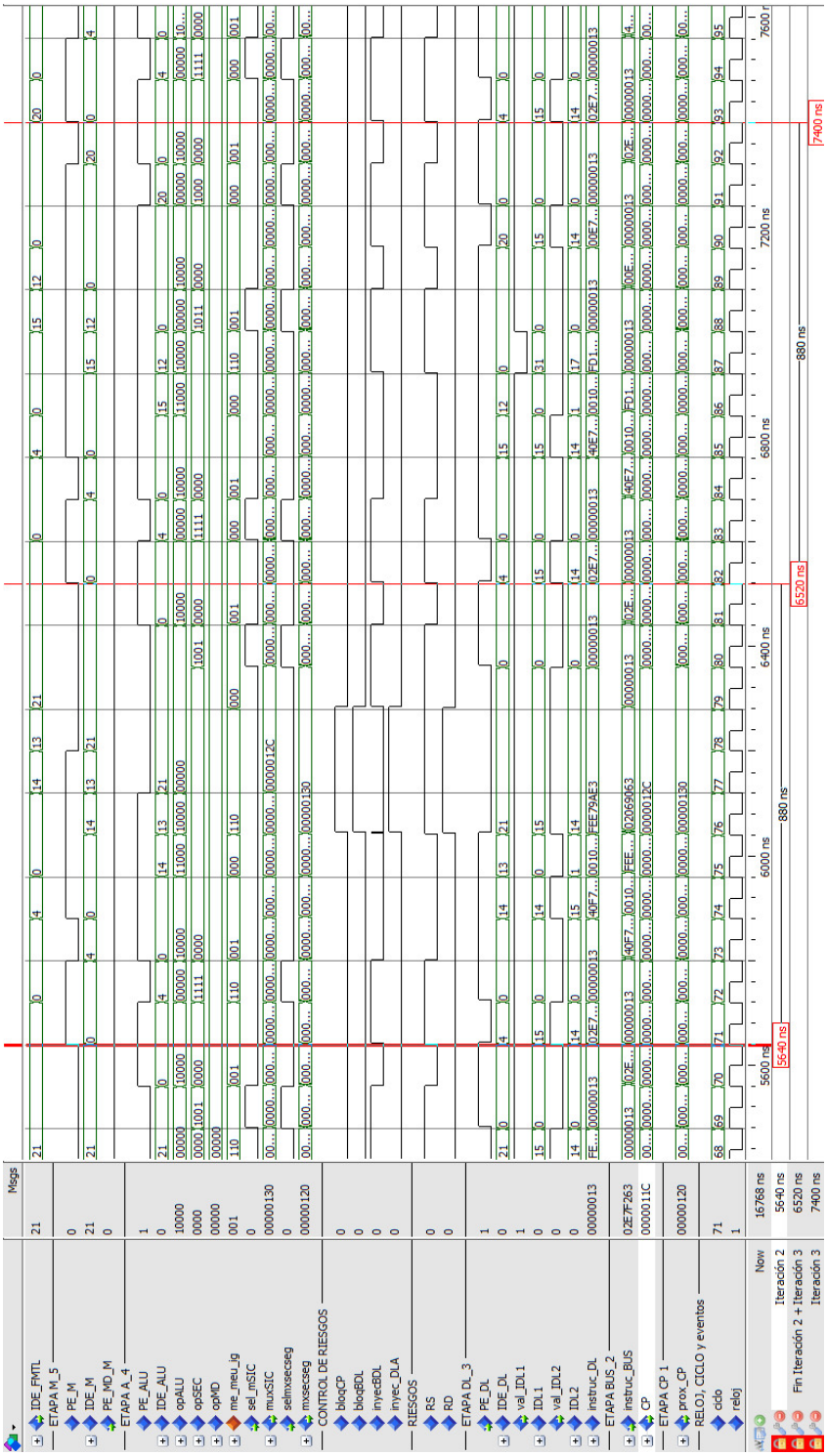
$$CPI = \frac{Ciclos\ totales}{Instrucciones\ ejecutadas} = \frac{209}{104} = \mathbf{2.009\ CPI}$$

Para calcular la Ganancia se ha efectuado de la siguiente manera: (sabiendo que el tiempo de ciclo del procesador serie es de 35ns como nos dicen en el enunciado de la practica 4 y el tiempo de ciclo del procesador segmentado es de 8ns)

$$Guany = \frac{tc_{serie} * ciclos}{tc_{seg} * ciclos} = \frac{35 * 104}{8 * (104 + 41 + 64)} = 2.177$$

7. Pregunta 7

Entregue una copia de la ventana de tiempo correspondiente a la ejecución de la segunda y tercera iteración del programa de prueba euclides. Tenga en cuenta la forma de representar las instrucciones para explicar la ventana temporal. Identifique claramente los ciclos perdidos por riesgos de datos. Para ello, marque los ciclos en los cuales la etapa E está procesando nops inyectadas por la lógica de interbloqueos.



8. Pregunta 8

Suponga que se modifica el periodo de la señal de reloj para permitir leer en un mismo ciclo el valor con el cual se está actualizando un registro del banco de registros. Deduzca cuál debería ser el tiempo el tiempo de ciclo mínimo.

tiempo de ciclo minimo (ns)	10 ns

Para calcular el tiempo de ciclo minimo, simplemente sabiendo que el tiempo de etapa es de 5ns (datos del enunciado) y ahora sabemos que sabemos que el tiempo de ciclo sera la suma de las 2 etapas ya que en un ciclo se completan dos etapas, por tanto el tiempo de ciclo minimo es 10ns.

Utilizando los resultados de la pregunta 6 para el programa euclides, cuantifique si el rendimiento de esta opción sería mejor.

Instrucciones ejecutadas	104	Instrucciones de secuenciamiento	32
Dependencias de datos 4 ciclos de bloqueo	0	Dependencias de datos 3 ciclos de bloqueo	3
Dependencias de datos 2 ciclos de bloqueo	9	Dependencias de datos 1 ciclos de bloqueo	1
Ciclos perdidos por riesgos de datos	28	Ciclos perdidos por riesgo de secuenciamiento	64
CPI	1.88	Ganancia/Pérdida	0.853

En este caso como se puede leer y escribir en el mismo ciclo, los casos de dependencias de datos se reducen, por tanto mirando los datos del ejercicio 6 y las dependencias de datos de 4 ciclos de bloqueo pasan a ser 0, las dependencias de datos de 3 ciclos de bloqueo pasan a tener el valor de las dependencias de datos de 4 ciclos de bloqueo y asi sucesivamente, de esta manera se reducen los ciclos perdidos por riesgo de datos a 28.

Para calcular el CPI se ha efectuado de la siguiente manera:

$$CPI = \frac{Ciclos\ totales}{Instrucciones\ ejecutadas} = \frac{196}{104} = \mathbf{1.885\ CPI}$$

Para calcular la Ganancia se ha efectuado de la siguiente manera:

$$Guany = \frac{tc_{segEj6} * ciclos}{tc_{segEj8} * ciclos} = \frac{8 * (104 + 41 + 64)}{10 * (104 + 28 + 64)} = 0.853$$