Práctica 4. Programación del controlador de audio

Esta práctica consta de 4 actividades. Las tres primeras requieren la realización de un programa independiente en ensamblador de la arquitectura de repertorio de instrucciones del procesador NIOS II. La cuarta actividad utiliza un programa ya desarrollado para medir y comparar las prestaciones de tres versiones del procesador NIOS II.

Lo nuevo de esta práctica en comparación con las tres anteriores es que se utiliza la parte del hardware de la estructura del computador DE2 que se encarga de registrar y reproducir audio y que se denomina "Puerto de Audio (Audio Port)". Este controlador de E/S se encuentra integrado en la estructura "Media Computer" de la placa DE2 y sus características técnicas se encuentran explicadas en el documento titulado *Media Computer System for the Altera DE2 Board* (DE2_Media_Computer.pdf, página 31). Durante la práctica se utilizarán unos auriculares y un micrófono conectados a los conectores etiquetados *Line Out* y *Mic In* respectivamente, como se puede ver en la Figura 1.

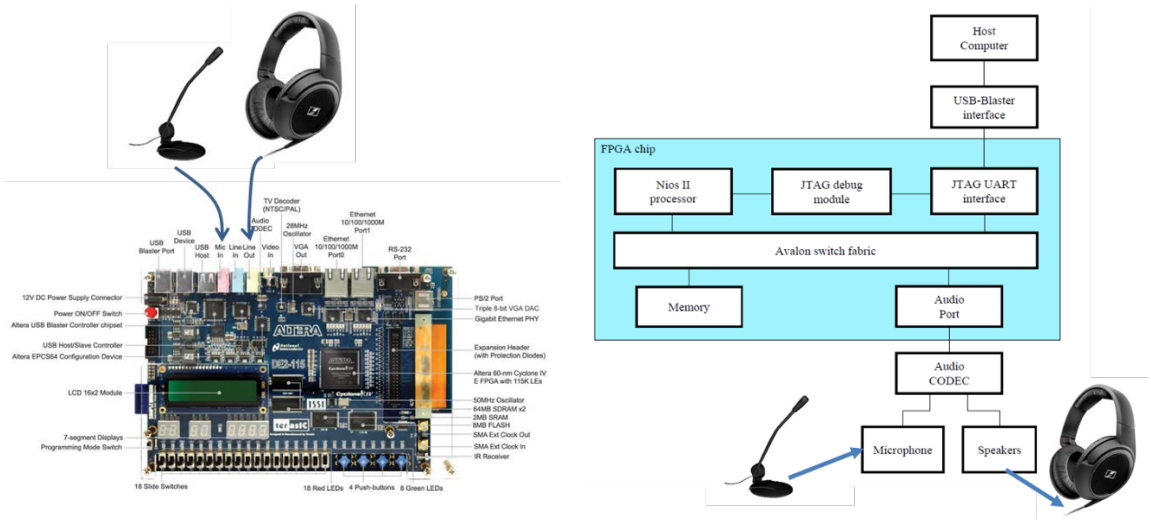

Figura 1. Organización de la estructura "Media Computer" de la placa DE2.

En la Figura 1 se representa también un diagrama de bloques de la estructura "Media Computer" una vez que el hardware de la FPGA que está en la placa DE2 queda configurado. Observar que en la placa DE2 existe un chip denominado "Audio CODEC" que está conectado al puerto de audio. El audio CODEC se encarga de generar o grabar el sonido a través de unos altavoces o de un micrófono, respectivamente. Una de sus características consiste en que el ritmo de grabación y reproducción es de 48 KHz; es decir, 48000 muestras de audio grabadas o reproducidas por segundo. Las muestras son

bien obtenidas y guardadas en las grabaciones o enviadas a los altavoces en las reproducciones. En ambos casos se utilizan dos parejas de memorias FIFO, una pareja para las grabaciones y la otra pareja para las reproducciones. Estas FIFOs forman parte del puerto de audio que se conecta al audio CODEC y cuyo hardware interno se representa en la Figura 2. Cada FIFO de una pareja se conecta a un canal de audio: izquierdo (left) o derecho (right).

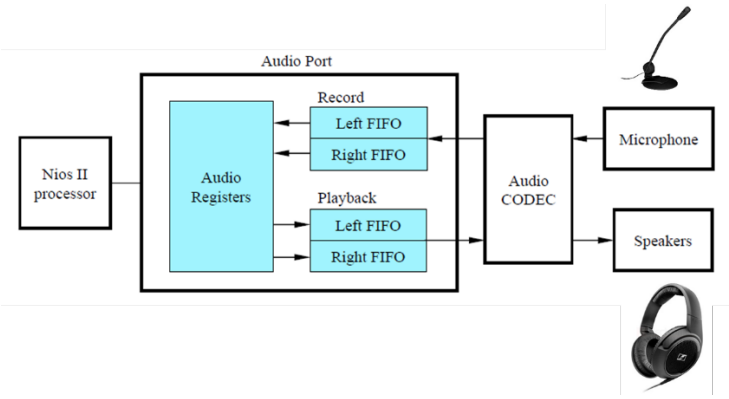

Figura 2. Organización interna del controlador de audio en DE2

Los registros del puerto de audio que son accesibles desde programa se representan en la Figura 3. El registro base del puerto de audio que corresponde al registro de control está asignado a la dirección `0x10003040`. Observar que existen dos registros de datos, uno para el canal de audio izquierdo y otro para el canal derecho de audio.

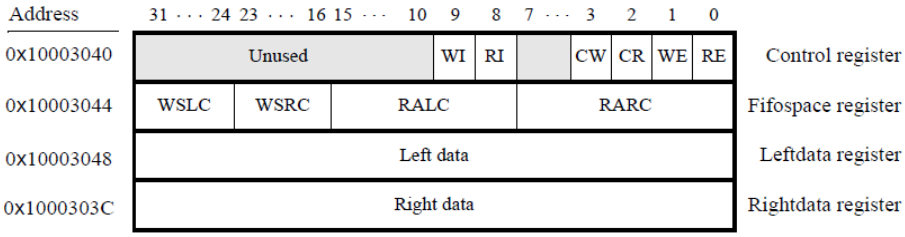

Figura 3. Registros del controlador del puerto de audio en DE2 asignados a direcciones del espacio de direccionamiento del procesador NIOS II

Actividad 1. Grabación y reproducción de audio

Actividad 1.1 - Crea un programa en ensamblador para el procesador NIOS II que realice una grabación de 3 segundos (150000 muestras de audio de 32 bits) cuando se pulse el pulsador KEY1, y que vuelva a reproducirlo posteriormente cuando se pulse el pulsador KEY3. Para ello, un micrófono y unos auriculares deben estar conectados a la placa DE2, en los conectores *Mic In* y *Line Out* respectivamente.

Para realizar este programa se propone seguir el diagrama de flujo de la Figura 4. El programa ensamblador debe realizar las siguientes tareas:

- Cuando se apriete el pulsador KEY1, se deberán leer los datos registrados por el controlador de audio durante 3 segundos y almacenarlos en memoria a partir de una dirección de memoria SDRAM que debe abarcar 600000 bytes (150000 × 4) para el canal izquierdo de audio, y otros 600000 bytes para el canal derecho de audio.
- Cuando se apriete el pulsador KEY3, se deberán leer los datos guardados en memoria SDRAM durante la grabación y enviarlos al controlador de audio para su reproducción.

Actividad 1.2 – Modifica el programa anterior para que la reproducción de audio se vea afectada por los interruptores de desplazamiento:

  - Cuando el interruptor SW1 sea 1, se oirá sólo un dato o muestra de audio de cada dos consecutivos.
  - Cuando el interruptor SW0 sea 1, cada dato o muestra se duplicará.
  - Cuando SW0 y SW1 sean 0, la reproducción es normal.

Pasos a seguir para las actividades 1.1 y 1.2: Escribir el programa en ensamblador, crear un nuevo proyecto seleccionando la estructura “Media Computer”, insertar el programa en código ensamblador, ensamblar y linkar, cargar la configuración de la placa DE2, cargar el programa ensamblado y linkado, ejecutar programa, y comprobar su funcionamiento utilizando el micrófono y auriculares.

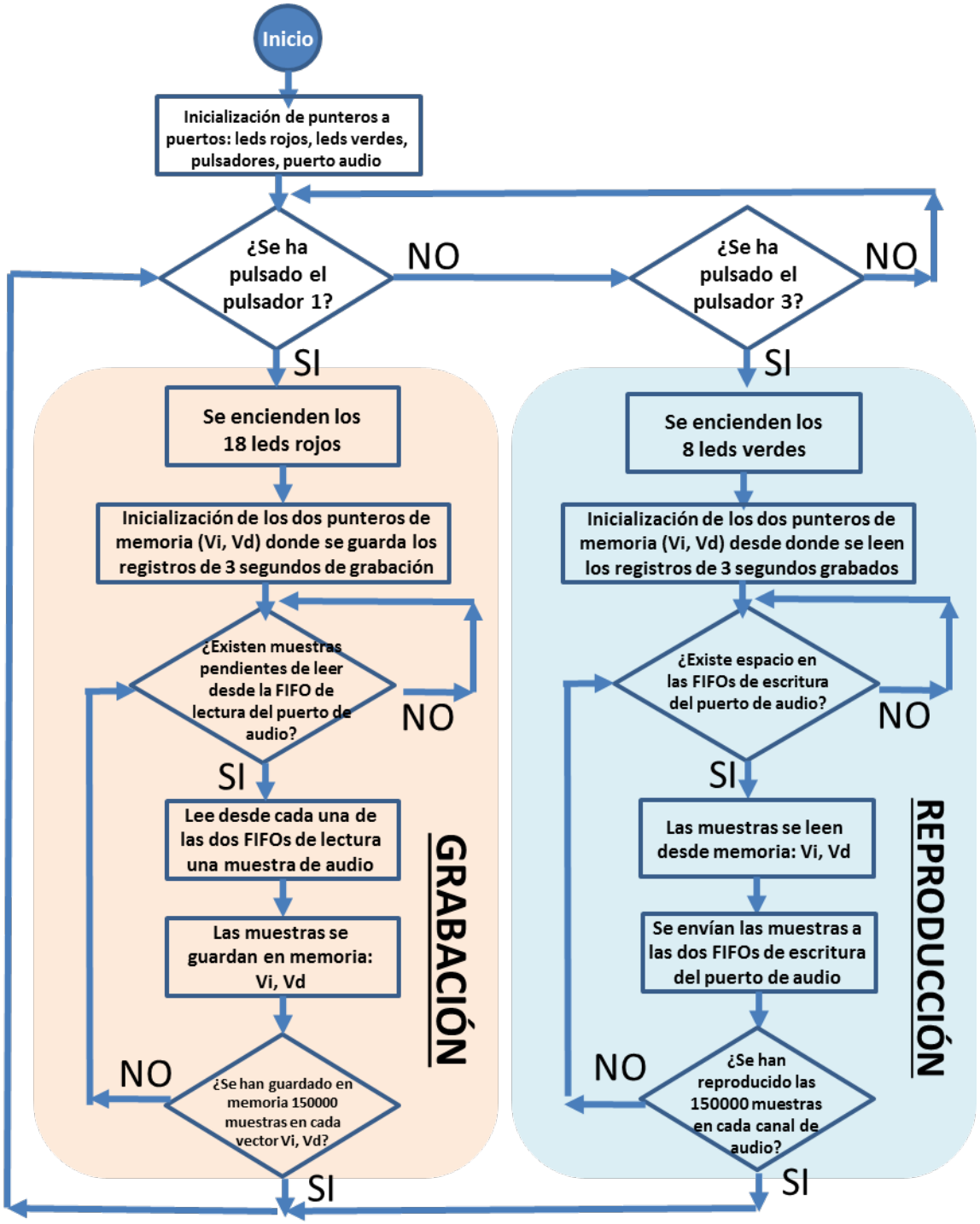

Figura 4. Ejercicio 1.1 de la Actividad 1

Actividad 2. Simulación del eco

En este ejercicio se realizará un programa en ensamblador que lea el sonido grabado desde el micrófono y reproducirá posteriormente el mismo sonido grabado. Sin embargo, el mismo sonido se repetirá un poco más tarde pero con menor volumen, y se repetirá de nuevo posteriormente a un volumen todavía inferior. Todo esto simulará el efecto del eco.

Pasos a seguir: Los mismos que para la Actividad 1.

Actividad 3. Simulación de un piano

En este ejercicio se generarán sonidos a través del controlador de audio de DE2. Un "tono musical" se genera a través de una onda senoidal de una determinada frecuencia. Por ejemplo, los tonos musicales en la cuarta octava de un piano tienen las siguientes frecuencias (*f*):

C = 262 Hz
C# = 278 Hz
D = 292 Hz
D# = 310 Hz
E = 328 Hz
F = 348 Hz
F# = 370 Hz
G = 392 Hz
G# = 416 Hz
A = 440 Hz
A# = 466 Hz
B = 494 Hz

Para generar una onda senoidal de una frecuencia determinada tienes que generar muestras de audio cuya amplitud venga determinada por la función matemática de una senoide: $v(t) = A\sin(2\pi ft)$.

Donde *A* es la amplitud de la onda, *f* es la frecuencia, y *t* es el índice de tiempo. En la placa DE2, la onda de sonido se genera enviando los valores *v(t)* al puerto de audio. El controlador de audio CODEC lee las FIFOs del puerto de audio cada cierto intervalo de tiempo, denominado "intervalo de muestreo". El número de muestras de audio que emite el circuito CODEC de la placa DE2 es de 48000 cada segundo. La inversa de este valor determina el intervalo de muestreo del CODEC. Por ello, dividiendo el orden de la muestra por 48000 se obtiene el valor del tiempo que habría que incluir en la función senoide *v(t)*.

Actividad práctica. Crea un programa que simule un piano con los tonos de la cuarta octava. El usuario tocará el piano apretando las teclas del teclado del ordenador. Utiliza

las teclas a, w, s, e, d, f, t, g, y, h, u, j para representar cada tono musical respectivamente. Cuando se apriete cada tecla, el programa reproducirá durante medio segundo el tono musical correspondiente.

Pasos a seguir:

1. Crear un fichero, tonos.h, con 12 vectores (.word) que representen los tonos musicales de la cuarta octava. Cada vector debe consistir en 24000 datos que se correspondan con las muestras de la correspondiente nota musical durante 0,5 segundo. El intervalo entre muestra y muestra es 2,1 10^{-5} segundos = 0,5 segundos / 24000. La amplitud (A) de la señal musical debe ser 2,0 10^{7}.

2. Escribe el programa en ensamblador que simule el piano. Asegura que el fichero tonos.h se incluye en el programa.

3. Crea un nuevo proyecto seleccionando la estructura "Media Computer", inserta el programa ensamblador, ensambla, carga la configuración de la placa DE2, carga el programa ensamblado y linkado, ejecuta el programa y comprueba su funcionamiento utilizando los auriculares.

Actividad 4. Análisis de prestaciones de los procesadores NIOS II / {e, s, f} para aplicaciones de audio

En esta actividad se compararán varias medidas de las prestaciones de los tres procesadores NIOS II / {e, s, f} para analizar qué procesador ejecuta en menos tiempo las instrucciones de un programa que genera sonidos musicales de forma sintética.

Actividad práctica. Utiliza el programa benchmark denominado *benchNIOSII* que genera durante 0,75 segundo una secuencia de varios tonos musicales (ficheros: `benchNIOSII.s, BCD.s, DIV.s, JTAG.s, DO.s, RE.s, MI.s`). El programa empieza por solicitar a través del terminal de AMP que se apriete la tecla *a* para comenzar el benchmark (ver Figura 5). Como se puede observar, cuando el programa termina, varias medidas de prestaciones son enviadas a la terminal.

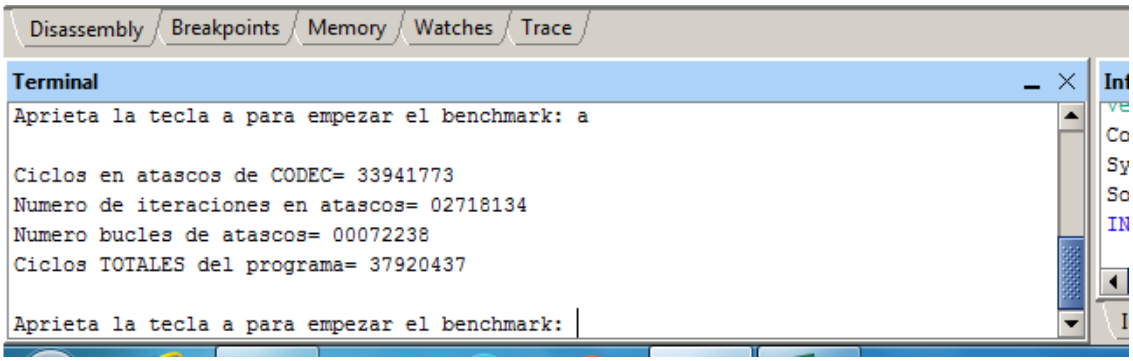

Figura 5. Resultados del programa *benchNIOSII*.

El diagrama de flujo del benchmark *benchNIOSII* se muestra en la Figura 6. Se identifican principalmente tres bucles anidados dentro del código.

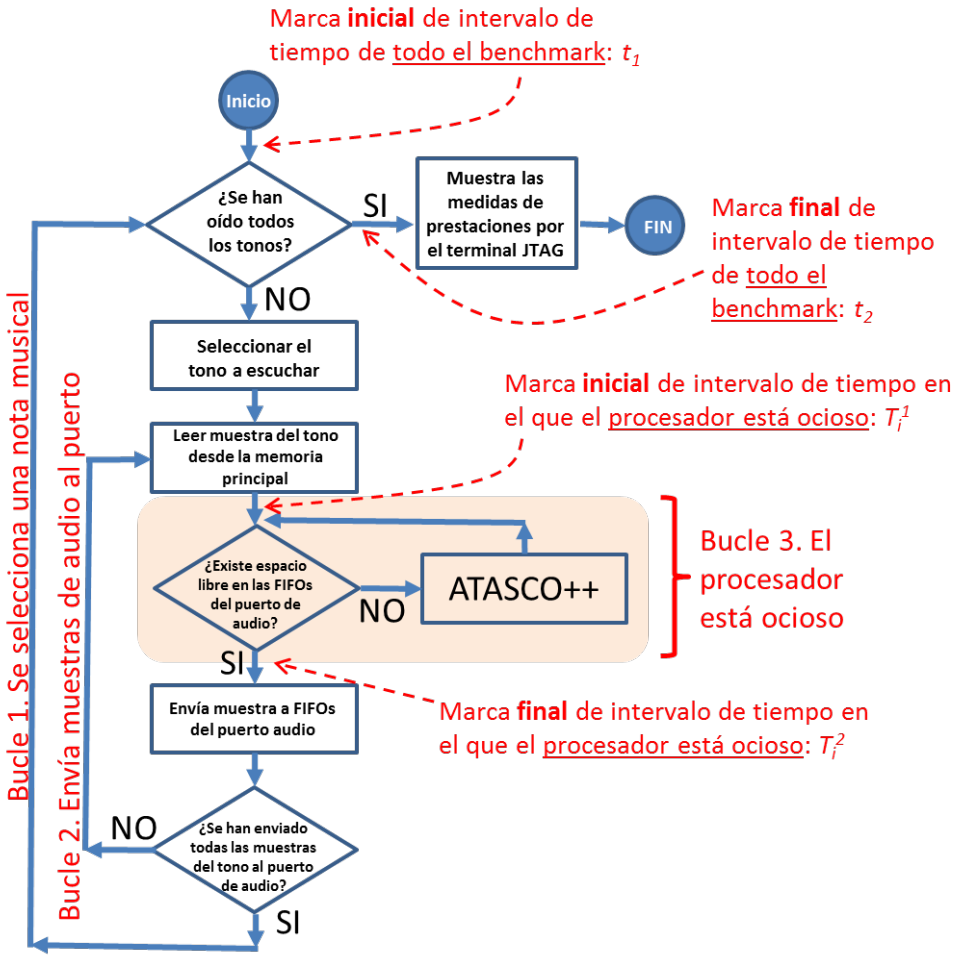

Figura 6. Diagrama de bloques del programa *benchNIOSII*.

Bucle 1. Es el más externo y en cada iteración se selecciona una nota musical distinta: do, re, o mi. Este bucle se encuentra en el programa principal (`_start`) a partir de la etiqueta `secuencia` del fichero `benchNIOSII.s`. En total se realizan tres iteraciones, en cada una de ellas se reproduce durante 0,25 segundo una de las notas musicales.

Bucle 2. En este otro bucle se leen desde memoria SDRAM las distintas muestras de audio de 32 bits y se envían al puerto de audio. El Bucle 2 se encuentra en la rutina `audioOUT` (`benchNIOSII.s`) a partir de la etiqueta `LOOP4`.

Bucle 3. Este tercer bucle anidado es el más interno del programa y se ejecuta entre el envío de las sucesivas muestras de audio al puerto de audio. Durante este bucle, el procesador no puede enviar ninguna muestra de audio al puerto de audio porque éste no tiene espacio para alojar una nueva muestra de audio (ver Figura 7). El Bucle 3 se encuentra en la rutina `audioOUT` (`benchNIOSII.s`) a partir de la etiqueta `LOOP6`.

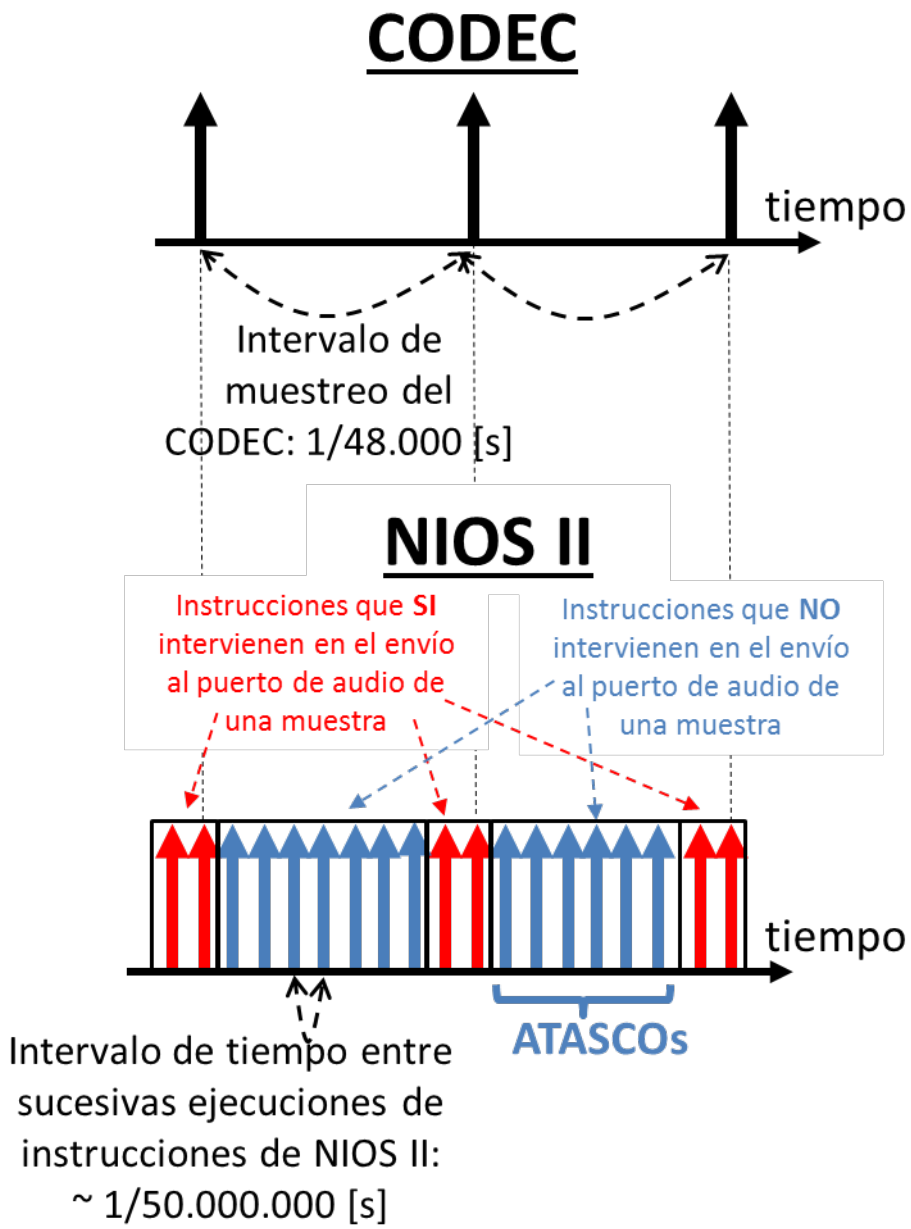

Figura 7. Representación de los intervalos de tiempo en los que el procesador está ejecutando instrucciones que se encargan de reproducir audio (representadas como flechas en rojo), y de los intervalos en los que el procesador está esperando a que el puerto de audio pueda alojar una nueva muestra de audio (las respectivas instrucciones se representan con flechas azules).

Al finalizar el programa *benchNIOSII*, las siguientes cuatro medidas se proporcionan a través de la terminal de Altera Monitor Program:

- *Ciclos en atascos de CODEC*, t_{sb}: Tiempo en número de ciclos de reloj (50 MHz) en el que el procesador está esperando a que las FIFOs de salida del puerto de audio

permitan introducir una muestra de audio adicional; $t_{sb} = \Sigma_i(T_i^2 - T_i^1)$, donde T_i^1 y T_i^2 son las marcas de tiempo en ciclos de reloj del inicio y final de la *i*-ésima ejecución del Bucle 3. Esta medida representa al tiempo durante el cual el procesador no se encuentra enviando muestras al puerto de audio de la placa DE2. Este tiempo lo proporciona el programa y se visualiza a través de la terminal (ver Figura 5). Para obtener t_{sb} se utiliza el Timer de DE2-MediaComputer.

- *Número total de iteraciones en atascos, I_{sb}*: Número total de iteraciones del Bucle 3 (ver Figura 6) de espera durante las cuales las FIFOs de salida no permiten introducir nuevas muestras de audio, y por ello, decimos que el procesador está ocioso. Este número lo proporciona el programa y se visualiza a través de la terminal (ver Figura 5).
- *Número de bucles de atasco, B_{sb}*: Número de veces que se entra en el Bucle 3 (ver Figura 6) de espera durante el cual las FIFO de salida no permiten introducir nuevas muestras de audio, y por ello, decimos que el procesador está ocioso (tiempo representado por ATASCOs en Figura 7). Este número B_{sb} lo proporciona el programa y se visualiza a través de la terminal (ver Figura 5). Observar que el número de veces que se entra en el Bucle 3 no tiene por qué ser igual al número de iteraciones en atascos (I_{sb}).
- *Ciclos totales del programa, t_T*: Tiempo total de ejecución en ciclos de reloj (50 MHz) de todo el benchmark, incluido el tiempo de reproducción; $t_T = t_2 - t_1$, donde t_1 y t_2 son las marcas de tiempo en ciclos de reloj que corresponden al inicio y final de la ejecución del programa, respectivamente. Este tiempo lo proporciona el programa y se visualiza a través de la terminal (ver Figura 5).

Este programa benchmark se tendrá que ejecutar con tres configuraciones distintas de procesador (ver Figura 8):

- **NIOS II/e**, procesador multiciclo con al menos 6 ciclos por instrucción, sin memoria cache de ningún tipo, frecuencia de reloj de 50 MHz. Ficheros de configuración:
  - DE2: `DE2_P4_MediaComputer_N2e.{sopcinfo, sof}`
  - DE2-115: `DE2_115_ P4_MediaComputer_N2e.{sopcinfo, sof}`
- **NIOS II/s**, procesador segmentado con 5 etapas de segmentación, sin memoria cache de datos pero con memoria cache de instrucciones, frecuencia de reloj de 50 MHz. Ficheros de configuración:
  - Configuración por defecto Media Computer de DE2 y DE2-115 que proporciona AMP
- **NIOS II/f**, procesador segmentado con 6 etapas de segmentación, con memoria cache de datos L1 de 32 KB con 32 bytes por bloque y memoria cache de instrucciones, frecuencia de reloj de 50 MHz. Ficheros de configuración:
  - DE2: `DE2_ P4_MediaComputer_N2f_16K32B.{sopcinfo, sof}`
  - DE2-115: `DE2_115_ P4_MediaComputer_N2f_32K32B.{sopcinfo, sof}`

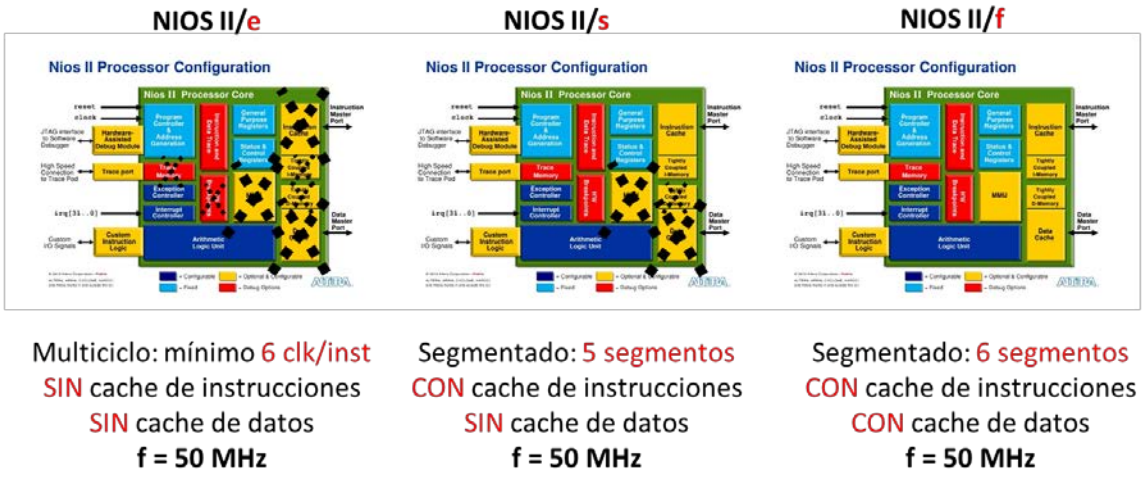

Figura 8. Procesadores NIOS II / {e, s, f}.

Rellena las dos tablas siguientes con los datos obtenidos en las tres distintas ejecuciones del benchmark *benchNIOSII* cuando se usan en cada ejecución uno de los tres procesadores NIOS II en la placa DE2 que dispones en el laboratorio:

Placa (DE2 o DE2-115)	**Procesador**	t_{sb}	t_T	*%t*
	NIOS II/e			
	NIOS II/s			
	NIOS II/f			

Placa (DE2 o DE2-115)	**Procesador**	B_{sb}	I_{sb}	N_{sb}	CPI_{sb}	P_{sb}
	NIOS II/e					
	NIOS II/s					
	NIOS II/f					

Donde ***%t*** representa al porcentaje del tiempo total en el que el procesador se encuentra ocioso (***%t*** =100 × t_{sb} / t_T). N_{sb} es el número de instrucciones ejecutadas mientras el procesador está ocioso. Para su obtención es necesario fijarse en el código fuente del Bucle 3 del programa. CPI_{sb} es el número promedio de ciclos por instrucción del procesador mientras el procesador está ocioso en el Bucle 3. Para su obtención se divide el número de ciclos t_{sb} entre el número de instrucciones N_{sb}. Y P_{sb} es el número promedio de ciclos de penalización por instrucción en los que el procesador está ocioso. Para su cálculo es necesario restar a CPI_{sb} el número promedio de ciclos en el que se ejecuta una instrucción en cada procesador suponiendo que no incurre en ningún ciclo de penalización (ver Figura 8): 1 para NIOSII/s y NIOSII/f, y 6 para NIOSII/e.

A continuación, responde a las siguientes preguntas:

1. ¿Qué procesador es de mayores prestaciones en tiempo de ejecución? Justifica la respuesta con los datos de la tabla.
2. Analiza el código fuente del programa benchmark *benchNIOSII* y descubre cuál es el bucle en el que el procesador está ocioso. Cuenta el número de instrucciones que se ejecutan en el bucle ocioso (Bucle 3). Para esto último, se recomienda utilizar las medidas B_{sb} e I_{sb}.
3. Calcula el número de ciclos por instrucción (CPI) durante el bucle en el que el procesador está ocioso (Bucle 3). Para ello necesitas calcular el número de instrucciones que se ejecutan en dicho bucle. Este número se puede encontrar al analizar la rutina `audioOUT` (`benchNIOSII.s`) a partir de la etiqueta `LOOP6`.
4. ¿Qué procesador es de mayores prestaciones en número de ciclos por instrucción (CPI) durante el bucle en el que el procesador está ocioso?
5. ¿Existe alguna relación entre las respuestas a las preguntas 1 y 4? Justifica las respuestas con los datos de la tabla.

Documentación básica:

- Altera - University Program; Media Computer System for the Altera DE2 Board, Altera Corporation, 2009 (DE2_Media_Computer.pdf)
- Altera - University Program; Laboratory Exercise 8 - Audio CODEC, Altera Corporation, 2009 (lab8.pdf)
- Altera; Nios II Core Implementation Details (2/4/2015): `https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/hb/nios2/n2cpu_nii51015.pdf`