TEC | Tecnológico de Costa Rica

Instituto Tecnológico de Costa Rica
Área Académica de Ingeniería en Computadores
CE 4301 — Arquitectura de Computadores I

Proyecto Grupal 1
Diseño e Implementación de un ASIP para reverberación de audio

Fecha de asignación:	14 de abril de 2021	Fecha de entrega:	14 de mayo de 2021
Grupos:	3-4 personas	Profesores:	Luis Chavarría Zamora

Mediante el desarrollo de este proyecto, el estudiante aplicará los conceptos de arquitectura de computadores en el diseño e implementación en hardware de un *Application Specific Instruction Set Processor* (ASIP) para introducción y corrección de eco. Atributos relacionados: **A**nálisis de **P**roblemas (AP), el cual se encuentra en **A**vanzado (A).

1. Descripción General: Reverberación

La reverberación es un fenómeno acústico relacionado a la persistencia del sonido en un medio cerrado. En el medio (paredes, muebles, personas, entre otros objetos) se se reflecta el sonido emitido por la fuente. Estas reflexiones van decayendo gradualmente mientras son absorbidas por los objetos del medio. El fenómeno de reverberación es representado gráficamente en la Figura 1.

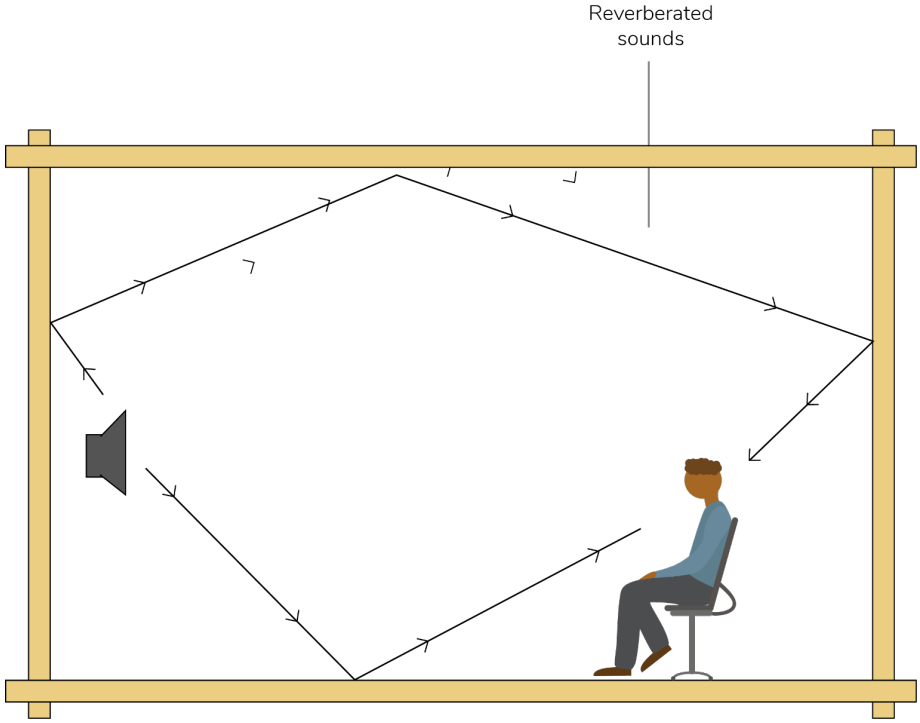

Figura 1: Representación gráfica del fenómeno de reverberación

1.1. Inserción de reverberación a una señal de audio

El efecto de reverberación puede ser modelado con la función de transferencia

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1-\alpha}{1-\alpha z^{-k}}, \tag{1}$$

donde $H(z)$ es la función de transferencia, $Y(z)$ es la salida del sistema y $X(z)$ es la entrada del sistema. α es la atenuación del medio y k es el retardo. Luego de despeje matemático y transformaciones de dominio, se obtiene la ecuación de diferencias, la cual permitiría implementar este fenómeno en un sistema digital

$$y(n) = (1-\alpha)x(n) + \alpha y(n-k), \tag{2}$$

donde $y(n)$, es la muestra número n a la salida del sistema, $x(n)$ es la muestra número n a la entrada del sistema y $y(n-k)$ es la muestra n a la salida con un retardo de k muestras.

Como se observa, este sistema tiene un requerimiento de memoria importante pues necesita acceder a salidas anteriores. Si se almacenan en memoria los datos anteriores a la salida se necesitaría una memoria igual al doble del archivo de entrada, o inclusive esta podría ser infinita si el procesamiento fuera en vivo. Una técnica usada para aliviar este requerimiento de memoria es usar un *buffer* circular. Este se ejemplifica en la Tabla 1 con un *buffer* circular para 4 datos.

Tabla 1: *Buffer* circular para 4 entradas de audio

Memoria	Dato	Índice
0x0100		
0x0104	0.304	y(n-2)
0x0108	0.350	y(n-1)
0x010c	-0.604	y(n)
0x0110	0.566	y(n-3)
0x0118		

(a) Buffer circular en un momento n

Memoria	Dato	Índice
0x0100		
0x0104	0.304	y(n-3)
0x0108	0.350	y(n-2)
0x010c	-0.604	y(n-1)
0x0110	-0.348	y(n)
0x0118		

(b) Buffer circular después de la siguiente muestra

Este *buffer* es útil para usar eficientemente la memoria porque la ecuación (2) solo necesita a lo sumo la muestra $n-k$ y luego, no se vuelve a utilizar para el procesamiento. **¿Cual sería el tamaño recomendado para el *buffer* circular?**

1.2. Reducción de reverberación en una señal de audio

Para eliminar el eco en una señal se necesita hacer el proceso inverso, este puede ser modelado con la función de transferencia

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1 - \alpha z^{-k}}{1 - \alpha}. \tag{3}$$

Luego de despeje matemático y transformaciones de dominio, se obtiene la ecuación de diferencias, la cual permitiría implementar este fenómeno en un sistema digital

$$y(n) = \frac{x(n) - \alpha x(n-k)}{1 - \alpha} = \frac{1}{1 - \alpha}[x(n) - \alpha x(n-k)], \tag{4}$$

donde $x(n-k)$ es la muestra k anterior a la actual. Como un aspecto interesante, el factor del denominador es un ajuste de la magnitud para volver el sonido a la magnitud original antes de insertarle el efecto de reverberación.

Algunos valores para α y k se muestran en la Tabla 2, para una frecuencia de muestreo F_s. Se pueden combinar en diferentes combinaciones obteniendo diferentes ejemplos de reverberación.

Tabla 2: Valores recomendados para k y α

Retardo (k)	Atenuación (α)
$F_s \times 50ms$	0.6
$F_s \times 250ms$	0.4
$F_s \times 500ms$	0.2

Se recomienda usar Octave como software de alto nivel para modelado. El código se muestra a continuación para $k = F_s \times 50ms$ y $\alpha = 0{,}6$ [1]:

```
% Autor: Luis Chavarr a Zamora
% Software: GNU/Octave
% C digo de ejemplo para inserci n y retiro de reverberaci n

% Borra lo anterior
clear
clc

% Lectura del archivo
[archivo, fs]=audioread('nombre_archivo');
```

[1] Sin embargo, pueden ser diferentes combinaciones

```
% Par metros del filtro
alpha=0.6;
k_sinfs=0.05;
% Ajuste para el retardo k
k=fs*k_sinfs;
% Redondeo hacia abajo, pues k debe ser entero
k=floor(k);

% Se crea una matriz de k filas llena de
% ceros para implementar el retardo hasta k muestras
polos=zeros(1,k);
% Se coloca el y(n)
polos(1)=1;
% Se coloca el retardo a la muestra y(n-k)
polos(k)=alpha;
% Se coloca la atenuaci n (1 - alpha) a la muestra x(n)
ceros=1-alpha;
% Aplica el filtro
salida_filtro=filter(ceros,polos,archivo);
% Reproduce el sonido
player_salida=audioplayer(salida_filtro,fs);
play(player_salida);

% Eliminaci n de reverberaci n
input('Presione enter para escuchar reverberaci n eliminada');
% Se invierten Y y X al ser un proceso inverso
salida_filtro=filter(polos,ceros,salida_filtro);
player_salida=audioplayer(salida_filtro,fs);
play(player_salida);
```

El código mostrado anteriormente, es solo para modelar el resultado y efecto de la reverberación. Se recomienda usarlo como instrumento de comprobación. En el sistema digital solo se deberían implementar las ecuaciones de diferencias (2) y (4). Algunas observaciones importantes sobre los datos:

1. Los valores tanto a la entrada como a la salida de audio en cada momento están entre -1 y 1 (a veces es más reducido el rango). Cualquier valor fuera de este rango puede causar saturación en el sonido.

2. Los valores de α están entre 0 y 1. Por ende, $1 - \alpha$ también está en este rango.

3. El valor de k es entero al ser un entorno digital cuantizado.

Como se observa, al trabajar con valores en el rango entre -1 y 1, se necesita algún tipo de representación para esto, por esa razón se debe utilizar punto flotante o punto fijo.

1.3. Aritmética en punto fijo

Se debe usar esta representación para los valores negativos e inferiores a la unidad. A continuación se muestran ejemplos de suma y multiplicación en punto fijo con formato Q7.8 (conformado por un signo, 7 bits en parte entera y 8 bits en la parte decimal).

1.3.1. Suma

Para suma se muestran las Tablas 3, 4 y 5.

Tabla 3: Suma en punto fijo Q7.8

Número decimal	Signo	Parte entera	Parte fraccionaria
1.5	0	000 0001	1000 0000
3.25	0	000 0011	0100 0000
4.75	0	000 0100	1100 0000

Tabla 4: Suma en punto fijo Q7.8

Número decimal	Signo	Parte entera	Parte fraccionaria
1.5	0	000 0001	1000 0000
-3.25	1	111 1100	1100 0000
-1.75	1	111 1110	0100 0000

Tabla 5: Suma en punto fijo Q7.8

Número decimal	Signo	Parte entera	Parte fraccionaria
-1.5	1	111 1110	1000 0000
3.25	0	000 0011	0100 0000
1.75	0	000 0001	1100 0000

¿Cómo tratar los valores negativos?, ¿Cómo detectar los overflow?

1.3.2. Multiplicación

Para la multiplicación en punto fijo para Q7.8 se usa el siguiente formato:

$$result = high << 8 + mid + low >> 8, \tag{5}$$

donde si se multiplica $Qa.b \times Qc.d$

$$high = a \times c, \tag{6}$$

$$mid = a \times d + b \times c, \tag{7}$$

$$low = b \times d. \tag{8}$$

Se nota que por los corrimientos es sencillo caer en *overflow* o *underflow*, por esta razón, se debe evaluar cada caso. La multiplicación sin corrimientos se muestra en la Tabla 6. Con corrimientos en la Tabla 7. En la Tabla 8 se muestra el resumen de la operación.

Tabla 6: Multiplicación en punto fijo Q7.8, de 1,5 × 3,25 sin corrimientos

Parte operacion	Entera (MSB)	,	Decimal (MSB)	Decimal (LSB)
high (a×c)		,	0000 0011	
mid (a×d+b×c)	000 0001	,	1100 0000	
low (b×d)	010 0000	,	0000 0000	

Tabla 7: Multiplicación en punto fijo Q7.8, de 1,5 × 3,25 con corrimientos

Parte operacion	Entera (MSB)	,	Decimal (MSB)	Decimal (LSB)
high (a×c)<<8	000 0011	,	0000 0000	0000 0000
mid (a×d+b×c)	000 0001	,	1100 0000	0000 0000
low (b×d)>>8	000 0000	,	0010 0000	0000 0000
result (4.875)	000 0100	,	1110 0000	0000 0000

Tabla 8: Multiplicación en punto fijo Q7.8, de 1,5 × 3,25

Número decimal	Signo	Parte entera	Parte fraccionaria
1.5	0	000 0001	1000 0000
3.25	0	000 0011	0100 0000
4.875	0	000 0100	1110 0000

¿Cómo tratar los valores negativos?, ¿Cómo detectar los overflow?

2. Especificación

Se le solicita desarrollar una **arquitectura** y una **microarquitectura** que realice los siguientes dos procesos usando aritmética de punto fijo:

1. Introducción de reverberación a un audio.

2. Reducción de reverberación a un audio.

Cada uno de los dos audios deben tener una duración de por lo menos 7 segundos, la frecuencia de muestreo debe ser de al menos 44,1 kHz (la más común). Los dos audios son los siguientes:

1. Audio sin reverberación.

2. Audio con reverberación conocida (α y k) [2].

El audio sin reverberación debe ser diferente al que tenga reverberación (por ejemplo: dos canciones diferentes, o bien, que una sea una conversación y que otra sea una canción). Se deben seguir los siguientes requisitos generales de funcionalidad:

1. El diseño completo debe poder ser sintetizable en una tarjeta de desarrollo Terasic DE1-SoC-M TL2 (debe caber todo ahí, inclusive los dos audios de entrada).

2. Los dos audios de entrada deben ser almacenados en memoria (se recomienda usar el bloque IP de la biblioteca de Quartus). Debe tomar en cuenta el *buffer* circular.

3. Se deben usar periféricos para seleccionar el modo de operación (insertar o quitar reverberación) y el audio a procesar (reverberado o no reverberado).

4. La salida del sistema se debe reflejar en n pines GPIO en *stream*. Las/los estudiantes deben escoger el número de pines en función del estudio del problema. Esta salida será escrita en un txt que será leído e interpretado por un software de alto nivel libre.

5. El ISA debe ser eficiente y congruente, con criterios de diseño definidos. Es importante pero no mandatorio hacer reuniones con el profesor para guía.

[2] puede usar como base el script proveído anteriormente en este enunciado

2.1. Requisitos de Arquitectura ISA:

1. Debe diseñar un conjunto de instrucciones y arquitectura que permita solucionar el problema planteado, considerando detalles como:

   a) Modos de direccionamiento.

   b) Tamaño y tipo de datos.

   c) Tipo y sintaxis de las instrucciones.

   d) Registros disponibles y sus nombres.

   e) Codificación y descripción funcional de las instrucciones

   Tome en cuenta que estos detalles deben ser justificados desde el punto de vista de diseño (complejidad, costo, área, recursos disponibles). Las/los estudiantes deben realizar un análisis en un software de alto nivel para encontrar los valores idóneos.

2. Las instrucciones a desarrollar son libres así como el tipo de datos. Aunque no hay un límite en cuanto la cantidad de instrucciones, es importante que provea al menos instrucciones para control de flujo, operaciones aritméticas-lógicas, acceso a memoria.

3. El ISA debe ser personalizado y realizado por los estudiantes, no se aceptarán ISAs ya diseñados (e.g., ARM, x86, RISC-V, otros). Debe justificar cada característica del mismo.

4. Los productos finales de esta etapa son los siguientes:

   a) *Instruction reference sheet* o *green card*.

   b) Scripts usados para justificar las características del ISA. **No cuenta como entregable el mismo script proveído por el profesor en este enunciado.**

2.2. Requisitos de Microarquitectura

1. La implementación diseñada debe ser correcta respecto a las reglas definidas por la arquitectura, esto quiere decir que el procesador debe ser capaz de ejecutar todas las instrucciones definidas y su especificación respecto a errores y excepciones.

2. El procesador diseñado debe emplear pipelining. Tenga en cuenta las implicaciones respecto a riesgos de dicha técnica, el uso de registros y unidades de ejecución. **No se revisará si no tiene pipeline.**

3. Debe ser implementado usando SystemVerilog.

4. No se permite realizar módulos especializados de hardware, como los de aritmética de punto fijo. Es un curso de Arquitectura de Computadores, no de Diseño de Sistemas Digitales.

5. El procesador debe tener capacidad de segmentación de memoria en datos e instrucciones además debe ser capaz de acceder los dispositivos de entrada y salida del sistema (GPIO, volcado de memoria, switches, etc).

6. Cada unidad funcional del sistema debe ser debidamente probada en simulación, para verificar su funcionamiento correcto (unit tests). Además debe incluir pruebas de integración y sistema. Se le solicita un plan de pruebas donde especifique los objetivos y descripción de las pruebas junto con sus resultados.

7. Los productos finales de esta etapa son:

   a) El código fuente (SystemVerilog) y el bitstream para programar la tarjeta de desarrollo mediante simulación.

   b) Un diagrama de bloques de la microarquitectura y descripción de las interacciones entre ellos.

   c) Simulaciones de las pruebas unitarias y de integración.

   d) Reporte de consumo de recursos del FPGA para el modelo.

Requisitos de Software:

1. Crear una aplicación (software) empleando la arquitectura diseñada, con el fin de implementar las funcionalidades solicitadas para este proyecto. Debe usar un *buffer* circular.

2. Debe realizar un programa compilador que permita traducir las instrucciones del ISA a binario, con la finalidad de ejecutarlo en el procesador. No es necesario que realice análisis léxico, sintáctico y semántico (este curso no es de Compiladores).

3. Debe desarrollar un programa(s) en alto nivel con las siguientes funcionalidades:

   a) Generar archivos de entrada e inclusión de reverberación.

   b) Reproducir sonidos de entrada y salida.

4. Los productos finales de esta etapa son:

   a) Programa compilador internamente documentado.

   b) Programa de alto nivel para generación de archivos de entrada y reproducción de sonido.

   c) Código ensamblador para la arquitectura diseñada.

El proceso de diseño debe incluir propuestas y comparación de viabilidad de cada una de las tres etapas las mismas.

3. Evaluación y entregables

La defensa será el mismo día de la entrega y todos los archivos (incluyendo código fuente) serán entregados a las 11:59 pm ese mismo día (realícenlo progresivamente y no lo dejen para el final). **Si algo no queda claro o no sabe, no lo asuma, pregúntele al profesor.** Como recomendación se presenta el cronograma de trabajo de la Tabla 9.

Tabla 9: Cronograma sugerido para el proyecto

Semana	Actividades sugeridas
1	Estudio del problema y modelado del programa en alto nivel.
2	Análisis y generación de documentación y scripts para justificar el ISA. En esta parte se va a ir definiendo el *Green Card*.
3	Desarrollo de microarquitectura y pruebas unitarias sobre la misma. Desarrollo del compilador. Se pueden ir desarrollando los diagramas del sistema.
4	Desarrollo final de la microarquitectura. Validación de la arquitectura usando pruebas de integración (arquitectura sobre microarquitectura).
5	Validaciones de integración finales en las tres etapas.

La evaluación del proyecto se da bajos los siguientes rubros contra rúbrica correspondiente:

- Presentación proyecto 100 % funcional (75 %): La defensa se realizará de la siguiente manera: Debido a la situación actual (COVID-19) no se puede tener acceso a hardware u otros instrumentos y medios que requieran presencia y contacto físico tanto entre el profesor como l@s estudiantes. Por esta razón, para la defensa se debe presentar lo siguiente en un espacio de una hora:

    1. Todo el diseño debe ser sintetizable en una tarjeta: **Terasic DE1-SoC-M TL2**. Es decir, debe llegar hasta la generación de un **.sof**. Se garantiza que la tarjeta tenga suficientes recursos para almacenar y ejecutar el diseño según el reporte.
    2. Debe reservar los espacios de memoria para los dos audios.
    3. Mediante ModelSim debe crear un *testbench* de los mismos archivos de SystemVerilog que se usaron para sintetizar. Con este *testbench* debe escribir un archivo con el sonido procesado.
    4. Debe generar un script de alto nivel para reproducir los sonidos de entrada como de salida.

    Los entregables adicionales que se revisarán durante la defensa son los siguientes:

    1. Arquitectura:

   a) *Instruction reference sheet* o *green sheet*.
   b) Scripts usados para justificar las características del ISA. **No cuenta como entregable el mismo script proveído por el profesor en este enunciado.**
2. Microarquitectura:
   a) Diagrama de bloques de la microarquitectura.
   b) Reporte de consumo de recursos del FPGA.
3. Software:
   a) Compilador usado.
   b) Código ensamblador para la arquitectura diseñada.
   c) Programa de alto nivel para generación de archivos de entrada y reproducción de sonido.

- Documentación de diseño (25 %): Este documento se encuentra directamente ligado con el atributo AP. La documentación del diseño deberá contener las siguientes secciones:

  1. Listado de requerimientos del sistema: Cada estudiante deberá determinar los requerimientos de ingeniería del problema planteado, considerando partes involucradas, estado del arte, estándares, normas, entre otros.
  2. Elaboración de opciones de solución al problema: Para el problema planteado deberán documentarse al menos dos opciones de solución. Cada solución deberá ser acompañada de algún tipo de diagrama. **Estas opciones de solución no deben ser fácilmente descartables y deben llevar un análisis objetivo con base en criterios técnicos o teóricos.** Al ser un curso de Arquitectura de Computadores, las opciones deben ser esencialmente basadas en el ISA.
  3. Comparación de opciones de solución: Se deberán comparar explícitamente las opciones de solución, de acuerdo con los requerimientos y otros aspectos aplicables de salud, seguridad, ambientales, económicos, culturales, sociales y de estándares.
  4. Selección de la propuesta final: Se deberá evaluar de forma objetiva, válida y precisa las soluciones planteadas al problema y escoger una solución final.
  5. Archivo tipo README donde especifiquen las herramientas que usaron junto con las instrucciones de uso. **Es un documento README.MD aparte.**

Se seguirán los siguientes lineamientos:

1. Los documentos serán sometidos a control de plagios para eliminar cualquier intento de plagio con trabajos de semestres anteriores, actual o copias textuales, tendrán nota de cero los datos detectados. Se prohíbe el uso de referencias hacia sitios no confiables.

2. No coloque código fuente en los documentos, quita espacio y aporta poco. Mejor explique el código, páselo a pseudocódigo o use un diagrama.