DIGITAL SOUND SYNTHESIS USING PROGRAMMABLE LOGIC DEVICES.

SINTESIS DIGITAL DE SONIDO USANDO DISPOSITIVOS DE LOGICA PROGRAMABLE.

Sergio D. Baron
Universidad Nacional de La Plata
Centro de Técnicas Analógico Digitales (CeTAD)
Calle 48 y 116, La Plata 1900, Argentina
s.d.baron@ieee.org

ABSTRACT

In this work we propose the use of programmable logic devices for digital sound synthesis of musical sounds.

We have studied different types of sound synthesis techniques, we have evaluated which one of these techniques was interesting to be implemented and we proposed the complex programmable logic device (CPLD) as a hardware solution for digital synthesis.

Furthermore we have implemented the Karplus-Strong plucked string algorithm using VHDL and an Altera CPLD. With this implementation we have obtained measurable results and musically significant sounds. We also demonstrate the use of this programmable technology in the field of sound synthesis.

RESUMEN

En este trabajo plateamos el uso de dispositivos de lógica programable para la síntesis digital de sonido con fines musicales.

Se estudió la síntesis digital de sonido en sus distintas manifestaciones, se evaluó cuál de las técnicas estudiadas era interesante para su implementación y se propuso al dispositivo de lógica programable complejo (CPLD) como una solución de hardware para la síntesis digital.

A continuación se implementó el algoritmo de la cuerda tañida de Karplus y Strong usando VHDL y una CPLD de Altera. Mediante esta implementación obtuvimos resultados medibles y apreciables musicalmente. También demostramos en concreto el uso de esta tecnología programable en el campo de la síntesis de sonido.

SINTESIS DIGITAL DE SONIDO USANDO DISPOSITIVOS DE LOGICA PROGRAMABLE

Sergio D. Baron
Universidad Nacional de La Plata
Centro de Técnicas Analógico Digitales (CeTAD)
Calle 48 y 116, La Plata 1900, Argentina
s.d.baron@ieee.org

RESUMEN

En este trabajo plateamos el uso de dispositivos de lógica programable para la síntesis digital de sonido con fines musicales.

Se estudió la síntesis digital de sonido en sus distintas manifestaciones, se evaluó cuál de las técnicas estudiadas era interesante para su implementación y se propuso al dispositivo de lógica programable complejo (CPLD) como una solución de hardware para la síntesis digital.

A continuación se implementó el algoritmo de la cuerda tañida de Karplus y Strong usando VHDL y una CPLD de Altera. Mediante esta implementación obtuvimos resultados medibles y apreciables musicalmente. También demostramos en concreto el uso de esta tecnología programable en el campo de la síntesis de sonido.

1. INTRODUCCION

La síntesis de sonido con fines musicales es un artefacto tecnológico que tiene su razón de ser en la necesidad artística del músico. Planteado de esta manera, el problema de generar sonidos musicales mediante medios electrónicos es resuelto mediante técnicas donde el rigor científico es tan importante como los resultados musicales (que son subjetivos y dependen de la percepción de cada uno).

En este trabajo hemos investigado y experimentado en la materia, obteniendo resultados que son musicalmente aceptables y que además nos proveen de diseños y datos que permiten un avanzar en la implementación de estas técnicas.

2. RESUMEN SOBRE LA SÍNTESIS DE SONIDO CON FINES MUSICALES

En el campo de la síntesis pueden distinguirse dos fuentes de las cuales proviene el sonido: las fuentes sintéticas, que pueden ser electrónicas, electromecánicas, etc. y los sonidos provenientes de la naturaleza o del hombre que son procesados para su uso musical.

Algunos de los métodos de generación sintética de sonido que mencionaremos son: síntesis substractiva, síntesis aditiva y síntesis digital por modulación en frecuencia. Con respecto a las fuentes “naturales” destacaremos el uso de la técnica de muestreo. La generación de sonido mediante el método de “wavetable” podría ubicarse como un paso intermedio, ya que utiliza tablas de muestras de un periodo de duración provenientes generalmente de sonidos naturales, las cuales son leídas y releídas produciendo una forma de onda con características periódicas, la cual es luego procesada como se hace en la síntesis substractiva.

La síntesis substractiva se basa en la generación de una forma de onda rica en armónicos (generalmente cuadrada, triangular, diente de sierra o pulso de ancho variable) para luego ser procesada a través de distintos tipos de filtros (aunque en realidad el tipo de filtro mas usado es el pasa-bajos) y ser modulada en amplitud por una envolvente.

La síntesis aditiva parte de la generación de múltiples sinusoides con diferentes amplitudes, las cuales conformaran el sonido. En general existe algún tipo de control del contenido espectral con respecto al tiempo como así también un control sobre la amplitud total del sonido.

La síntesis digital por modulación en frecuencia se basa en la modulación en frecuencia de sinusoides por otras sinusoides de frecuencia similar, produciendo sonidos ricos armónicamente. En este tipo de sistemas es común encontrar varias fuentes de sinusoides que pueden ser configuradas ya sea como portadoras o moduladoras.

La técnica de muestreo (sampling) es muy utilizada y existen numerosas variaciones alrededor de esta. En general se graba y se muestrea un sonido “natural” pero a diferencia de la técnica de wavetable se almacena el sonido casi por completo, de principio a fin, para luego ser procesada por distintos modificadores como los vistos en la síntesis substractiva.

La síntesis por modelización física (Physical Modeling (PM)), es una técnica que implementa modelos físicos de elementos sonoros como cuerdas, parches y

columnas de aire que oscilan, mediante ecuaciones que describen estos modelos. Esto permite obtener no sólo sonidos musicalmente agradables, sino que también pueden ser controlados en forma intuitiva por el músico.

Todas las técnicas mencionadas son usadas en la actualidad y existen numerosos productos comerciales que se basan en ellas (sintetizadores, samplers, grabadores digitales, sistemas de edición digital de audio, placas de sonido para PC, etc.).

3. SÍNTESIS POR MODELIZACIÓN FÍSICA

3.1. ¿Por qué Modelización Física?

Uno podría preguntarse por qué molestarse en simular instrumentos tradicionales cuando la computadora es capaz de generar cualquier sonido posible. ¿Por qué no nos concentramos principalmente en crear nuevos instrumentos que estén mucho más avanzados que los instrumentos preexistentes?

Una respuesta concisa a esta pregunta es que los sonidos computados artificialmente tienden a sonar artificiales. En otras palabras, no conocemos muchas maneras de generar sonidos intensamente comunicativos a partir de cero.

3.2. Elementos de los modelos físicos para la síntesis de sonido

Para la síntesis digital de sonido por modelización física, se utilizan modelos concentrados y/o distribuidos. Los Modelos concentrados consisten generalmente en masas, resortes, atenuadores y elementos no lineales; que sirven para aproximar sistemas físicos como los labios de un trompetista, las cuerdas vocales de un cantante o los martillos de un piano.

Cuando una masa y un resorte son conectados, un resonador elemental de segundo orden es formado. En el campo del procesamiento digital de audio, un resonador de segundo orden es implementado usando un filtro digital de dos polos. Así resulta que los modelos concentrados son típicamente implementados usando filtros digitales de segundo orden como elementos básicos.

Por otra parte, los modelos distribuidos consisten típicamente de líneas de retardo (muy a menudo llamadas "guías de onda digitales" *"Digital Waveguides"*), en combinación con filtros digitales y elementos no lineales. Se utilizan para simular cuerdas, cornetas, placas, membranas y espacios acústicos. En los modelos que se utilizan *Digital Waveguides*, las perdidas distribuidas y la dispersión son modeladas en forma concentrada en puntos discretos, a través de filtros digitales, separando las líneas de retardo puras que representan el retraso de propagación ideal. Los modelos distribuidos de *waveguides* pueden combinarse libremente con modelos concentrados; por ejemplo el modelo de un instrumento de viento consiste típicamente de un modelo concentrado para la "lengüeta-labio" y una *waveguide* distribuida para modelar la corneta.

4. IMPLEMENTACIÓN DE LOS ALGORITMOS DE SÍNTESIS POR MODELIZACIÓN FÍSICA EN VHDL Y CPLD

En el proceso de implementación de los algoritmos de PM en VHDL [1] nos encontramos con dos tareas. Una es la de mapear los elementos básicos de procesamiento de señales a su equivalente en hardware, y la otra es la de programar el algoritmo en sí mismo utilizando estos bloques constructivos. Debido a que muchos de los elementos utilizados en la síntesis PM son comunes a todo tipo de algoritmos, usamos la habilidad que posee el VHDL de generar bibliotecas de componentes particulares, que en su momento son llamadas desde el modelo VHDL principal.

Cuando una gran biblioteca de elementos básicos para la síntesis PM es completada, uno puede programar casi cualquier motor de síntesis utilizando un número relativamente pequeño de líneas de código.

4.1. La Lógica Programable

Para la implementación en hardware usamos dispositivos Altera [2]. Elegimos específicamente esta marca debido a la experiencia previa con ella, el conocimiento de la arquitectura de las diferentes familias de estos dispositivos y principalmente porque posee un ambiente de programación muy efectivo.

Es claro que necesitamos una arquitectura microgranular con una capacidad de ruteo de señales muy flexible, de tal manera que elegimos la familia FLEX de Altera para nuestra aplicación. Estos chips programables usan una memoria de configuración externa, esta memoria puede ser EEPROM o RAM. Entonces es posible usar una memoria RAM de doble puerta para la configuración, haciendo el sistema reprogramable instantáneamente y posibilitando su uso en una plaqueta de síntesis para PC o un sintetizador independiente con capacidad de cambiar de un timbre a otro al presionar un botón.

5. IMPLEMENTACIÓN DEL ALGORITMO DE LA CUERDA TAÑIDA DE KARPLUS Y STRONG

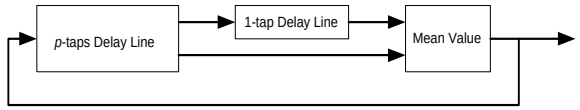
La elección lógica para esta implementación es la de usar el algoritmo de Karplus y Strong (K-S) [3]: este algoritmo dio origen a la teoría moderna de la síntesis que usa el *waveguide filter* [4] y además es simple de implementar.

5.1.- El algoritmo básico de Karplus y Strong

El algoritmo de Karplus y Strong fue concebido originalmente como una simple modificación a la muy conocida síntesis de *wavetable*. Los autores de este algoritmo agregaron un modificador compuesto de un filtro FIR (en la forma de un operador de promediación) y un lazo de realimentación. Esto se representa en la siguiente función:

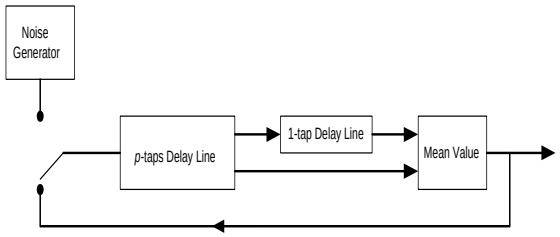
$Y_t = {}^1/_2 \ (Y_{t-p} + Y_{t-p-1})$

Y se puede ver en la siguiente figura:



5.2.-Mapeo del algoritmo a Hardware usando VHDL.

Para este ejemplo hemos implementado el algoritmo que se muestra en la siguiente figura:



5.2.1.- La Biblioteca

Hemos compilado una biblioteca básica donde algunos componentes fueron implementados:

```
------------------------------------------------------
    LIBRARY ieee;
    USE ieee.std_logic_1164.all;
------------------------------------------------------
    PACKAGE pm IS
      COMPONENT Adder8a
        PORT(dataa,datab:IN STD_LOGIC_VECTOR(7 DOWNTO 0);
             aclr:       IN STD_LOGIC := '0';
             clock:      IN STD_LOGIC := '0';
             cin:        IN STD_LOGIC := '0';
             add_sub:    IN STD_LOGIC := '1';
             result:     OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
             cout, overflow: OUT STD_LOGIC);
      END COMPONENT;

      COMPONENT Mux8
          PORT(Noisy,Plunky:IN STD_LOGIC_VECTOR(7 DOWNTO 0);
               selm1:        IN STD_LOGIC;
               Outy:        OUT STD_LOGIC_VECTOR(7 DOWNTO 0));
      END COMPONENT;

      COMPONENT Divis
          PORT(D:            IN STD_LOGIC_VECTOR(7 DOWNTO 0);
               cin:          IN STD_LOGIC;
               pre:          IN STD_LOGIC;
               clr:          IN STD_LOGIC;
               load:         IN STD_LOGIC;
               res:          OUT STD_LOGIC;
               Q:            INOUT STD_LOGIC_VECTOR(7 DOWNTO 0));
      END COMPONENT;

      COMPONENT Shifter
          PORT(D:   IN        STD_LOGIC_VECTOR(7 DOWNTO 0);
               pre: IN        STD_LOGIC;
               clr: IN        STD_LOGIC;
               load:IN        STD_LOGIC;
               Q:   OUT       STD_LOGIC_VECTOR(7 DOWNTO 0));
      END COMPONENT;

      COMPONENT barrelns
          PORT (D:    IN      STD_LOGIC_VECTOR(7 downto 0);
                pre:       IN STD_LOGIC;
                clr:       IN STD_LOGIC;
                load:      IN STD_LOGIC;

                Q:    OUT     STD_LOGIC_VECTOR(7 downto 0));
      END COMPONENT;

   END pm;
```

Adder8a es un sumador de ocho bits con carry-out proveniente de la biblioteca de Altera. Este es usado en la operación de promediación.

Mux8 es un multiplexador de ocho bits con dos entradas y una salida, el cual es usado para seleccionar la entrada a la línea de retardo de p-etapas como se ve en la figura.

Divis es un registro de nueve bits que divide su entrada por 2. Es usado en la operación de promediación.

Shifter es un registro de desplazamiento de ocho bits usado para mapear la línea de retardo de p-etapas.

Barrelns es un registro de desplazamiento de ocho bits usado para mapear la línea de retardo de una etapa.

5.2.2.- Implementación de la arquitectura del algoritmo K-S.

Este es el código VHDL estructural que describe nuestra implementación de la arquitectura de nuestro algoritmo:

```
------------------------------------------------------------
-- K-S algorithm.
------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.all;
library work;
USE work.pm.all;
------------------------------------------------------------
ENTITY Pmunit2 IS
         PORT( selmux              : IN          STD_LOGIC;
               preshi,clrshi,ldshi : IN          STD_LOGIC;
               prebns,clrbns,ldbns : IN          STD_LOGIC;
               zero                : IN          STD_LOGIC;
```

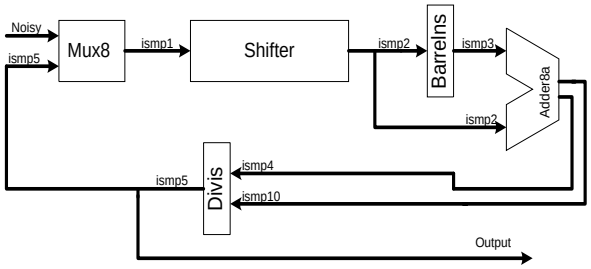
```
                predv,clrdv,lddv    : IN          STD_LOGIC;
                aclr,clock,add_sub  : IN       STD_LOGIC;
                Noisy               : IN STD_LOGIC_VECTOR(7
downto 0);
                overflow,res        : OUT         STD_LOGIC;
                Q                   : OUT STD_LOGIC_VECTOR(7
downto 0));
          END Pmunit2;

ARCHITECTURE structure OF Pmunit2 IS
          SIGNAL ismp1 : STD_LOGIC_VECTOR(7 downto 0);
          SIGNAL ismp2 : STD_LOGIC_VECTOR(7 downto 0);
          SIGNAL ismp3 : STD_LOGIC_VECTOR(7 downto 0);
          SIGNAL ismp4 : STD_LOGIC_VECTOR(7 downto 0);
          SIGNAL ismp5 : STD_LOGIC_VECTOR(7 downto 0);
          SIGNAL ismp10 : STD_LOGIC;
          SIGNAL ismp11 : STD_LOGIC;

BEGIN

m8:  mux8 PORT MAP(noisy,ismp5,selmux,ismp1);
sh:  shifter PORT MAP(ismp1,preshi,clrshi,ldshi,ismp2);
bns: barrelns PORT MAP(ismp2,prebns,clrbns,ldbns,ismp3);
a8:  adder8a PORT MAP
(ismp2,ismp3,aclr,clock,zero,add_sub,smp4,ismp10,overflow);
d: divis PORT MAP(ismp4,ismp10,predv,clrdv,lddv,ismp11,ismp5);

Q<= ismp5;
res<=ismp11;

END structure;
```

Como se puede ver en éste código, la arquitectura principal utiliza los componentes de la biblioteca y mapea las entradas y salidas de cada uno para generar el algoritmo K-S.

El código correspondiente a los elementos de la biblioteca puede verse en:

http://www.wap.com.ar/web/cpldpm.html

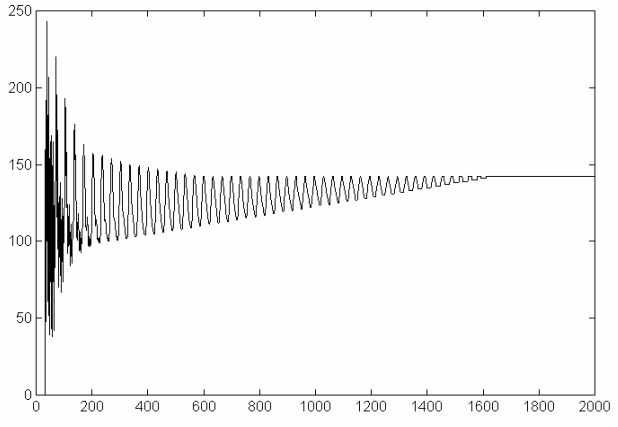
La implementación de esta arquitectura se visualiza en la siguiente figura:



5.3. Comprobación

En el transcurso de esta implementación usamos la habilidad del ambiente de desarrollo para comprobar el funcionamiento del diseño mapeando el mismo directamente a una CPLD en particular, de tal manera que la coincidencia de la salida de la simulación y lo que se obtendrá del hardware está garantizada.

Hemos pre-cargado la línea de retardo con una forma de onda aleatoria y hemos simulado el circuito muchas veces, obteniendo mayoritariamente sonidos de cuerda tañida muy agradables al oído. En la figura se puede ver una representación temporal de uno de estos sonidos:



Ejemplos de audio en:
http://www.wap.com.ar/web/cpldpm.html

6. CONCLUSIONES

Se han estudiado distintas técnicas de síntesis digital de sonido y se ha elegido la síntesis por modelización física para su implementación en hardware. El algoritmo de Karplus y Strong se utilizó como ejemplo ilustrativo de los que es la técnica de guías de onda digitales para la síntesis PM.

Hemos propuesto el uso del VHDL y la lógica programable para la síntesis por modelización física. Una biblioteca básica de componentes PM fue compilada y una arquitectura del algoritmo de Karplus y Strong fue diseñada.

Mientras trabajamos en la implementación nos encontramos con obstáculos como problemas de sincronización, glitches, desbordes aritméticos, etc. Ninguno de los cuales son encontrados en las realizaciones en software. Sin embargo tenemos un sistema, que para el caso particular del algoritmo K-S, es capaz de generar una muestra por cada ciclo de clock.

7. REFERENCIAS

[1] Compass Design Automation, VHDL Scout: A Practical Introduction to IEEE Std 1067-1987, the VHSIC Hardware Description Language.

[2] http://www.altera.com.

[3] Kevin Karplus, Alex Strong, “Digital Synthesis of Plucked-String and Drum Timbres”, Computer Music Journal, Vol. 7, No. 2, summer 1983.

[4] Julius O. Smith III, “Music Applications of Digital Waveguides”, Technical Report Stan-M-39, CCRMA, Department of Music, Stanford University. Stanford, California.