EVALUACIÓN DE UN MODELO PARA LA IMPLEMENTACIÓN DE LA FFT SOBRE FPGA UTILIZANDO UNA HERRAMIENTA DE SÍNTESIS DE ALTO NIVEL

Mauricio Marzán Arcila

Ingeniería Electrónica

Director(es) del trabajo de grado

Luis Fernando Castaño

INSTITUTO TECNOLÓGICO METROPOLITANO

MEDELLIN - COLOMBIA

Junio 22 de 2018

RESUMEN

Una de las herramientas utilizadas para el desarrollo de sistemas basados en FPGA es el Vivado HLS, con la cual se realiza parte del diseño con un lenguaje de alto nivel. En este trabajo se implementa un programa que permite realizar la FFT en Vivado HLS. Para esto se realiza un código en C++ para obtener como resultado la FFT de una señal de entrada. Se realizan simulaciones con diferentes señales para verificar que se logra la identificación de cada una de las frecuencias incluidas en ella. Al momento de la generación del Bitstream no se puede completar la implementación hardware debido a un error en uno de los módulos. Un aspecto de mejora para trabajo futuro es la corrección de este inconveniente para completar la implementación. Vivado HLS resulta ser una gran herramienta a la hora de programar un FPGA ya que evita utilizar el lenguaje de descripción de hardware el cual puede tardar mucho tiempo su implementación.

Palabras claves: FPGA, FFT, Vivado HLS, ZedBoard, C/C++ .

RECONOCIMIENTOS

Especialmente quiero agradecer a Dios por su respaldo para realizar al realizar este trabajo, a mi esposa y mi familia por su comprensión en este proceso. Agradecer al profesor Luis Fernando Castaño por su asesoría en el desarrollo del producto, al profesor David Márquez, al laboratorista Cristian Alzate y en general al laboratorio sistemas de control y robótica por todo su apoyo.

ACRÓNIMOS

FPGA (Field Programmable Gate Array): Matriz de puertas programables en campo.

HLS (High Level Sisntesis): Síntesis de alto nivel.

FFT (Fast Fourier Transform): Transformada Rápida de Fourier.

RISC (Reduced Instruction Set Computer): Ordenador con Conjunto Reducido de Instrucciones.

FPGA (Field-Programmable Gate Array): Matriz de puertas programables.

RTL (Registration Transfer Level): Nivel de transferencia de registro.

DSP (Digital signal processor): Procesador digital de señales.

GPU (Graphics Processing Unit): Unidad de procesamiento gráfico.

TABLA DE CONTENIDO

TABLA DE FIGURAS

1. INTRODUCCIÓN

Los FPGA son dispositivos electrónicos que se programan utilizando un leguaje de descripción de hardware. En comparación a los leguajes de programación de software, con la utilización del HDLs se puede tardar mucho tiempo en realizar una implementación. Vivado HLS permite agilizar este proceso porque permite describir módulos en leguaje C o C++.

En este trabajo se emplea la herramienta Vivado HLS para generar un módulo que realiza la FFT. Simulando la entrada de diferentes señales para luego comparar el resultado obtenido con el de la herramienta de Matlab, en la cual se hacen pruebas con los mismos datos.

La FFT ocupa un papel importante en el procesamiento de señales, ya que permite realizar el análisis en el espectro de la frecuencia e identificar ruidos en las señales, para luego ser filtrados. A través del desarrollo de este trabajo se busca estudiar la utilización de la herramienta FFT del Vivado HLS.

2. MARCO TEÓRICO

En los sistemas de comunicaciones electrónicos es significativo tener un buen procesamiento de las señales. Una señal periódica en el tiempo puede ser expresada mediante la serie de Fourier. Es allí donde la transformada de Fourier tiene gran aplicación, debido a que permite analizar las señales en el espectro de la frecuencia para luego ser procesadas. La FFT divide el número de muestras en términos pares e impares, logrando una alta eficiencia al momento de hallar el espectro de potencia de una señal.

Huan Li y Wenhua Ye (2016), demuestran la eficiencia de una FPGA utilizando Vivado HLS y realizan la comparación con otros procesadores de señales digitales como el x86, DSP y GPU. En este informe se puede evidenciar que el FPGA tiene mejor respuesta en rendimiento y tiempo cuando se utiliza la síntesis de alto nivel.

Acerca de la FFT y su inversa (IFFT) se han realizado investigaciones sobre cómo optimizar la implementación en hardware como en el trabajo presentado por Salaskar & Chandrachoodan (2016). Los autores lograron una reducción en la latencia, utilizado ciclos anidados para minimizar la sobre carga de estos.

En el área de las telecomunicaciones como la radio y otras en las cuales se aplique la multiplicación por frecuencias ortogonales (OFDM), se espera que los desarrollos en la tecnología FPGA tengan una aplicabilidad significativa. Estos pueden reducir el tiempo de diseño usando la herramienta de síntesis de alto nivel. Tran, M. T., Casseau, E., & Gautier (2016) efectúan una implementación flexible de la FFT que permite realizar el cálculo utilizando diferentes tamaños como 128, 256, 512, 1024, 1536, 2048, valiéndose de un módulo de control para variar entre ellos.

3. METODOLOGÍA

3.1. Aplicación de la FFT en Vivado HLS

Se implementa un programa tomando el ejemplo “fft_single” de la herramienta Vivado HLS, el cual se divide básicamente en dos partes. Un archivo principal que contiene la operación de la FFT con diferentes paramentos. Otro archivo para la simulación o prueba de funcionamiento del principal, en el cual se definen los parámetros y las señales que se van a analizar.

Manejo de los datos de entrada para la simulación

Para configurar la función de la FFT se dejan valores fijos, igualando a cero todos los parámetros exceptuando NFFT, el cual se deja con valor de 10 para procesar 1024 datos de la señal de entrada. La señal se crea utilizando Matlab para generar los valores con el formato correcto, en representación de punto fijo y normalizados. La entrada tiene un componente real e imaginario.

Programa principal que realiza la FFT

En este programa solo se modifican dos variables booleanas “direction” y “ovflo”, estas pasan de ser variables locales a globales, ya que se necesita que sean de este tipo para ejecutar el programa de forma correcta.

Simulación de la FFT y salida de los datos

En Vivado HLS se realiza la simulación con el objetivo de calcular la FFT de la señal ingresada, la cual entrega dos valores uno real y otro imaginario. Se les da el formato a los datos de salida para posteriormente ser visualizados en Matlab. Se realizan simulaciones con una onda seno de 10Hz y con una señal generada con la suma de tres ondas seno de 10Hz,150Hz y 400Hz. Para validar los resultados se realiza la comparación entre las respuestas obtenidas con el Vivado HLS y MATLAB.

3.2. Directivas para la implementación del programa en la FPGA

Para realizar la implementación en hardware del programa realizado en Vivado HLS, se utiliza la interface BRAM. La cual para ser utilizada se adicionan las siguientes directivas, las cuales sirven para crear los puertos de entrada y salida, utilizar un solo puerto para cada BRAM, empaquetar la salida que contiene dos números de 16 Bits (Real e imaginario) en un solo número de 32 Bits.

```
#pragma HLS INTERFACE bram port=out
#pragma HLS INTERFACE bram port=in
#pragma HLS RESOURCE variable=out core=RAM_1P_BRAM
#pragma HLS RESOURCE variable=in core=RAM_1P_BRAM
#pragma HLS data_pack variable=in
#pragma HLS data_pack variable=out
#pragma HLS INTERFACE s_axilite port=return bundle=CRTL_BUS
```

Con las directivas creadas se sintetiza y se exporta el archivo RTL para implementar el programa en el FPGA. Con este archivo generado podemos utilizar Vivado para diseñar la implementación del BRAM (Figura 1.), interconectando los puertos de los módulos ZYNQ, BRAM, Controlador BRAM y el IP, que contiene el programa exportado desde Vivado HLS.

Validación del diseño y generación del Bitstream

Después de realizar la validación del diseño, la prueba resulta correcta. Sin embargo, al generar el Bitstream surge un problema en el módulo “fft_config1_U0” el cual no se logra identificar concretamente para dar la solución. Por esta razón no se puede llevar a cabo la implementación en hardware.

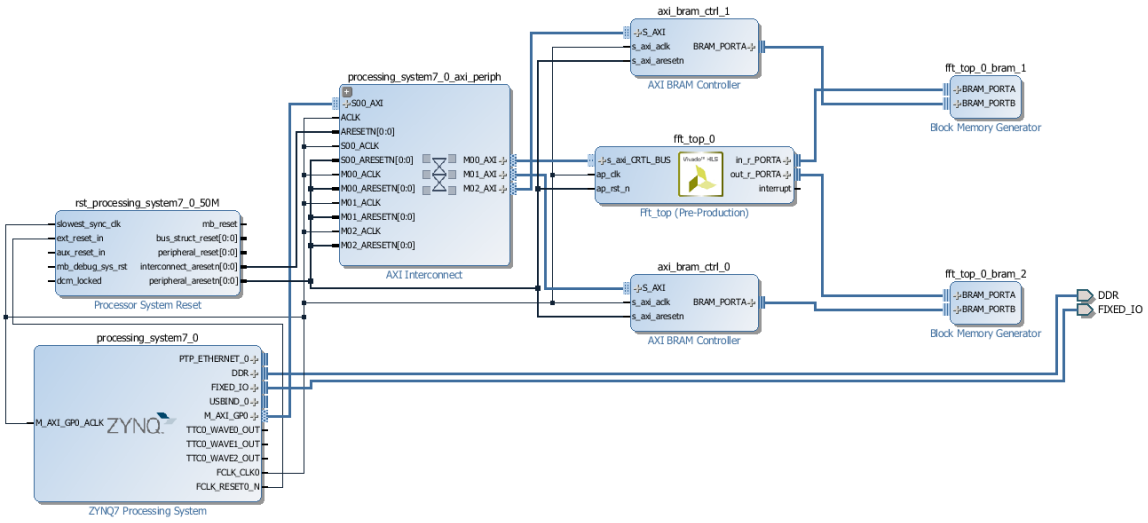

Figura 1. Diseño para la implementación BRAM.

4.RESULTADOS Y DISCUSIÓN

Resultados para una señal

Se utiliza una onda seno de 10Hz como señal de entada, como se ve en la Figura 2. Esta es procesada utilizando Vivado HLS para obtener como señal de salida la Figura 3, en la cual se muestra el espectro de potencia.

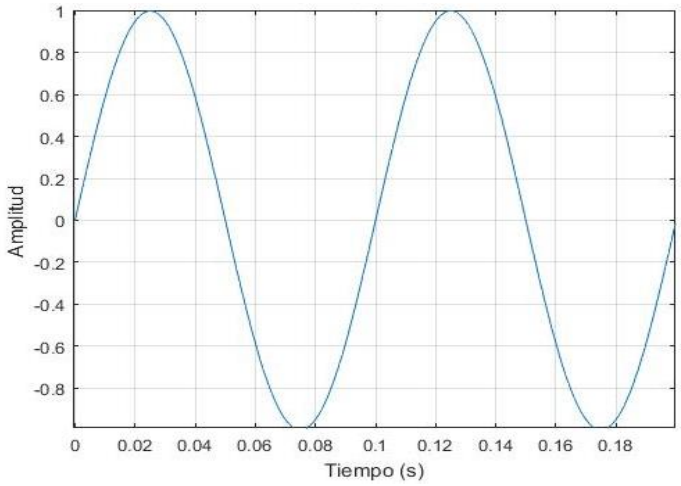

Figura 2. Onda Seno de 10Hz.

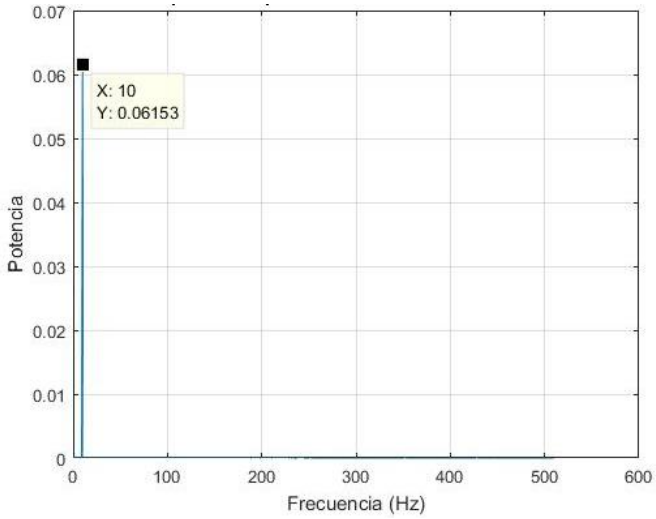

Figura 3. Espectro de potencia obtenido en Vivado HLS

Para comprobar este resultado se crea otro programa en Matlab con el objetivo de validar los datos obtenidos en Vivado HLS. En la Figura 4 podemos observar la gráfica de los datos de este programa, donde se evidencia el mismo pico de potencia a los 10Hz, mismo valor de frecuencia de la señal de entrada. Con lo cual se puede decir que la simulación se está realizando de forma correcta.

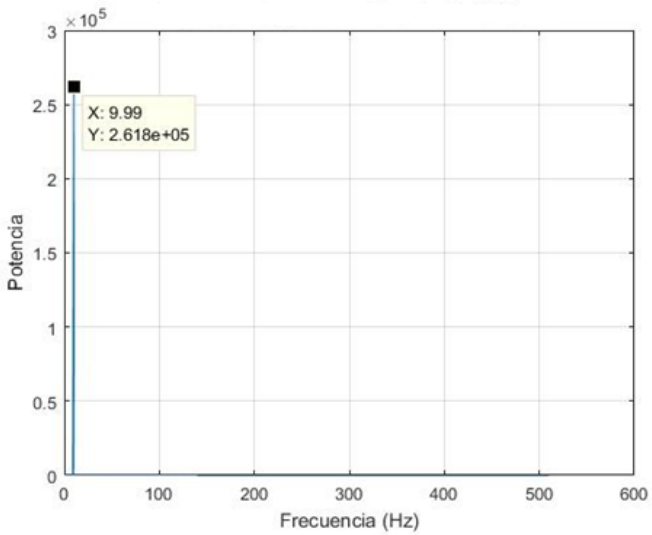

Figura 4. Espectro de potencia obtenido en Matlab

Resultados para una señal conformada por la suma de ondas seno

Se prueba con la suma de señales seno a 10Hz,150Hz y 400Hz. En la Figura 5 se muestra la señal resultante con la suma de las ondas seno.

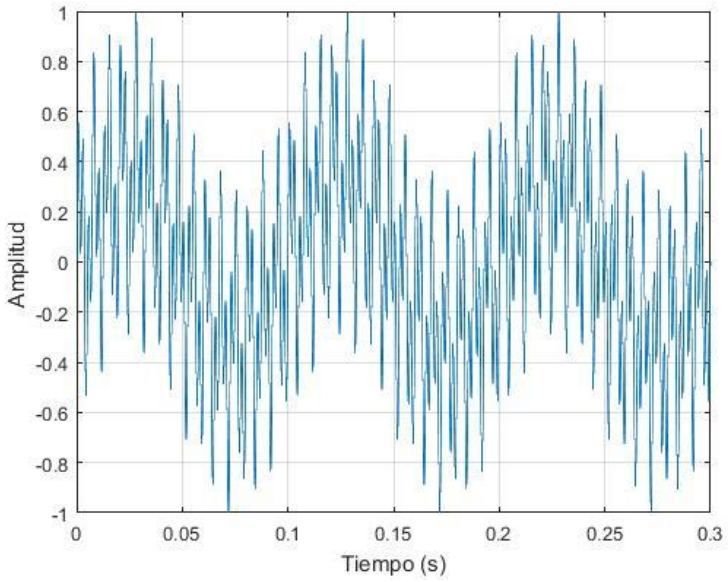

Figura 5. Suma de ondas seno a 10Hz, 150Hz y 400Hz

Al analizar la señal con la FFT se obtiene el resultado mostrado en la Figura 6, en la cual se observa que se pueden identificar claramente las frecuencias de las ondas seno incluidas en la señal de entrada.

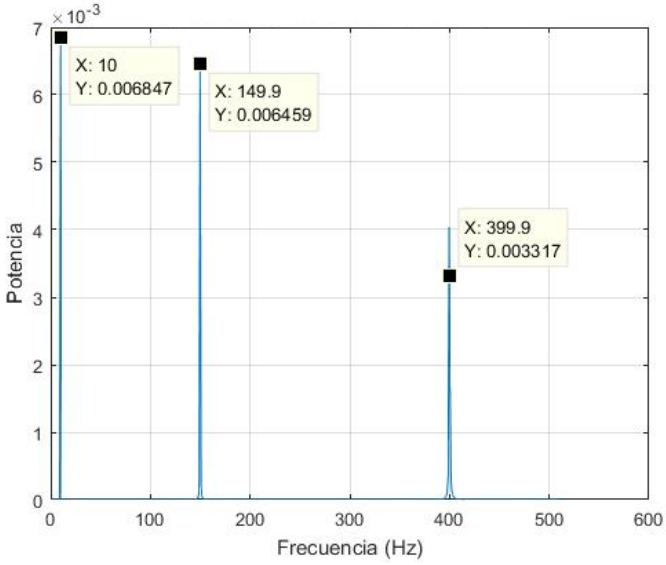

Figura 6. Espectro de potencia de suma de ondas seno a 10Hz, 150Hz y 400Hz.

5. CONCLUSIONES, RECOMENDACIONES Y TRABAJO FUTURO

5.1. Conclusiones

Se modifica un programa en C++, el cual permite realizar la FFT en un leguaje de alto nivel, logrando reducir el tiempo de programación comparando con un leguaje de descripción de hardware, el cual se utiliza en las FPGA.

Se utiliza la herramienta de Vivado HLS para realizar la exportación del RTL, con el fin de tener un módulo que se pueda implementar en un FPGA.

El análisis de los resultados obtenidos de la FFT en Vivado HLS es satisfactorio, ya que al compararlos con resultados de procesar las mismas señales de entrada en otro programa como lo es Matlab, se puede notar que corresponden a la misma señal.

Al realizar la suma de señales seno la respuesta hallada de la FFT corresponde al espectro de potencia de la señal ingresado, diferenciando cada una de las frecuencias incluidas en dicha señal.

Se logra la evaluación de un modelo de la FFT para implementar sobre un FPGA utilizando Vivado HLS, el cual facilita el análisis en frecuencia de la entrada.

5.2. Recomendaciones

Al realizar implementaciones en hardware, como en este caso que se deseaba realizar la FFT de una señal utilizando una FPGA, se debe tener en cuenta las funciones recursivas, estas son funciones que se utilizan a sí mismas y en hardware esto puede causar un error. Esta recomendación es debido a que en uno de los primeros programas ensayados tenían funciones recursivas y no fue posible su implementación.

5.3. Trabajos futuros

Teniendo el módulo que realiza la FFT implementado en software mediante la herramienta de Vivado, se puede continuar con la implementación en hardware utilizando un FPGA, lo cual podría ser importante en aplicaciones de comunicación, en la etapa de procesamiento de las señales.

REFERENCIAS

A. Salaskar and N. Chandrachoodan, "FFT/IFFT implementation using Vivado™ HLS," 2016 20th International Symposium on VLSI Design and Test (VDAT), Guwahati, 2016, pp. 1-2. doi: 10.1109/ISVDAT.2016.8064896

Huan Li and Wenhua Ye, "Efficient implementation of FPGA based on Vivado High Level Synthesis," 2016 2nd IEEE International Conference on Computer and Communications (ICCC), Chengdu, 2016, pp. 2810-2813.
doi: 10.1109/CompComm.2016.7925210

Tran, M. T., Casseau, E., & Gautier, M. (2016, October). Demo abstract: FPGA-based implementation of a flexible FFT dedicated to LTE standard. In Design and Architectures for Signal and Image Processing (DASIP), 2016 Conference on (pp. 241-242). IEEE.
doi: 10.1109/DASIP.2016.7853833

APÉNDICE A

Código de Test Bench

```
#include <iostream>
#include <cstdlib>
#include <cstring>
#include <sstream>
#include "fft_top.h"
#include <stdio.h>
#include <fstream>
#include <string>
#include <sstream>

using namespace std;

#define BUF_SIZE 64
int main()
{
    const int SAMPLES = (1 << FFT_NFFT_MAX);
    static cmpxDataIn xn_input[SAMPLES];
    static cmpxDataOut xk_output[SAMPLES];
        int NFFT = 10;
        int CP_LEN = 0;
        int FWD_INV = 0;
        int sc_sch = 0;
        float input_real,input_imag;
        FILE *stimfile;

        // Abrir el archivo datos.txt
        stimfile = fopen("datos.txt", "r");
        if( stimfile )
             printf( "existe (ABIERTO)\n" );
          else
          {
             printf( "Error (NO ABIERTO)\n" );
             return 1;
          }
        // Ciclo para lectura de datos
        for (int i=1;i<1024;i++ )
        {
        fscanf(stimfile,"%f %f",&input_real,&input_imag);
        xn_input[i].real()=(data_in_t)input_real;
        xn_input[i].imag()=(data_in_t)input_imag;
        }// fin ciclo de lectura de datos


        fclose(stimfile); // Cerrar el archivo



       //  Entra a la funcion FFT_TOP y Realiza la FFT
        //fft_top(FWD_INV, xn_input, xk_output,&ovflo);
        fft_top( xn_input, xk_output);
        // Salida de la Funcion
       for (int i=0;i<1024;i++)
          cout <<  xk_output[i].real().to_float() << ","  << xk_output[i].imag().to_float()
<<endl;
         // cout <<  FWD_INV <<endl;
     return 0;
}
```

APÉNDICE B

Código Principal

```
#include "fft_top.h"

void dummy_proc_fe(
    bool direction,
    config_t* config,
    cmpxDataIn in[FFT_LENGTH],
    cmpxDataIn out[FFT_LENGTH])
{
    int i;
    config->setDir(direction);
    config->setSch(0x2AB);
    for (i=0; i< FFT_LENGTH; i++)
        out[i] = in[i];
}

void dummy_proc_be(
    status_t* status_in,
    cmpxDataOut in[FFT_LENGTH],
    cmpxDataOut out[FFT_LENGTH])
{
    int i;
    for (i=0; i< FFT_LENGTH; i++)
        out[i] = in[i];
}

void fft_top(
       complex<data_in_t> in[FFT_LENGTH],
    complex<data_out_t> out[FFT_LENGTH])
{
#pragma HLS RESOURCE variable=out core=RAM_1P_BRAM // Para que salga solo PORTA
#pragma HLS RESOURCE variable=in core=RAM_1P_BRAM // Para que salga solo PORTA
#pragma HLS INTERFACE bram port=out
#pragma HLS INTERFACE bram port=in
#pragma HLS INTERFACE s_axilite port=return bundle=CRTL_BUS

#pragma HLS data_pack variable=in  // Compacta el dato real con el imaginario
#pragma HLS data_pack variable=out // Compacta el dato real con el imaginario
#pragma HLS dataflow
       bool direction = 0; //Agregado
    complex<data_in_t> xn[FFT_LENGTH];
    complex<data_out_t> xk[FFT_LENGTH];
    config_t fft_config;
    status_t fft_status;

    dummy_proc_fe(direction, &fft_config, in, xn);
    // FFT IP
    hls::fft<config1>(xn, xk, &fft_status, &fft_config);
    dummy_proc_be(&fft_status,xk, out); // Borrado ccc
}
```

FIRMA ESTUDIANTES Mauricio Marzán Arcila

FIRMA ASESOR Luis Fernando Cortés Londoño

FECHA ENTREGA: 26-06-2018

FIRMA COMITÉ TRABAJO DE GRADO DE LA FACULTAD _____

RECHAZADO___ ACEPTADO____ ACEPTADO CON MODIFICACIONES_____

ACTA NO._____

FECHA ENTREGA: _____

FIRMA CONSEJO DE FACULTAD_____

ACTA NO._____

FECHA ENTREGA: _____