Instituto Tecnológico de Costa Rica – Laboratorio de Diseño de Circuitos Integrados (DCILab)
Manual de uso para ambiente de verificación en Python para la Unidad de Coma Flotante en Verilog – Version 0.3 (1/7/16)
Francis Alexander Lopez Montero – falm81427@gmail.com

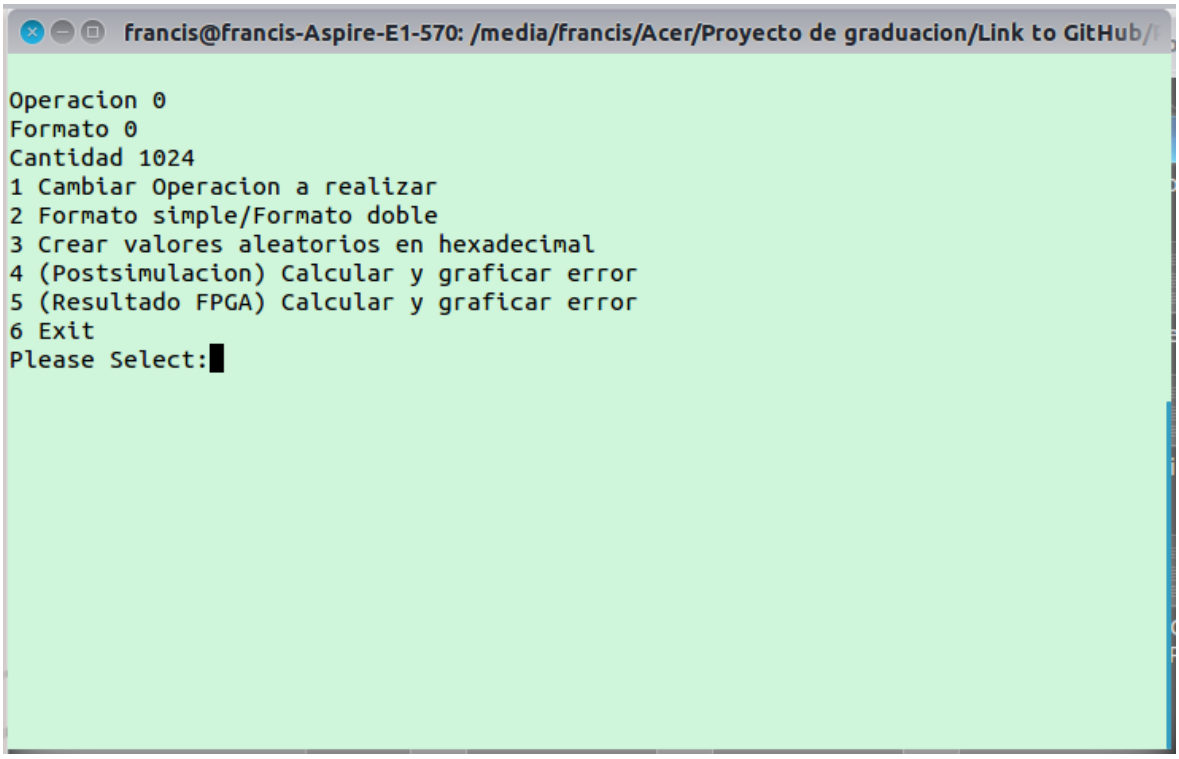
Descripción

En el siguiente documento se hablará acerca de uso del programa creado en python para la generación de vectores de prueba y comparación de errores que fue usado para la verificación de la unidad aritmética de coma flotante para el DCILab. Este documento se menciona tanto del uso del programa como de la descripción del código para futuras expansiones/mejoras que se le desean agregar.

Requisitos

- El programa y este documento está pensado para el sistema operativo Ubuntu (15.40 o superior), no se asegura su funcionamiento en Windows
- Programa Python
- Programa matemático (Pensado para Octave)
- Instalar paquete *oct2py* (#$pip install oct2py)
- Instalar paquete *scipy* (#sudo apt-get install python-scipy)
- Instalar complemento para octave *instrument-control*
- Archivos testbench en HDL de los modulos de prueba en Verilog y el Modulo UART para transmisión USB

Menu principal



El programa cuenta con 3 valores precargados con selecciones default para la selección del tipo de datos aleatorios para generar:

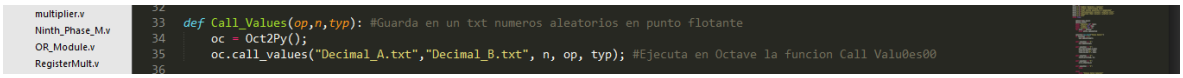
- Operación *(Se cambia usando la opción 1)*
  - 0: Suma
  - 1: Resta
  - 2: Multiplicación
- Formato de precisión del valor en decimal: *(Se cambia usando la opción 2)*
  - 0: Precisión simple (32 bits)
  - 1: Precisión doble (64 bits)
- Valor definido para cantidad de valores generados (definidos 1024 valores)

```
def main():

    oper=0;#Suma=0;Resta=1
    typ=0#Formato simple=0;Formato doble=1
    n=1024;#Cantidad de datos

    menu = {}
    menu['1']="Cambiar Operacion a realizar"
    menu['2']="Formato simple/Formato doble"
    menu['3']="Crear valores aleatorios en hexadecimal"
    menu['4']="(Postsimulacion) Calcular y graficar error"
    menu['5']="(Resultado FPGA) Calcular y graficar error"
    menu['6']="Exit"
    while True:

        options=menu.keys()
        options.sort()
        print "Operacion %d" %oper;
        print "Formato %d" %typ;
        print "Cantidad %d" %n;
        for entry in options:
            print entry, menu[entry]

        selection=raw_input("Please Select:")
        if selection =='1':
            oper=addsubt(oper);

        elif selection == '2':
            typ=simpdob(typ);

        elif selection == '3':
            Call_Values(oper,n,typ);
            float_to_hex("A", typ);
            float_to_hex("B", typ);
```

El código presenta en pantalla las opciones a elegir y los valores seleccionados. Antes de generar los valores y realizar la comprobación de los errores se debe realizar la selección respectiva y se debe mantener durante todos los pasos ya que puede llevar a errores tanto en los cálculos como en el funcionamiento del programa.

Generar valores aleatorios en hexadecimal

El programa genera por defecto 3 archivos “txt”: “Decimal_A” y “Decimal_B” para cargar los operandos que fueron generados para el cálculo de la operación y “Decimal_R” con el resultado teórico de la operación seleccionada.



Los nombres de estos archivos se referencian en la función *call_values*, que llama a la rutina octave con los datos de los nombres de los archivos txt asi como los parámetros seleccionados para la operación seleccionada. En esta rutina se calcula el resultado según la operación seleccionada y se guarda tanto el valor decimal como el hexadecimal en archivos apartes.

Para utilizar los valores generados en los testbenchs de verilog se deben añadir en el Xilinx utilizando "Add_sources" los archivos en hexadecimal. (Solo se necesita hacer una vez ya que al actualizarse los archivos, se utilizan los nuevos valores generados)

Generar resultado de simulación de Vivado

Actualmente este programa en Python no cuenta con las líneas de código para generar las simulaciones desde este archivo, por lo que para obtener el archivo de resultado experimental (**ResultadoXilinx*.txt)** se debe extraer de la carpeta de la simulación generada (*carpeta_del_proyecto/proyecto*.sim/sim_1/*simulación_realizada/* **ResultadoXilinx*.txt)** y copiarla en la carpeta donde se encuentra el script del programa de Python.

Calcular porcentaje de error post-simulación (finalresult.m)

Este archivo lo que realiza es el calculo del porcentaje de error del resultado de la simulación realizada (**ResultadoXilinxFLM.txt)** en comparación con dos valores tomados de diferentes referencias (el valor teorico que fue calculado con octave en **call_values.m (hexadecimal_R.txt)** y el valor calculado desde otra simulación (**ResultadoXilinxDRV.txt)**) en donde se realiza la conversión del valor hexadecimal a decimal (se hace un acondicionamiento de los datos para eliminar el salto de línea que realiza el programa Vivado) y el calculo de los porcentajes de error para ser cargados en archivos .txt (**FLMvsTeorico.txt** y **FLMvsDRV.txt)**

Obtener valores a partir de la FPGA (serialcom.m/serialcom32.m)

Para la captura de los datos de la FPGA es necesario agregar el modulo a utilizar en el modulo FPU_uart y cargarlo en la FPGA. Este modulo cuenta con que al presionar un botón, se envía todos los resultados de la operación a partir de las ROM cargadas con los archivos .txt precargadas en el módulo vía USB.

```
76
77  def error_FPGA(typ,n):
78      oc = Oct2Py();
79      print "Presionar botón en la FPGA para ejecutar";
80      if typ==0:#Formato simple
81          oc.serialcom32("/dev/ttyUSB1",9600,4096,typ);
82
83      elif typ==1:#Formato doble
84          oc.serialcom("/dev/ttyUSB1",9600,8192,typ);
85      oc.grapgresult(1);
86
```

En el archivo python se tiene definida la ruta del USB donde se tiene conectado la FPGA (esta dirección debe ser verificada por el usuario en caso de que tenga otra ruta). Para el correcto funcionamiento, primero se selecciona la opción del menú principal y salda el mensaje para presionar el botón, se corrobora que se están transmitiendo los datos con el LED correspondiente en la FPGA y al terminar de cargar los datos se graficarán en comparación el valor teórico.

Graficación de datos (grapresult.m)

En este archivo se encuentra las opciones para la generación de los graficos ya sea la graficación de los porcentajes de error como de la comparación del valor en la FPGA con el valor teórico.