DISEÑO DE SISTEMAS DIGITALES

Tópico Práctico. No. 1
"ispLEVER: ABEL – HDL, Diseño Lógico Combinatorio"

Dr. Juan Carlos Herrera Lozada.
jlozada@ipn.mx

Centro de Innovación y Desarrollo Tecnológico en Cómputo
Lab. de Diseño de Sistemas Digitales
Instituto Politécnico Nacional

Octubre, 2015

Campo 1: **Datos Personales.**

Campo 2: **Objetivos**.

- Instalación del Software de diseño.
- Realizar la descripción de circuitos de lógica combinatoria, a través de ABEL - HDL.
- Evaluar la mejor posibilidad para realizar una descripción de acuerdo a un diseño específico.
- Comprender la importancia de la simulación en un diseño formal. Familiarizarse con el flujo de diseño a través de un HDL vertido en PLDs.

Campo 3: **Desarrollo de la Práctica**.

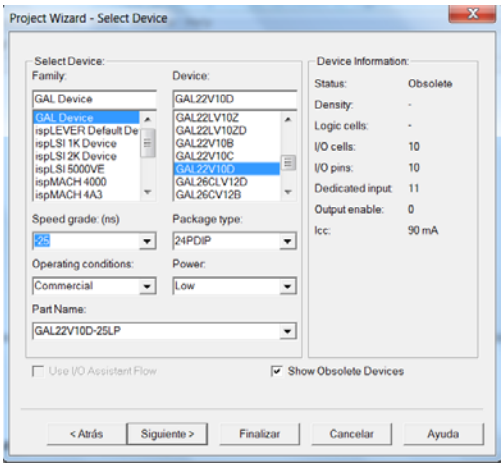
Nota: Para los diseños, anexar los respectivos códigos y simulaciones según el caso.

Preparación:
Descarga ispLEVER Classic e instálalo. La información pertinente está disponible en la página web, bajo el nombre isplever.txt. También requerirás la hoja de datos del GAL22V10.

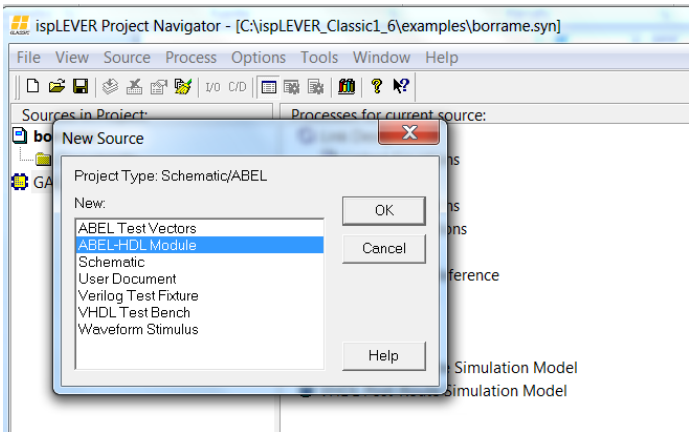
1. (2 puntos) Tutorial de Inicio.

1. Accede a *ispLEVER Project Navigator* y genera un nuevo proyecto a través de la opción *File* en el menú, seguido de *New Project*. Nómbralo como lo desees. Recuerda que en la opción *Design Entry Type* debes elegir Schematic/ABEL. Con respecto a las herramientas de síntesis y simulación (*Synthesis tools* y *Simulator tools*) debes mantener por omisión *Synplifi* y *Active-HDL*, respectivamente.

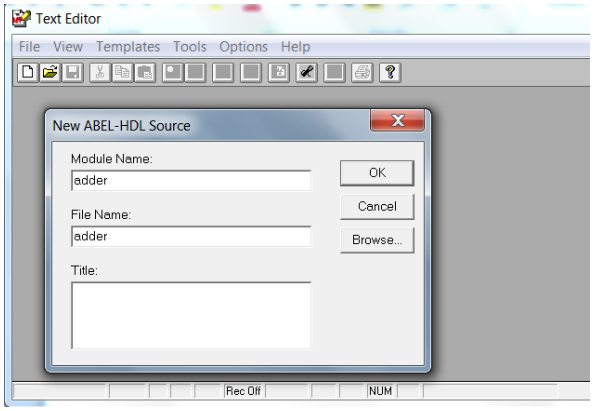
2. En la ventana siguiente, habilita la opción *Show obsolete devices* y selecciona en la columna *Family* la opción *Gal Device*. Elige el dispositivo GAL22V10D-25LP cumpliendo con los parámetros indicados en la siguiente imagen. Es importante el tipo de empaquetado (*Package type*), en nuestro caso 24PDIP. Finaliza la generación del proyecto.



3. En este punto incluirás tu archivo en ABEL, para tal propósito, en el menú principal del *Project Navigator*, selecciona *Source -> New.* El *new source* debe ser de tipo *ABEL-HDL Module*. Observa la siguiente figura.



4. Utiliza, tanto para el nombre del módulo como para el nombre del archivo, la palabra *adder*. Observa la figura siguiente.



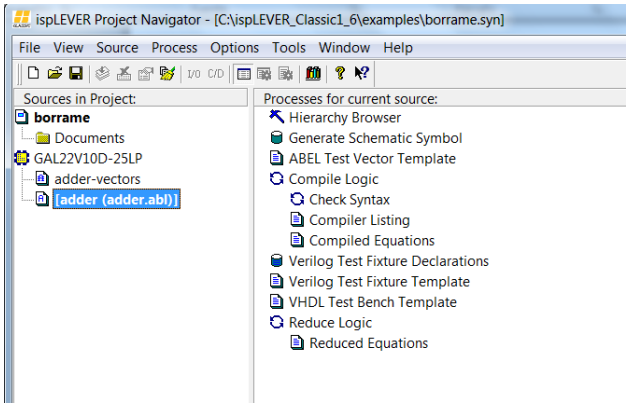
5. Captura el siguiente código que modela un *sumador completo* de un bit:

```
MODULE adder
"inputs
A,B,CI pin 1, 2, 3;
"outputs
CO, SUM pin 14, 15 istype 'com';
EQUATIONS
SUM = A$B$CI;
CO = A&B # A&CI # B&CI;
Test_vectors
([ A,  B, CI  ] -> [SUM, CO]);
[ 0,  0,  0  ] -> [0, 0];
[ 0,  1,  1  ] -> [0, 1];
[ 0,  1,  0  ] -> [1, 0];
[ 1,  1,  1  ] -> [1, 1];
END
```

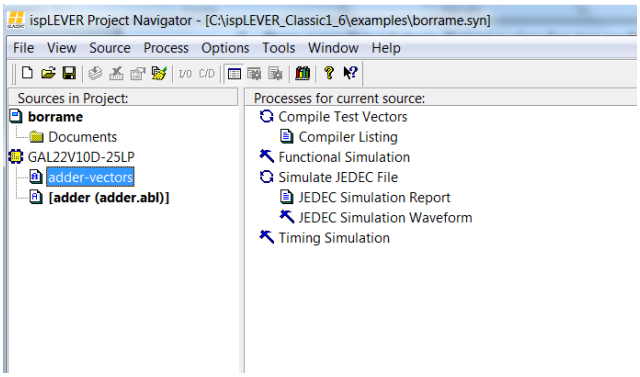
6. Guárdalo y observa que en tu proyecto, regresando al *Project Navigator*, se ha adicionado el archivo *adder.abl*, haciendo referencia al módulo que creaste al inicio. También verás el archivo `adder_vectors` adicionado de manera automática y que representa precisamente el vector para la simulación.

7. Para compilar (síntesis lógica) y simular tus archivos, sigue las siguientes indicaciones:

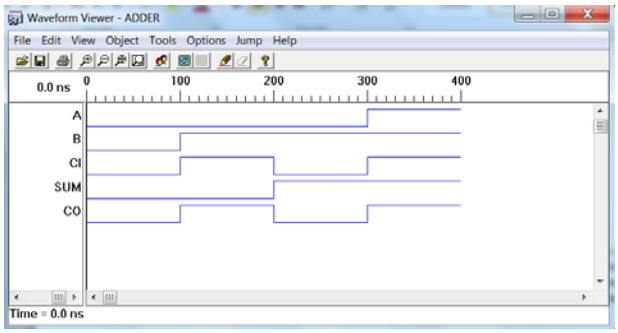
a. Da un clic sobre el archivo *adder.abl* para activar las opciones de compilación para este código. Ejecútalas con un doble clic, de una en una, comenzando por la primera (*Compile Logic*) hasta terminar con *Reduced Equations*.



b. Con un clic activa las opciones de compilación para `adder_vectors`. Ejecuta *Compile test vectors* y posteriormente *Simulate JEDEC file*.



c. Para acceder a las formas de onda de la simulación, ejecuta en esta misma ventana, la opción `Functional Simulation`, se abrirá una ventana individual etiquetada como *Simulator Control Panel*. En el menú de este mismo panel, accede a la opción *Simulate* y presiona `Run` para que automáticamente se abra el `Waveform Viewer` mostrando las formas de onda de la simulación.



d. Finalmente, selecciona con un clic en el *Navigator Project,* el dispositivo a programar (GAL22V10D) y ejecuta las opciones de síntesis. Recuerda observar el *Chip Report*, que te

indica cómo debes conectar el dispositivo, y el *JEDEC file* que te servirá para programar el PLD.

Discusión:

1.1 ¿Qué sucede con los archivos reportados y con los resultados gráficos de la simulación cuando en los vectores de prueba colocas como salida un valor erróneo, por ejemplo: 1+1+1 = 00

1.2 ¿Qué sucede en los resultados gráficos de la simulación cuando los estímulos anotados a las salidas son escritos como *.X.* en los vectores?

1.3 Elimina de tu código la designación numérica de pines y verifica el archivo *Chip Report*, después de repetir todo el proceso de desarrollo. ¿Qué pines asigno de manera automática la herramienta de síntesis?

1.4 Captura el siguiente código que modela un detector (de 3 bits) de números pares mayores que cero (Q) y de números primos (P). Verifica la simulación y muestra funcionando en un GAL el circuito.

```
MODULE pripar
"inputs
a2, a1, a0 pin 1,2;
"outputs
Q, P pin 14, 15 istype 'com';
EQUATIONS
Q = !a0 & (a1 # a2);
P = a0 # (!a2 & a1);
Test_vectors
([ a2, a1, a0  ] -> [Q, P]);
[0, 0, 0] -> [0, 0];
[0, 1, 1] -> [0, 1];
[0, 1, 0] -> [1, 1];
[1, 1, 0] -> [1, 0];
END
```

2. (2 puntos) Diseña un decodificador capturando una *tabla de verdad*, que muestre en un display a siete segmentos el resultado de multiplicar dos números de dos bits cada uno; captúrala en ABEL HDL e impleméntalo físicamente en un GAL.

2.1. Simula tu diseño e incluye las formas de onda.

2.2. ¿Qué porcentaje del GAL utilizaste en este diseño? ¿Dónde encuentras esta información?

El siguiente código modela un decodificador BCD de cátodo común. Auxíliate de él para la sintaxis de la tabla de verdad.

```
MODULE BCD7SEG
"Entradas
A, B, C, D pin 1, 2, 3, 4;
"Salidas
a,b,c,d,e,f,g pin 19, 18, 17, 16, 15, 14, 13 istype 'com, dc';
"Declaración de sets
Dato=[D,C,B,A];
"Declaración de una constante sólo para simulación"
x=.x.;

Equations
truth_table ([Dato] ->[ a,b,c,d,e,f,g]);
            [0] -> [1,1,1,1,1,1,0];
            [1] -> [0,1,1,0,0,0,0];
            [2] -> [1,1,0,1,1,0,1];
            [3] -> [1,1,1,1,0,0,1];
            [4] -> [0,1,1,0,0,1,1];
            [5] -> [1,0,1,1,0,1,1];
            [6] -> [1,0,1,1,1,1,1];
```

```
            [7] -> [1,1,1,0,0,0,0];
            [8] -> [1,1,1,1,1,1,1];
            [9] -> [1,1,1,0,0,1,1];

test_vectors
([Dato] ->[a,b,c,d,e,f,g]);
 [0] -> [x,x,x,x,x,x,x];
 [1] -> [x,x,x,x,x,x,x];
 [5] -> [x,x,x,x,x,x,x];
 [9] -> [x,x,x,x,x,x,x];
END
```

3. **(2 puntos)** Diseña un sumador de dos números de 2 bits cada uno, descrito de modo estructural (ecuaciones) y el resultado de la suma se escribe directamente en un display a siete segmentos de cátodo común. Deberás considerar que tanto el sumador como el decodificador para el display se implementaron en el mismo código. Verifica la simulación y programa fisicamente este diseño en el dispositivo GAL.

4. **(2 puntos)** Realiza la descripción estructural (con ecuaciones) de un sumador completo de dos números de 3 bits en ABEL HDL. El resultado debe mostrarse en un display a siete segmentos, con caracteres hexadecimales. Implementa tu diseño en dos dispositivos GAL22V10, uno para el sumador y otro para el decodificador a siete segmentos.

5. **(2 puntos)** Diseña una ALU en ABEL HDL que reciba dos números de 3 bits cada uno, se recomienda que utilices una descripción comportamental. La ALU debe realizar al menos 6 operaciones: dos aritméticas (suma y resta) y cuatro lógicas NOT (invirtiendo uno sólo de los dos números), AND, OR y XOR. Impleméntala en un GAL22V10.
Auxíliate del siguiente código que implementa una ALU con 4 operaciones básicas.

```
MODULE alu_ini

TITLE 'Esta es una aproximación inicial a una ALU'

a2, a1, a0, b2, b1, b0, s1, s0 pin 1, 2, 3, 4, 5, 6, 7, 8;  "Verificar pines
r3, r2, r1, r0 pin 19, 18, 17, 16 istype 'com, dc';
c3=0;
A=[c3, a2, a1, a0];
B=[c3, b2, b1, b0];
sel=[s1, s0];
R=[r3, r2, r1, r0];


equations
@carry 2;
when sel == 0 then R = A + B
else when sel == 1 then R = A - B
else when sel == 2 then R = A & B
else R = A $ B;

test_vectors
([a2, a1, a0, b2, b1, b0, s1, s0]->[R])
 [0, 0, 1, 0, 0, 1, 0, 0] -> [.X.]; "La suma de 1 + 1
 [0, 1, 0, 0, 0, 1, 0, 1] -> [.X.]; "La resta de 2 - 1
 [1, 0, 1, 1, 1, 1, 1, 0] -> [.X.]; "La and entre 5 y 7
 [0, 0, 0, 1, 1, 0, 1, 1] -> [.X.]; "La xor entre 0 y 6
END
```

Campo 4: **Conclusiones individuales.**