DISEÑO DE SISTEMAS DIGITALES

Tópico Práctico. No. 4

"Aplicaciones prácticas en ingeniería: Oscilador embebido, modulación por ancho de pulsos (PWM) y motor a pasos unipolar"

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

Mayo 2013

Campo 1: **Datos Personales.**

Campo 2: **Objetivos**.

- Diseño de circuitos de lógica combinatoria y secuencial.
- Solución de problemas prácticos en ingeniería.

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

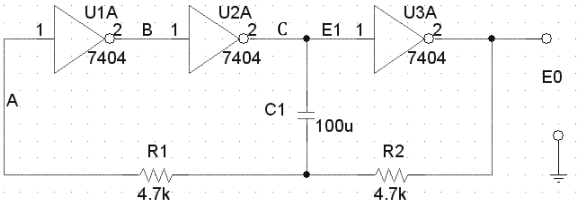
Nota: Para los diseños, anexar los respectivos códigos y simulaciones según el caso.

1. Oscilador embebido.

Un oscilador embebido es muy útil en el diseño de circuitos secuenciales, toda vez que se estaría prescindiendo del C. I. 555 para generar el pulso de reloj. Revisa el material correspondiente a este tópico práctico, disponible en línea en la página web del curso. De acuerdo al apunte, las resistencias a utilizar son fijas con un valor de 4.7KΩ y para el capacitor se calculó C= 100 μF, acorde a la relación para lograr una frecuencia de aproximadamente 1Hz.

$$f = \frac{0.559}{(4.7K\Omega)(100\mu F)} = 1.1Hz$$

El circuito a describir es el siguiente



El código en ABEL-HDL es muy sencillo, no obstante es muy importante respetar las conexiones a realizar. A través de la variable E0, colocando un LED, se apreciará la salida del oscilador.

```
module oscilador
A, E1 pin 8, 9;
C, E0 pin 16, 17 istype 'com,dc';
B node istype 'com,dc';
Equations
B = !A;
C = !B;
E0 = !E1;
End
```

Una solución más práctica infiere cambiar la resistencia R1 por un potenciómetro de 10KΩ, lo que permitirá variar la duración del pulso de reloj.

Actividad 1:

Describe el oscilador embebido utilizando ABEL-HDL e impleméntalo en un dispositivo GAL.

Actividad 2:

Accede al siguiente enlace: http://electronicacompleta.com/lecciones/fabricacion-de-dispositivos-con-compuertas-logicas/ y revisa los casos prácticos para los osciladores con compuertas lógicas. Selecciona cualquiera de los osciladores mostrados, que no sea el descrito en la Actividad 1, e impleméntalo en un GAL.

Discusión 1: ¿Es posible simular estos diseños en ISIS de Proteus? Revisa la teoría, prueba la simulación y justifica tu respuesta.

2. Diseño de una unidad PWM.

Actividad 3:

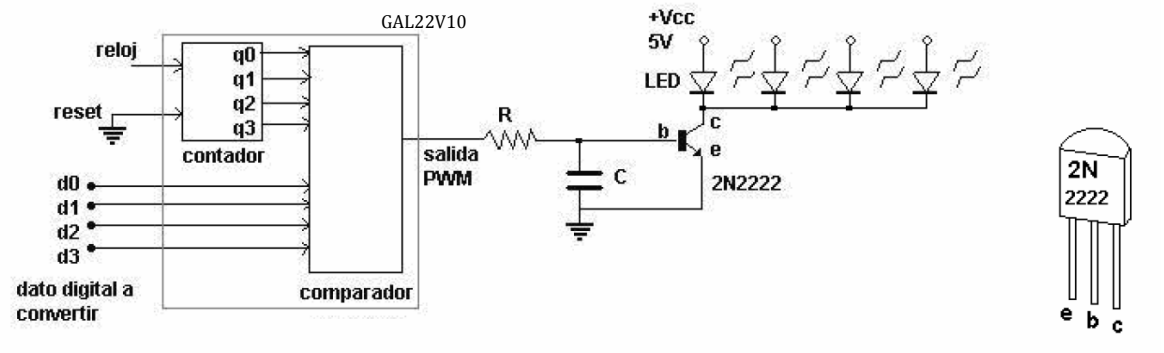
La unidad PWM, de acuerdo al material de la práctica 4, se modelaría con el siguiente código en ABEL-HDL. Es de suma importancia revisar los valores del filtro R-C que se debe conectar para lograr la modulación por ancho de pulsos, dado que éstos fueron calculados para una frecuencia particular. La idea básica de este módulo es utilizarlo como una aproximación a un DAC (*Digital – Analogic Converter*): de acuerdo a un dato digital de entrada, de 4 bits, etiquetado como *entdn3..entd0*, se obtendrá un dato analógico de salida, etiquetado en el código como *salida_pwm*, equivalente, entre 0 y 5 Volts; por ejemplo, para una entrada digital 0000 se obtendrá, idealmente, una salida analógica de 0V y para una entrada digital de 1111 se obtendrá el voltaje analógico máximo de 5V.

```
Module pwm
Title 'pwm en GAL'
"Pines de entrada
reloj, rst, entd3..entd0 pin 1..6;

"Pines de salida
cont3..cont0 pin 23..20 istype 'reg';
salida_pwm pin 14 istype 'com';

"sets
ent = [entd3..entd0];
cuenta =[cont3..cont0];

"constantes
c,x=.c.,.x.;

Equations
@carry 2;
cuenta.clk=reloj;
cuenta.ar=rst;
"Comparacion de entrada digital y contador
when cuenta<ent then salida_pwm=1;
else salida_pwm = 0;
"Incrementa contador
cuenta:= cuenta + 1;

Test_vectors
([reloj,rst,ent]->[salida_pwm,cuenta]);
[c,1,3]->[x,x];
@repeat 15 {[c,0,1]->[x,x];};
@repeat 15 {[c,0,4]->[x,x];};
@repeat 15 {[c,0,9]->[x,x];};
@repeat 15 {[c,0,13]->[x,x];};

End
```

Para ver funcionando el módulo PWM, utilizaremos unos LEDs, con la finalidad de regular su intensidad luminosa. Es posible que en vez de los LEDs, conectes un motor a CD pequeño o un speaker sencillo, en efecto sería muy similar al comportamiento de los LEDs. El circuito a alambrar se muestra en el siguiente diagrama.

Discusión 2:

El código en ABEL, anteriormente listado, incluye los vectores de prueba para el simulador funcional de ISP Classic, con lo que comprobaríamos la salida con respecto a las 4 entradas binarias, pero, ¿qué sucederá con la simulación en Proteus? Resulta factible en ISIS de Proteus, incluir el filtro R-C a la salida del GAL en tu diseño, inclusive, se puede colocar hasta un osciloscopio pata ver el ancho del pulso a la salida del filtro. Realiza la simulación.



Actividad 4:

En el numeral 1 de esta misma práctica, diseñamos un oscilador embebido. A continuación se lista el código para implementar la unidad PWM que trabaja con un oscilador embebido en el mismo GAL22V10. Físicamente, la salida del oscilador E1, debe conectarse al pin 1 del GAL, para "engañar" a éste, haciéndole creer que es una señal de reloj independiente. La complejidad en este diseño radica en el hecho de que se debe calcular los valores R y C del oscilador embebido para que sean congruentes con los valores del filtro R-C de la unidad PWM.

```
MODULE pwm_oscila
Title 'pwm en GAL con oscilador embebido'

"Pines de entrada
reloj, rst, entd3..entd0 pin 1..6;
A,E1 pin 7,8;   "del oscilador

"Pines de salida
cont3..cont0 pin 23..20 istype 'reg';
salida_pwm pin 14 istype 'com';
C, E0 pin 15,16 istype 'com';   "del oscilador
B node istype 'com';    "del oscilador

"sets
ent = [entd3..entd0];
cuenta =[cont3..cont0];

"constantes
c,x=.c.,.x.;

"oscilador aprox a 1 seg
Equations
B=!A;
C=!B;
E0=!E1;

Equations
@carry 2;
cuenta.clk=reloj;
cuenta.ar=rst;
"Comparacion de entrada digital y contador
when cuenta < ent then salida_pwm = 1;
else salida_pwm = 0;
"Incrementa contador
cuenta:= cuenta + 1;

Test_vectors
([reloj,rst,ent]->[salida_pwm,cuenta]);
```

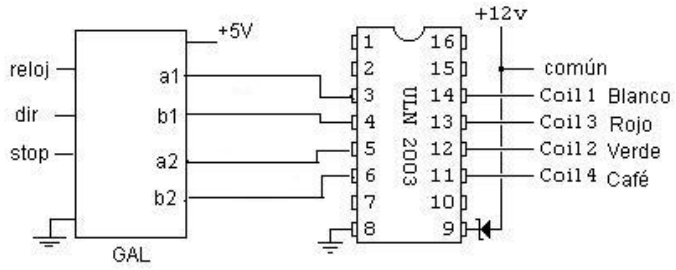
```
[c,1,1]->[x,x];
@repeat 16 {[c,0,1]->[x,x];};
@repeat 16 {[c,0,7]->[x,x];};
@repeat 16 {[c,0,13]->[x,x];};
End
```

La realización de un módulo para controlar el sentido de giro de un motor a pasos se logra a través de una sencilla máquina de estados. El siguiente código nos permite diseñar dicho controlador digital; es recomendable revisar el artículo disponible en línea, en la página web del curso, para reforzar la teoría del funcionamiento de los motores a pasos unipolares.
Nótese en el código, que la variable *dir* permitirá el cambio en el sentido del giro, invirtiendo la secuencia de estados de la máquina, mientras que la variable *stop* mantendrá el estado actual deteniendo el motor.
Recuerda que la frecuencia de reloj será quien dé la pauta para la velocidad de giro del motor, siendo que con una frecuencia muy lenta, el avance o retroceso del motor, según sea el caso, se verá afectado. Lo más recomendable es utilizar la frecuencia de 1Hz o un poco más rápida, por lo que al C. I. 555 se le puede cambiar una resistencia fija por un potenciómetro para que se pueda experimentar con diferentes frecuencias.

```
MODULE motor_pasos
"Secuencia de Medio Paso
reloj,dir,stop, reset pin 1,2,3,4;
"salidas registradas
a1, b1, a2, b2 pin 14,15,16,17 istype 'reg, dc';

"declaración de set
sreg=[a1,b1,a2,b2];
E0=[0,0,0,0];
E1=[1,0,0,0];
E2=[1,1,0,0];
E3=[0,1,0,0];
E4=[0,1,1,0];
E5=[0,0,1,0];
E6=[0,0,1,1];
E7=[0,0,0,1];
E8=[1,0,0,1];

Equations
sreg.clk=reloj;
sreg.ar=reset;

state_diagram sreg

state E0:
IF dir THEN E1 ELSE E8;

state E1:
IF !stop THEN E1
ELSE IF dir THEN E2 ELSE E0;

state E2:
IF !stop THEN E2
ELSE IF dir THEN E3 ELSE E1;

state E3:
IF !stop THEN E3
ELSE IF dir THEN E4 ELSE E2;

state E4:
IF !stop THEN E4
ELSE IF dir THEN E5 ELSE E3;

state E5:
IF !stop THEN E5
ELSE IF dir THEN E6 ELSE E4;

state E6:
IF !stop THEN E6
ELSE IF dir THEN E7 ELSE E5;
```

```
state E7:
IF !stop THEN E7
ELSE IF dir THEN E8 ELSE E6;

state E8:
IF !stop THEN E8
ELSE IF dir THEN E1 ELSE E7;

END
```

Por lo general, los motores a pasos requieren un voltaje nominal de 12V y una corriente de 150 mA por bobina, claro, existen muchas variaciones al respecto. Se debe considerar un voltaje de 5 Volts para el GAL y, como ya se mencionó, uno de 12 Volts para el motor, por lo que se trata de dos fuentes de voltaje diferentes. Si fuera el caso utilizar C. I. reguladores (LM7805 Y LM7812), es necesario adecuar el voltaje de alimentación general para soportar ambos dispositivos, no obstante, una opción más simple es recurrir al uso de una fuente de alimentación de computadora, ésta ya dispone de ambos voltajes, precisamente 5 y 12 Volts.
El GAL no entrega la suficiente corriente para excitar las bobinas por lo que es necesario agregar una etapa de potencia a su salida, para amplificar correctamente la corriente de alimentación a las 4 bobinas del motor a pasos unipolar. Esta etapa de potencia (driver) se puede implementar con un integrado monolítico ULN2003 que entrega hasta 500mA y presenta la característica de ser un driver con trabajo invertido, es decir, complementará los datos entrantes. En realidad no existe problema alguno, porque sólo basta con conectar el común a los 12 Volts nominales y conservar la misma tabla de secuencias establecida, ya que un "1" al entrar al driver se convertirá en un "0" y viceversa.
El circuito a armar es el siguiente. El diodo zener es de 12 Volts.



Actividad 5:

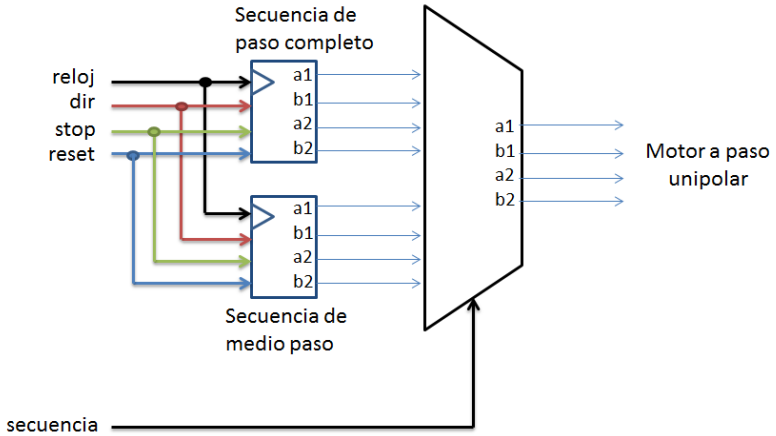
Adecúa el código integrando los vectores para simulación en el simulador funcional de ISP Classic. Realiza la simulación en ISIS de Proteus utilizando un motor a pasos unipolar de la biblioteca de componentes; no es necesario el driver, por lo menos en simulación.
Posteriormente implementa el diseño en un GAL22V10 y muéstralo funcionando para cumplir este numeral de la práctica.

Actividad 6:

Dado que en el apunte correspondiente se indica que existen básicamente dos secuencias para trabajar con los motores a pasos (paso completo y medio paso). El diseño a resolver es un sistema digital que permita seleccionar una de las dos secuencias posibles, además de mantener el reset, el stop y el sentido del giro (*dir*), para tal efecto se requiere una variable que permita conmutar o multiplexar entre ambas; en el siguiente diagrama, la variable *secuencia* tiene esta función.

Ahora bien, poder elegir entre las secuencias no es el único problema a resolver, dado que independientemente de que ya se haya seleccionado una secuencia, es posible advertir que el reloj sigue activo en ambas máquinas de estado, por lo que la operación correcta es que sólo la secuencia elegida debe mantener activa la señal de reloj y la otra secuencia debe inhabilitar la entrada de dicha señal de reloj. El primer ejercicio de la práctica 3, puede auxiliarnos a lograr esto, considerando la directiva punto *.ce* (*clock enable*), que trabajaría de manera inversa entre ambas máquinas de estado, es decir, cuando *.ce* esté habilitado en una máquina de estados, en la otra está deshabilitado. Para

controlar *.ce* bastará con la misma variable que se nombró *secuencia* y que controla la selección de la máquina de estados que ha de mover al motor.
Implementa este diseño y muéstralo funcionando; puedes proponer una implementación modular y presentar tantos dispositivos como te sean necesarios, sin olvidar la etapa de potencia para que funcione adecuadamente el motor. Te recomiendo colocar LEDs a la salida del GAL, antes de conectar la etapa de potencia, sólo para verificar que tus máquinas de estados trabajen correctamente.



Campo 4: **Conclusiones individuales.**