**2017-12-14. ERROR EN IMPLEMENTACIÓN DEL PLL. HW. Se muestra la secuencia de correos con Gerardo Medrano**

El 14/12/2017 a las 0:31, Emilio Bueno escribió:

Gerardo, le he vuelto a dar una vuelta y TIENES RAZÓN tenemos un error en el SOGI. Lo que está programado es:

error\_qv      = in1       - (  \*qv);   
pre\_v         = error\_qv  \* (wn\*ts);  
pre\_qv        = (\*v)      \* (wn\*ts);  
(\*v)         += pre\_v;  
(\*qv)        += pre\_qv;

Este es un error que se arrastra desde que se creo el PLL HW, porque se bailó una línea del PLL. Este tema se ha camuflado siempre porque el TS es muy pequeño, y por la realimentación que se hace en el SOGI-QSG. Pero es un error.  Es un error, porque para el calculo de pre\_qv estamos usando la v(k-1) y deberíamos emplear v(k). La opción correcta es:  
  
error\_qalfa = input – qalfa;   
STEP\_W = STEP \* W;     
pre\_alfa = error\_qalfa\*STEP\_W;   
alfa += pre\_alfa;    
pre\_qalfa = STEP\_W \* alfa;    
qalfa += pre\_qalfa;   
  
Esto es justo lo que tú implementas. Hablamos cuando tengas un ratillo. Sorry por mi email anterior, y si quieres revisamos función por función el PLL.  Saludos,  
  
       Emilio Bueno  
       
  
  
El 13/12/2017 a las 15:50, Emilio Bueno escribió:

Buenas Gerardo, te voy contestando a continuación.

El 07/12/2017 a las 9:34, Gerardo Medrano Arana escribió:

Hola Emilio!

En este correo te cuento los resultados del análisis preliminar del PLL, y lanzo también algunas preguntas.

**En primer lugar:**

* ¿Por qué son formalmente tan distintos los ficheros C del PLL que funciona en el control y del que va en la FPGA?. Entiendo que la implementación en la FPGA requerirá de algunos ajustes, pero parece un código rescrito desde cero, y de cara a tener una sistemática de migración de algoritmos a FPGA, deberíamos ser capaces de modificar las funciones del control para que con ajustes mínimos el nuevo C nos valga para la herramienta de traducción a VHDL.

Gerardo, en el código en C, como empleamos un período de muestreo de 100us o 200us, programamos los SOGI-QSGs como funciones de transferencia, discretizando por el método Tustin. En el PLL, dado que el periodo de muestreo es 12.5us, discretizamos los PLLs por emulación de cada integrador. Efectivamente esta segunda opción es menos exacta para periodos de 100us o 200us, pero para 12.5us, la precisión es muy buena, y tiene la ventaja que hacer la adaptación en frecuencia es muy sencilla. Cualquiera de los dos métodos es válido, la exactitud depende del periodo de muestreo. Si queremos hacer adaptación en frecuencia a partir de la discretización de la función de transferencia completa deberíamos implementar ecuaciones en diferencias, forma directa I o forma directa II, igual de sencillo, pero hay ya que gastar algún recurso más

* Para analizar el código del PLL, empecé función por función, y la primera que escogí fue el SOGI, que dio resultado negativo (no se comporta como debiera un SOGI) y a continuación detallo. Aparte, una inspección visual del código nos hizo pensar que en algunas funciones el paso de argumentos estaba mal hecho, pero según me comenta Javier, parece que esto no es así. En cualquier caso, detuve mi análisis del resto del código por estos dos motivos y porque me gustaría, como explico en el punto anterior, que el código C que va a la FPGA y el código C de control se parecieran no sólo en fondo si no también en forma.

**Análisis de la implementación del SOGI:**

* He desarrollado un entorno de validación que permite ejecutar código C desde Simulink. He ejecutado la función del SOGI del PLL y he dibujado la salida ‘v’ para:

* 1. El código original (single precission).
  2. Una modificación del código hecha por mí (single precission).
  3. El SOGI implementado tal y como está en el modelo de Simulink (double precission).

Ts???

* Un escalón unitario (contiene todas las frecuencias) debería de ser capaz de excitar al SOGI para que se quede resonando a una amplitud constante. Se aprecia que la implementación original (1) crece en amplitud indefinidamente y por tanto tiene que ser incorrecta. Analicé la implementación de Simulink (3) y en base a ella escribí el código modificado (2), realizando las mismas operaciones en el mismo orden (A, B, C, D, E). Con esto parece que la respuesta es la adecuada y se observan errores del orden 1e-6 probablemente debidos al uso de datos de diferente resolución.

OK, pero qué Ts estás empleando?? El truco de esto está en plantearse las ecuaciones diferenciales planteadas y hacer el rlocus, y ver dónde caen las raíces del sistema.

|  |  |
| --- | --- |
| **1.Código Original** | **2.Código Modificado (GMA)** |
| void sogi(float in1, const float ts, const float wn, float \*v, float \*qv)  {    float error\_qv, pre\_v, pre\_qv;    error\_qv      = in1       - (  \*qv);    pre\_v         = error\_qv  \* (wn\*ts);    pre\_qv        = (\*v)      \* (wn\*ts);    (\*v)         += pre\_v;    (\*qv)        += pre\_qv;    } | void sogi(float in1, const float ts, const float wn, float \*v, float \*qv)  {      float A, B, C, D, E; **// ver código Simulink**        A = in1 - (\*qv);      B = A\*ts\*wn;      C = B   + (\*v);      D = C\*ts\*wn;      E = D   + (\*qv);        \*v = C;      \*qv= E;  } |
| **Respuesta a escalón unitario** | **Respuesta a escalón unitario** |
|  |  |
| **3.Código Simulink** | |
|  | |
|  | |

Por cierto, mira en el paper adjunto, creo que te falta en esta figura un retardo computacional. De alguna forma, en tu implementación tienes que estar incluyendo un retardo computacional, porque si no tendrías un lazo algrebraico. Por último, yo te recomiendo que también pruebes la respuesta con SOGI-QSG. Al realimentar se compensa bastante este efecto de derivas que se producen debido a trabajar en el dominio discreto.

Bueno, pues estos son los resultados del análisis que en resumen vienen a ser:

* El sogi no parece validar, pero tenemos una propuesta (2) que parece que sí.
* Necesitaríamos que el código que va a la FPGA sea parecido formalmente al de la tarjeta de control para poder desarrollar una metodología de migración de algoritmos.
* Tenemos algunos asuntos pendientes con el paso de valores por referencia que Javi y yo revisaremos.

Como conclusión, con objerto de hacer adaptable en frecuencia el PLL, si queréis planteamos un nuevo fichero .cpp donde cambiamos la forma de implementar los SOGI-QSGs, los implementemos como ecuaciones en diferencias, pero con la posibilidad de hacer adaptación en frecuencia.

Quedo a la espera de tus comentarios que seguro que pueden arrojar luz sobre estos asuntos,

Un saludo!

Gerardo