IP core DALII

Conference Paper · March 2016

CITATIONS
0

READS
110

4 authors, including:

Bruno Valinoti
Instituto Nacional de Tecnologia Industrial
20 PUBLICATIONS **5** CITATIONS
SEE PROFILE

Rodrigo Alejandro Melo
Instituto Nacional de Tecnologia Industrial
34 PUBLICATIONS **15** CITATIONS
SEE PROFILE

Francisco Salomón
Instituto Nacional de Tecnologia Industrial
9 PUBLICATIONS **48** CITATIONS
SEE PROFILE

IP core DALI

Bruno Valinoti, Rodrigo A. Melo, Leandro Tozzi, Francisco Salomón
Instituto Nacional de Tecnología Industrial
Centro de Micro y Nanoelectrónica
Email: {valinoti,rmelo,ltozzi,fsalomon}@inti.gob.ar

***Resumen*—DALI es una interfaz de control de luminarias, con una estructura *master - slave* destinada al ahorro de energía, simplicidad y compatibilidad entre productos de diferentes fabricantes. En este trabajo se presenta el diseño de un bloque IP para un *slave* DALI para el control del *led driver* que está desarrollando el grupo de diseño analógico de nuestra institución. Para asegurar que la implementación esté acorde con el estándar se realizaron los casos de prueba de la parte digital propuestos por la especificación. Para validar el diseño se utilizó un kit que contiene una FPGA Spartan 6, comprobándose el correcto funcionamiento.**

I. Introducción

El estándar *Digital Addressable Lighting Interface* (DALI) fue descripto en el Anexo E de la IEC 60929 y está estandarizado en la IEC 62386. Es sucesor de los sistemas de control 0-10 V y una alternativa a *Digital Signal Interface* (DSI), en el cuál se basa, siendo la principal diferencia el agregado de direccionamiento individual de lámparas. En contraposición a *Digital MultipleX* (DMX), que fue concebido principalmente para espectáculos, da solución y control a sistemas de iluminación de edificios. Es un protocolo de datos y un mecanismo de transporte desarrollado por varias empresas de equipamiento de iluminación, pensado en asegurar interoperabilidad entre diferentes fabricantes, funcionalidad, simplicidad, ahorro de energía y bajos costos.

El área de diseño analógico de nuestro grupo de trabajo, se encuentra desarrollando un *driver* de potencia integrado para luminaria *led* (*Led Driver*). Para darle valor agregado al circuito integrado en desarrollo, se optó por incorporarle el protocolo DALI dentro del mismo ASIC, cuando lo común según la búsqueda llevada a cabo, es utilizar un procesador externo que lo resuelva en *firmware* [1].

En este trabajo se presenta el prototipo desarrollado sobre FPGA de un *Control Gear*, *slave* DALI, con la finalidad de validarlo antes de enviarlo a fabricar en silicio. Se mencionan las características del *core* desarrollado, pruebas realizadas y resultados de síntesis sobre un *kit* que incorpora una Spartan 6 de Xilinx, emulando la memoria no volátil (NVM) necesaria mediante la utilización de BRAMs.

La estructura del trabajo es la siguiente: La sección II introduce brevemente el estándar DALI y sus características. La sección III presenta la implementación del *core* desarrollado mientras que la sección IV indica las pruebas y la validación realizada en *hardware*. Finalmente, las secciones V y VI contienen resultados y conclusiones.

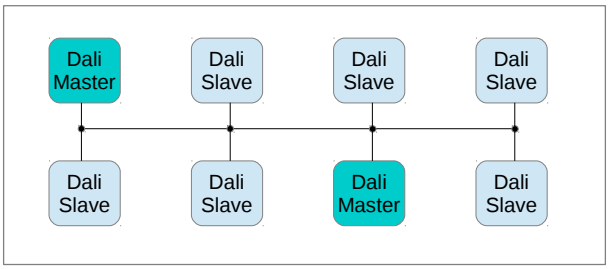


Figura 1. Topología del bus

II. Estándar DALI

El estándar IEC 62386 está dividido en tres grandes partes que definen, requerimientos generales, de los *Control Gears* y de los *Control Devices*:

- Parte 100 - Requerimientos generales
- Parte 200 - Requerimientos particulares de *Control Gears*
- Parte 300 - Requerimientos particulares de *Control Devices*

A su vez, cada parte principal está dividida en sub-partes, definidas en documentos individuales, y cada una de estás posee cláusulas o secciones. Estas indican cuestiones particulares de cada una de las partes y muchas veces sólo hacen referencia a las diferencias con las cláusulas de otra parte precedente.

Para la implementación de un *slave* DALI *led*, las partes necesarias, y en las que basamos nuestro diseño, son la 101: *System* [2], 102: *Control Gears* [3] y 207: *Led Modules* [4], en su versión 2009.

La topología de bus está basada en un canal de comunicación de dos vías serie, donde hay al menos un controlador *master* y generalmente múltiples *slaves* (Fig.1). El estándar provee un mecanismo de prevención, detección y recuperación de colisiones para los controladores tipo *master*, los *slaves* no corren peligro de generar colisiones debido a que las consultas deben ser realizadas a direcciones particulares. Cuenta con un máximo de 64 direcciones, que pueden dividirse en hasta 16 grupos. Cada grupo comparte hasta 16 escenas configurables, donde cada escena está definida como un nivel de intensidad de salida preestablecido. Los *slaves* pueden ser direccionados individualmente, por grupo, o en su totalidad en el caso de la transmisión de mensajes *broadcast* o especiales de direccionamiento.

Los datos están codificados según Manchester diferencial *bi-phase*, donde 01 corresponde a 1 y 10 corresponde a 0.

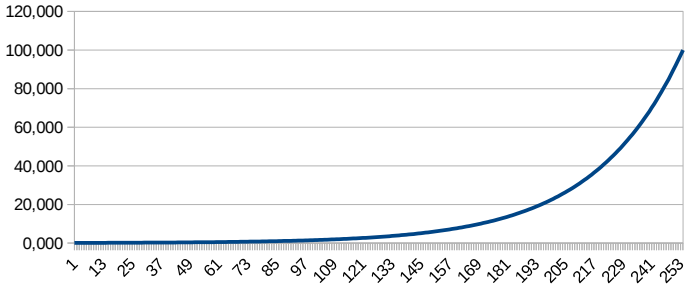


Figura 2. Curva de respuesta al dimming

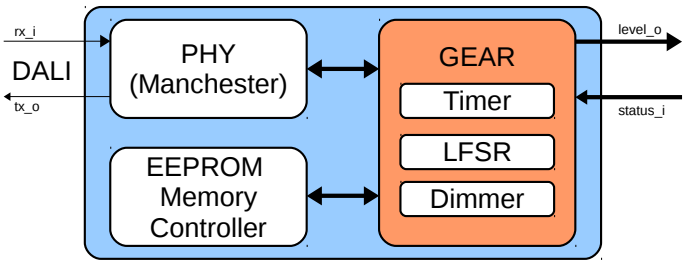
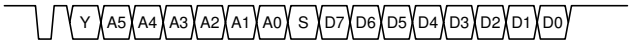


Figura 3. Diagrama en bloques



(a) Trama Forward

(b) Trama Backward

Figura 4. Tramas Backward y Forward

Dado que se utilizan dos símbolos por cada *bit* de información y que el *bit rate* está especificado en 1200 bps $\pm$ 10 %, la frecuencia a la cual funciona el canal es de 2400 Hz.

Para un módulo LED está especificado que la curva de respuesta de intensidad a la salida del módulo puede ser lineal o logarítmica, pudiendo elegirse cual de las dos utilizar. Por defecto se utiliza la representación logarítmica, que tiene fundamento en la respuesta del ojo humano, dónde una potencia de salida establecida en nivel 1 se corresponde con un nivel de un 0.1 % y un nivel 254 a un 100 %. La respuesta a su vez debe responder a la curva descripta en la Fig. 2 de acuerdo a:

$$X(n) = 10^{\frac{n-1}{\frac{253}{3}-1}}$$

$$|\frac{X(n) - X(n+1)}{X(n)}| = cte = 2{,}8\,\%$$

Si durante la operación del dispositivo se cambia el valor de intensidad de salida, el pasaje de uno a otro (*fading*) puede realizarse fijando un tiempo o una velocidad de cambio. Si se configura un valor mínimo por arriba del actual o uno máximo por debajo, la salida debe actualizarse inmediatamente al nuevo valor.

Respecto al direccionamiento, cada dispositivo en el *bus* debe poseer una dirección corta de 6 *bits* para identificarse unívocamente. En el primer arranque o en caso de haber conflictos por repetición de direcciones, entran en juego la mayoría de los comandos especiales y una dirección larga de 24 *bits* aleatorios.

III. Implementación

Se implementó un *core slave* DALI del tipo LED utilizando VHDL 93 estándar. El mismo consiste en una máquina de estados principal que coordina la comprobación y ejecución del protocolo, y un decodificador de comandos. Como componentes secundarios se implementaron un codificador/decodificador Manchester, un controlador de memoria EEPROM, un *Linear Feedback Shift Register* (LFSR) para generar direcciones pseaudo-aleatorias de 24 *bits*, una serie de *Timers* y un bloque que gestiona los niveles de salida (que denominamos *Dimmer*). El diagrama en bloques del sistema puede apreciarse en la Fig. 3.

Dado que se implementó un prototipo sobre una FPGA de Xilinx, para emular completamente el funcionamiento de la memoria EEPROM, se utilizó un bloque de de memoria configurable BRAM. De esta manera se deja la posibilidad de luego poder realizar la validación del controlador de memoria.

Cuadro I
Codificación de la dirección

Dirección	Descripción
Individual	0AAAAAAS
Grupal	100AAAAS
Broadcast	1111111S

En el bus DALI existen dos tipos de mensajes. Según donde se originen puede tratarse de tipo *forward* (Fig. 4(a)) cuando se generan en el *master* con destino *slave*, tienen un largo de 16 *bits*, donde los 8 primeros corresponden a la dirección y los últimos 8 al dato o comando. La dirección a su vez posee 2 *bits* especiales, ***Y*** que indican dirección individual (0) o grupal (1) (Cuadro I), y ***S*** si debe interpretarse como intensidad de salida a aplicarse (0) o si se trata de un comando (1). Cuando tienen sentido opuesto los comandos se denominan *backward* (Fig. 4(b)), tienen una longitud de 8 *bits* y se dan como respuesta a ciertos comandos y al tipo de respuesta (si, no o un dato).

En estado *IDLE* el *bus* permanece en nivel alto, hasta recibir un *bit* de inicio, a continuación *bits* de información, donde se transmite primero el más significativo, y finalmente dos *bits* de parada (Fig. 4). El protocolo no incluye chequeo ni corrección de errores, por lo que no fue necesario incluir un bloque dedicado a tal tarea.

Al tratarse de un *core* para un dispositivo *slave*, no habrá inconvenientes al recibir tramas del tipo *backward* debido a que serán descartadas, evitándose de esta manera el problema que pudiera causar el hecho de que el dispositivo esté escuchándose a sí mismo.

La parte 102 de la especificación contempla 171 comandos, implementados en su totalidad. La parte 207 adiciona 25 comandos, de los cuales no se implementaron 5 que hacen

referencia al control del sistema de protección de corriente y de estado de alimentación, debido a que no se contaba con el *hardware* analógico necesario.

La máquina de estados principal que se encarga del control y ejecución del protocolo se ejecuta coordinadamente con la de decodificación de los comandos y está organizada según el tipo de comando, que la norma organiza en cuatro grandes grupos:

- Control de nivel de salida: Directos, cuando el *bit* ***S*** = 0, o indirectos en caso contrario. Controlan la intensidad de salida del *slave*, que puede variarse linealmente, siguiendo una curva logarítmica o por saltos discontinuos de valores. Las variaciones pueden darse con un tiempo fijo y velocidad de cambio seleccionable, o puede seleccionarse el tiempo de cambio.
- Configuración: Permiten modificar los parámetros de funcionamiento, en su mayoría almacenados en NVM. Tienen la particularidad de enviarse por duplicado dentro de una ventana de tiempo de 100 ms sin que haya sido enviado otro comando intermedio al *slave* en cuestión.
- Consulta: Permiten conocer información del dispositivo, configuraciones y estado de operación. Son los únicos que generan respuestas.
- Especiales: Están dirigidos a todos los *slaves* del *bus* y su función principal está relacionada al direccionamiento del dispositivo en el mismo. Utilizan como especificación de comando las direcciones desde 101AAAA1 hasta 110AAAA1 (0xA1 - 0xDF), donde siempre se cumple ***Y***=1 y ***S***=1.

Los dispositivos de tipo *led* agregan los denominados Comandos de Aplicación Extendida, de Configuración y Consulta. Tienen la particularidad de que deben ser precedidos por un comando especial que los habilita.

La norma introduce el requerimiento de las llamadas *variables*, especificando tamaños, valores por defecto, si son constantes o variables, y si se guardan en RAM o NVM. Como regla general en NVM se almacena información de configuraciones (dirección individual, grupos, escenas, valores límites de niveles y otros) y constantes tales como versión de especificación y tipo de dispositivo. En RAM en cambio se almacena la información de estado y funcionamiento.

El polinomio generador utilizado para el LFSR es:

$$x^{24} + x^{23} + x^{22} + x^{17}$$

Esto permite generar un máximo de valores posibles:

$$2^{24} - 1 = 16{,}777{,}215$$

Esta configuración permitió superar satisfactoriamente los test de aleatoriedad propuestos por la especificación. Adicionalmente se corrieron seis pruebas de aleatoriedad NIST SP800-22 [5] (*Frequency Mono-bit Test, Frequency Test within a block, Run Test, Test for the Longest Run of ones in a block*) [6].

Entre las diferentes funciones que debe realizar el bloque de gestión de nivel de salida se halla el encendido y apagado, regulación de intensidad, selección de tipo de curva de variación (lineal-logarítmica), configuración y eliminación de escenas, agregado y remoción a grupos, entre otros, controlando que el rango de operación se encuentre siempre dentro de los límites configurados.

Para generar las diferentes opciones de *fading* posibles se utilizó en el Dimmer un un contador de 16 *bits* conjuntamente con *timer* de 5ms dónde los 8 *bits* mas significativos del contador reflejan el nivel de salida, dejando los 8 menores para incrementar la resolución de la cuenta, necesaria en aquellas transiciones menores a la unidad debido a tasas de variación muy bajas o a tiempos de dimerizado muy extensos entre niveles de salida muy próximos y aumentando la resolución.

El bloque Timer controla y gestiona la temporización de todos los tiempos en juego que están en la especificación, está basado en una serie de contadores de cuenta fija con entradas para comienzo y limpieza de cuenta y con salidas para indicaciones de fin de cuenta, indicando los diferentes *timeouts* vencidos.

Para la codificación/decodificación Manchester se adoptó como estrategia utilizar una frecuencia de reloj 16 veces mayor que la especificada de manera tal que puedan recuperarse y enviarse los mensajes al canal sin pérdida de información, se eligió esta frecuencia debido a que el *driver* analógico cuenta con un reloj de 19200 Hz.

Una gran parte de los parámetros pueden ser guardados dinámicamente dentro de NVM, de manera que los *slaves* al encenderse realicen acciones por defecto sin la necesidad permanente de un dispositivo DALI *master* que los esté gobernando. Estos parámetros deben ser guardados en sectores y direcciones de memoria establecidos por la especificación y pueden ser leídos en su totalidad, de la misma manera que el resto de las posiciones de memoria.

IV. Testeo y validación

La cláusula 12, *Test Procedures* de cada una de las partes indica los test que deben realizarse para asegurarse el correcto funcionamiento y desarrollo del IP *core*. Las secuencias de tests consisten en su mayoría en una fase de activación de un comando y una fase de validación. En la fase de activación se envía al DUT, en este caso al IP *core* el comando bajo prueba y en la etapa de validación se observa el estado interno mediante los comandos de consulta.

Los tests están diseñados siguiendo una determinada secuencia de manera tal que pueda detectarse precisamente en que comando o parte del diseño se encuentran los eventuales errores. Cada parámetro de entrada y respuesta estan debidamente detallados, de la misma manera los mensajes y códigos de error para cada uno de los tests.

De las secuencias de testeo estipuladas en la especificación se realizaron las de:

- Configuration commands
- Arc power control commands
- Physical address allocation
- Random address allocation
- Queries and reserved commands

Cuadro II
RESULTADOS DE SÍNTESIS

	FFs	LUTs	BRAMs
Completo	690	2483	1
Sin RESET	686	2351	1

- Application extended query commands for device type 6

No se tuvieron en cuenta los tests de *Physical operational parameters* debido a que hacen referencia a cuestiones que tienen que ver con la interfaz analógica.

Para comprobar el funcionamiento del *dimmer* se graficó la salida del *gear* con el eje de ordenadas en escala logarítmica y en escala lineal de manera que pueda observarse con mayor facilidad cuanto se aleja la respuesta de una exponencial pura. Pudieron verificarse correctamente todos los casos de prueba y además se corroboró la continuidad de las curvas cuando se cambiaban las pendientes de tasa de cambio y tiempos de *fading*.

Luego de la comprobación del correcto funcionamiento del *core* con los bancos de prueba, se hicieron verificaciones de funcionamiento sobre *hardware*. Se utilizó FPGA Spartan-6 XC6SLX9-2CSG324C de Xilinx y el ISE WebPack 14.7.

La prueba consistió en enviar comandos mediante un *script* escrito en *Python* e ir observando la respuesta del diseño, que al no contar con una interfaz analógica para realizar el chequeo de intensidad lumínica a la salida se optó por una respuesta digital plasmada en el nivel discreto transmitido mediante UART a la PC. El planteo de esta manera permite obtener una simulación válida de un *control gear* de tipo LED.

Los bancos de prueba fueron también escritos en VHDL y se simularon con GHDL 0.31 [7]. Al cumplirse satisfactoriamente con los tests de todos los comandos pudo comprobarse que el *core* cumple con los requisitos.

V. RESULTADOS

En el Cuadro II pueden apreciarse los resultados de la síntesis del *core* para una Spartan 6.

La BRAM insumida se debe a la emulación de EEPROM, necesaria para una implementación real. Hay que considerar que la cantidad de FFs y LUTs puede disminuir significativamente en la versión final, dado que al ser la actual para el primer prototipo a fabricar, se replica información de la memoria en registros internos, que una vez validado pueden ser eliminados.

En la introducción de este trabajo, se indicó que no se encontraron versiones en *hardware* del protocolo, sólo en *firmware*. Comandos como el *reset*, que indica si los valores en memoria son los *default*, parecen útiles sólo para pruebas y consumen un *hardware* considerable (en este caso, implica operaciones *and* de 28 comparaciones). En el Cuadro de resultados puede apreciarse una prueba con dicho comando deshabilitado, con pequeña disminución de FFs pero más de 100 LUTs de ahorro. Para un diseño en FPGA, puede ser útil indicar su utilización mediante *generic*.

VI. CONCLUSIONES

Se obtuvo la implementación en VHDL de un *core Control Gear* DALI del tipo LED. Aunque el objetivo final es fabricar un circuito integrado, se prototipo y validó sobre una FPGA. Para una aplicación real sobre estos dispositivos, hará falta implementar el manejo de la NVM, lo cuál dependerá del *hardware* externo utilizado en cada caso.

La utilización de los casos de prueba sugeridos por la especificación permitió despejar ambigüedades de la misma y realizar correcciones.

El uso de VHDL 93 estándar permite sintetizar el desarrollo para FPGAs e ICs sin cambios en el código.

El trabajo futuro implica la fabricación del circuito integrado y su validación, tras la cuál se podrán realizar corrección y optimizaciones destinadas a reducir cantidad de registros duplicados. Adicionalmente, para su uso con FPGAs, sería deseable incluir *generics* que deshabiliten comandos no precisados, en busca de hacer más compacto el diseño.

REFERENCIAS

[1] *Digital Addressable Lighting Interface (DALI) Implementation Using MSP430 Value Line Microcontrollers*, Texas Instruments, 10 2012.

[2] *Digital addressable lighting interface-Part 101: General requirements - System components*, IEC Std. 62 386-101, Rev. 1.0, 2009.

[3] *Digital addressable lighting interface-Part 102: General requirements - Control gear*, IEC Std. 62 386-102, Rev. 1.0, 2009.

[4] *Digital addressable lighting interface-Part 207: Particular requirements for control gear - LED modules (device type 6)*, IEC Std. 62 386-207, Rev. 1.0, 2009.

[5] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, and E. Barker, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," DTIC Document, Tech. Rep., 2001.

[6] S. Hazwani, S. Khan, M. U. Siddiqi, K. A. Al-Khateeb, M. H. Habaebi, and Z. Shahid, "Randomness analysis of pseudo random noise generator using 24-bits lfsr," in *Intelligent Systems, Modelling and Simulation (ISMS), 2014 5th International Conference on*, 2014, pp. 772–774.

[7] T. Gingold. Where VHDL meets gcc. [Online]. Available: http://ghdl.free.fr/