XXII CONGRESO NACIONAL INGENIERÍA MECÁNICA
19-21 SEP 2018 madrid
UNED

Implementación de Observador de Estados basado en Modelos Multicuerpo en Tiempo real en Plataformas Embebidas

Antonio J. Rodríguez[1], Roland Pastorino[2], Emilio Sanjurjo[1], Alberto Luaces[1], Miguel Á. Naya[1]

[1] *Departamento de Ingeniería Naval e Industrial, Universidade da Coruña, antonio.rodriguez.gonzalez@udc.es, emilio.sanjurjo@udc.es, aluaces@udc.es, minaya@udc.es*

[2] *Siemens Industry Software NV, Belgium, roland.pastorino@siemens.com*

Durante los últimos años, la industria del automóvil ha hecho un esfuerzo continuo por mejorar la seguridad y el control de los vehículos. Los avances en estos campos, han ido ligados al desarrollo de algoritmos de control cada vez más complejos, que necesitan de una amplia cantidad de sensores para monitorizar el comportamiento del vehículo. El principal problema es que, en ocasiones, no se dispone de la información deseada porque el sensor necesario tiene un coste excesivo o no existe. Una alternativa la proporcionan los observadores de estados basados en modelos multicuerpo. El principio básico de esta idea es que un modelo multicuerpo del vehículo sea corregido en tiempo real por la información obtenida de un número reducido de sensores situados en el vehículo. De esta forma, el modelo refleja el comportamiento del vehículo y se puede extraer información directamente del modelo sin necesidad de instalar sensores adicionales. Sin embargo, la limitada capacidad de los procesadores de las Unidades Electrónicas de Control (ECUs) de los vehículos, impide que esta solución se pueda ejecutar a bordo en tiempo real. Actualmente, las ECUs están mejorando en cuanto a capacidad de cálculo debido al desarrollo de los procesadores heterogéneos que embeben dos procesadores diferentes en un solo chip, permitiendo mejorar el rendimiento total del procesador. En particular, destacan los procesadores heterogéneos compuestos por un procesador ARM y una Field Programmable Gate Array (FPGA) como co-procesador para acelerar la ejecución de parte del código. El objetivo de este trabajo es implementar en tiempo real un observador de estados basado en modelos multicuerpo en un procesador heterogéneo, con el fin de poder dotar a vehículos comerciales de sensores virtuales.

1. Introducción

En la industria de la automoción, gran parte de los esfuerzos que se realizan para mejorar la seguridad y el confort en los vehículos dependen de información proporcionada por sensores sobre el estado del vehículo. Por norma general, cuanta mayor información esté disponible, mejores decisiones se podrán tomar. Por ejemplo, durante los primeros test en circuito de un nuevo modelo, recoger la mayor cantidad de datos posible permite corregir errores en la dinámica del vehículo. Por otro lado, para los sistemas de control que se instalan a bordo del vehículo, disponer de una mayor cantidad de información resulta clave a la hora de desarrollar sistemas más precisos y seguros, como suspensiones activas o sistemas de ayuda de frenado. Sin embargo, en muchas ocasiones, no es posible instrumentar un vehículo con todos los sensores necesarios para recoger toda la información deseada, bien porque el sensor es costoso y/o difícil de situar,o bien porque no existe un sensor que proporcione los datos que se desea obtener. Ante esta situación, se recurre al empleo de sensores virtuales. Esto es, utilizando un modelo del

vehículo simulado en tiempo real durante la conducción, y con ayuda de ciertos sensores emplazados en el vehículo real para corregir la simulación adaptándola a la situación real, se puede extraer del modelo virtual toda la información deseada sin necesidad de aumentar el número de sensores en el vehículo. El principal inconveniente de esta alternativa es lograr la simulación y la corrección del modelo virtual en tiempo real en las plataformas que suelen llevar a bordo los vehículos actuales.

En los vehículos, los controladores anteriormente mencionados se implementan a bordo en las Unidades Electrónicas de Control (ECUs). Así, un vehículo puede tener varias ECUs dependiendo de los sistemas de control de los que disponga. Esto da lugar a que haya una ECU para cada tarea sencilla como el control de las luces, aire acondicionado o regulación del asiento y otras (de mayor potencia) para implementar algoritmos de control más complejos para sistemas activos.

Con el fin de dotar de más potencia computacional a las ECUs, y poder así implementar algoritmos de control más complejos relacionados con la visión artificial y el empleo de radares, se están comenzando a emplear ECUs de nueva generación en los vehículos. Este tipo de ECUs están formadas por un procesador heterogéneo que suele estar compuesto por un procesador principal del tipo ARM y una Field Programmable Gate Array (FPGA) como co-procesador. Por otro lado, existen también otros tipos de procesadores donde se combina el ARM con varias GPUs, como alternativa al empleo de las FPGAs. Así mismo, este tipo de procesadores es similar al que se emplea en los sistemas de adquisición de datos.

Los procesadores ARM tienen una arquitectura del tipo RISC (conjunto de instrucciones reducidas), con unas características que les permite tener un menor coste y consumo de energía y una mejor capacidad de disipación de calor que los procesadores basados en arquitectura CISC (conjunto de instrucciones complejas), que son los empleados en ordenadores, por ejemplo. Esto hace que los procesadores ARM sean adecuados en este tipo de aplicaciones, como ECUs o smartphones.

Por otro lado, una FPGA es un dispositivo que permite combinar libremente los recursos físicos que posee (puertas lógicas que se conectan unas con otras) con el fin de crear una unidad dedicada a una aplicación específica, aumentando el rendimiento que tendría la misma aplicación en el procesador ARM. Es por esto que también se conocen las FPGAs como aceleradores hardware. Su principal limitación es la cantidad de recursos disponible, aunque es tecnología que está en continuo crecimiento hacia FPGAs con más recursos y menor coste.

La aparición de estos nuevos procesadores podría significar la posibilidad de implementar sensores virtuales en los vehículos comerciales y en los sistemas de adquisición de datos. Hasta ahora, en las ECUs que se instalaban en los vehículos era impensable esta solución, al igual que en los procesadores que se empleaban en los sistemas de adquisición de datos. Sin embargo, la mejora de prestaciones que ofrecen los procesadores heterogéneos supone una oportunidad para implementar un modelo numérico detallado del vehículo que corra en tiempo real y ofrecer por tanto una mayor información sobre el estado del vehículo sin necesidad de instalar una gran cantidad de sensores.

2. Estado del arte

Existen múltiples casos de aplicación de sensores virtuales en tiempo real en el mundo de la automoción. Un caso muy habitual es el de las suspensiones activas [1]. Para conseguir un control óptimo de la suspensión, se sitúan varios sensores como acelerómetros en la suspensión, con el fin de conocer la situación real del mecanismo. La información recogida por los sensores, se envía a una Unidad Electrónica de Control (ECU), donde se encuentra el modelo de la suspensión. Con los datos de los sensores, se corrige el estado de dicho modelo, de forma que los datos extraídos del modelo se corresponden con la situación real de la suspensión.

Otro campo de aplicación se encuentra en el análisis de la dinámica del vehículo durante los test que se realizan en circuitos durante su desarrollo. Ciertos parámetros como el ángulo de deriva o el deslizamiento y las fuerzas del neumático son de gran interés, pero obtenerlos mediante sensores físicos resulta imposible o es excesivamente complejo. En esta situación, poder estimar dichos parámetros a partir de un modelo del vehículo simulado en tiempo real en un Sistema de Adquisición de Datos sería una posible solución.

Como ya se ha comentado previamente, el principal problema es conseguir que la simulación se ejecute en tiempo real en una plataforma de las que se emplean comúnmente en los vehículos o en los sistemas de adquisición de datos. Como primera opción, es posible emplear modelos analíticos del vehículo o de parte de él [2]. Estos modelos son relativamente sencillos y permiten reducir el tiempo de simulación. Sin embargo, algunos parámetros necesarios para definirlos no se corresponden con variables fácilmente determinables y, dada la sencillez del modelo, la precisión que se puede lograr es limitada.

Como alternativa a los modelos analíticos, en [3] se propone el empleo de un modelo multicuerpo. La ventaja de este tipo de modelos es que permiten obtener una mayor cantidad de variables con mayor precisión que los modelos analíticos. El principal problema es que tienen un mayor coste computacional, lo que hace que su simulación en

tiempo real combinada con un estimador de estados sea difícil. No obstante, en [4], se ha desarrollado un observador que mejora en tiempo de ejecución a los expuestos hasta ahora, lo cual posibilita su implementación en un procesador como los empleados actualmente a bordo de vehículos y en los sistemas de adquisición de datos. Sin embargo, en todos estos trabajos las soluciones propuestas se ejecutan en plataformas diferentes a las instaladas en los vehículos comerciales y con mayor potencia computacional. El siguiente paso lógico es, por tanto, implementar el observador de estados basado en un modelo multicuerpo en un procesador heterogéneo como los mencionados anteriormente.

En [5], se muestran unos primeros resultados de ejecución de modelos multicuerpo en este tipo de procesadores. De los resultados obtenidos, se puede concluir que el empleo de una FPGA puede ayudar a conseguir alcanzar el tiempo real la estimación de estados. No obstante, los resultados se corresponden únicamente a simulación de modelos multicuerpo. Cuando se combina el modelo multicuerpo con un observador de estados, el tiempo de la simulación se ve incrementado. Por tanto, implementar un observador de estados basado en un modelo multicuerpo en un procesador heterogéneo y conseguir que vaya a tiempo real será el objetivo de este trabajo.

3. Implementación

Para implementar el modelo multicuerpo y el observador de estados, se dispone de un procesador tipo ZYNQ de *Xilinx*, en concreto el modelo Zynq7020 SoC. Este procesador incluye un ARM Cortex A-9, capaz de alcanzar una frecuencia máxima de 1 GHz, y una FPGA Artix-7, que dispone de 85.000 unidades lógicas. A continuación, se presentará el modelo de vehículo seleccionado, así como la formulación multicuerpo empleada, y el observador de estados implementado.

3.1. Modelo multicuerpo del vehículo

El vehículo que se va a implementar es un Nissan Leaf, que posee suspensión delantera del tipo MacPherson y suspensión Twist-Beam en la parte trasera. El modelo del vehículo constará de 16 grados de libertad (6 del chasis, 4 de las suspensiones, 4 del giro de las ruedas y 2 de la dirección de las ruedas delanteras), y se le ha añadido un modelo de neumático para poder estimar las fuerzas que se producen en los mismos durante la simulación. El primer paso para realizar el modelo es seleccionar el tipo de coordenadas en las que definir todo el mecanismo. Existen diferentes tipos de coordenadas [6], cada uno de los cuales posee sus ventajas y sus desventajas. Sin embargo, de acuerdo a lo expuesto en [5], el tipo de coordenadas más adecuado para esta aplicación son las coordenadas relativas, combinadas con un set de coordenadas absolutas para los seis grados de libertad del chasis. De esta forma, se consigue un definir el modelo con un número reducido de variables, de modo que se pueda optimizar el empleo de una FPGA para acelerar partes de la simulación, dado que el espacio en la FPGA es limitado.

El empleo de coordenadas relativas tiene el inconveniente de que definir las ecuaciones del movimiento resulta complejo, especialmente cuando en el modelo se encuentran mecanismos de cadena cerrada como las suspensiones, donde aparecen dependencias entre variables. Como solución, en este trabajo se emplea la formulación semi-recursiva presentada por Cuadrado *et al.* [7], dando lugar a un modelo de 24 variables.

El método semi-recursivo define un set doble de coordenadas durante el modelizado: un conjunto de seis coordenadas para cada cuerpo (tres traslaciones y tres rotaciones) y las coordenadas relativas del mecanismo. Las ecuaciones de la dinámica multicuerpo se expresan en las coordenadas de cada cuerpo y, empleando una proyección de velocidades, se transforman en ecuaciones expresadas en coordenadas relativas. Sin embargo, cuando las coordenadas relativas son dependientes (como es el caso en la definición de las suspensiones), deben imponerse ecuaciones de restricción, lo que incrementa el coste computacional de la simulación. En (1), se muestran las ecuaciones de movimiento utilizando la formulación Lagrangiana Aumentada de Índice-3 [7].

$$\mathbf{M}\ddot{\mathbf{z}} + \boldsymbol{\Phi}_{\mathbf{z}}^{\mathbf{t}}\boldsymbol{\alpha}\boldsymbol{\Phi} + \boldsymbol{\Phi}_{\mathbf{z}}^{\mathbf{t}}\boldsymbol{\lambda}^{*} = \mathbf{Q} \tag{1}$$

donde $\mathbf{z}$ son las coordenadas relativas, $\mathbf{M}$ es la matriz de masa del modelo expresada en términos de las coordenadas relativas, $\boldsymbol{\Phi}$ es el vector de restricciones, $\boldsymbol{\Phi}_{\boldsymbol{z}}$ es la matriz Jacobiana de las restricciones, $\boldsymbol{\alpha}$ es el penalizador, $\mathbf{Q}$ es el vector de las fuerzas aplicadas y de las dependientes de la velocidad y $\boldsymbol{\lambda}^{*}$ es el vector de los multiplicadores de Lagrange.

Expresar los términos de (1) en coordenadas relativas no es trivial. Sin embargo, utilizando las coordenadas de cada cuerpo (2), los términos $\mathbf{M}$ y $\mathbf{Q}$ se pueden formular de una forma más sencilla [7].

$$\mathbf{Z} = \begin{Bmatrix} \dot{\boldsymbol{s}} \\ \boldsymbol{\omega} \end{Bmatrix} \tag{2}$$

donde $\dot{\boldsymbol{s}}$ es la velocidad del punto del cuerpo que en ese instante coincide con el origen de coordenadas, y $\boldsymbol{\omega}$ es la velocidad angular del sólido.

Como se indica en [7], la matriz de masas y el vector de fuerzas generalizadas expresados en las coordenadas **Z** es sencilla de obtener y se muestra en las ecuaciones (3) y (4).

$$\bar{\mathbf{M}} = \begin{bmatrix} m\mathbf{I_3} & -m\tilde{\mathbf{g}} \\ m\tilde{\mathbf{g}} & \mathbf{J} - m\tilde{\mathbf{g}}\tilde{\mathbf{g}} \end{bmatrix} \tag{3}$$

$$\bar{\mathbf{Q}} = \begin{Bmatrix} \mathbf{f} - \boldsymbol{\omega} \times (\boldsymbol{\omega} \times m\mathbf{g}) \\ \mathbf{n} - \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} + \mathbf{g} \times (\mathbf{f} - \boldsymbol{\omega} \times (\boldsymbol{\omega} \times m\mathbf{g})) \end{Bmatrix} \tag{4}$$

donde $\bar{\mathbf{M}}$ y $\bar{\mathbf{Q}}$ son la matriz de masas y el vector de fuerzas generalizadas expresadas en las coordenadas del cuerpo **Z** respectivamente**,** *m* es la masa del sólido, $\mathbf{I_3}$ es una matriz identidad de 3x3, **g** es la posición global del centro de gravedad del sólido, $\tilde{\mathbf{g}}$ es la matriz antisimétrica de **g**, **J** es el tensor de inercia del sólido con respecto al sistema de referencia paralelo al sistema de referencia en el centro de masas del sólido, **f** es el vector de fuerzas aplicadas al sólido y **n** es el vector de momentos aplicados con respecto al centro de masas del sólido.

Posteriormente, será necesario aplicar una relación entra ambos tipos de coordenadas para obtener **M** y **Q** en coordenadas relativas. Puede definirse la matriz **R** tal que,

$$\mathbf{Z} = \mathbf{R}\dot{\mathbf{z}} \tag{5}$$

donde **Z** son las coordenadas de cada cuerpo y $\dot{\mathbf{z}}$ la velocidad de las coordenadas relativas. Si se aplica esta relación a las ecuaciones dinámicas del movimiento,

$$\mathbf{R^t}\bar{\mathbf{M}}\mathbf{R}\ddot{\mathbf{z}} = \mathbf{R^t}(\bar{\mathbf{Q}} - \bar{\mathbf{M}}\dot{\mathbf{R}}\dot{\mathbf{z}}) \tag{6}$$

La expresión (6), implica que la matriz de masas **M** y el vector de fuerzas **Q** del modelo expresados en coordenadas relativas será,

$$\mathbf{M} = \mathbf{R^t}\bar{\mathbf{M}}\mathbf{R} \tag{7}$$

$$\mathbf{Q} = \mathbf{R^t}(\bar{\mathbf{Q}} - \bar{\mathbf{M}}\dot{\mathbf{R}}\dot{\mathbf{z}}) \tag{8}$$

Una vez que los términos de (6) son conocidos, es posible integrar las ecuaciones y resolver la dinámica del modelo para cada paso de tiempo. Empleando el integrador implícito de la regla trapezoidal combinado con el método iterativo de Newton-Raphson, se obtienen un conjunto de posiciones que satisface tanto la ecuación (1) como el conjunto de ecuaciones de restricción del mecanismo. Una vez que se obtiene las posiciones, las velocidades y aceleraciones del paso de tiempo n+1 deben calcularse. De esta forma, el estado del modelo queda determinado para cada paso de tiempo [7].

3.2. Observador de estados: errorEKF

Como se ha mencionado anteriormente, se ha seleccionado como observador de estados el propuesto en [8], conocido como *errorEKF* con estimación de fuerzas. El conjunto de sensores que se necesita para corregir el modelo está formado por un conjunto de sensores comunes en los vehículos actuales, como un acelerómetro y un giróscopo en el chasis, un GPS y sensores angulares en las ruedas, a los que hay que sumar unos sensores lineales en las suspensiones que midan la elongación de la misma.

El *errorEKF* con estimación de fuerzas [8] es una versión indirecta del filtro de Kalman que permite realizar estimación de fuerzas de manera eficiente y sin una implementación compleja, de forma que las correcciones del filtro se apliquen no sólo a nivel de posición y velocidad, sino que también a nivel de aceleraciones y evaluando el error en las fuerzas que actúan sobre las variables del vector de estados. Un esquema simplificado de este filtro se puede ver en la Figura 1.

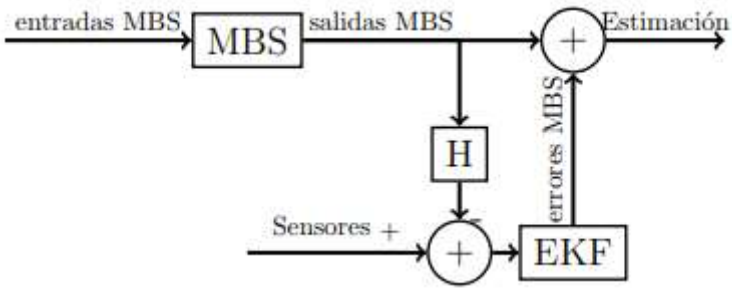

Figura 1: Esquema simplificado del *errorEKF*

Para cada paso de tiempo, primero se ejecuta el modelo multicuerpo, obteniendo el valor de las posiciones y velocidades de las variables del modelo. Posteriormente, el observador estima el error cometido por el modelo

atendiendo a la información de los sensores disponibles. Se define por tanto el vector de estados el formado por los errores en posición, en velocidad y aceleración de los grados de libertad del mecanismo, de tal forma que,

$$\mathbf{x^T} = [\Delta \mathbf{z^T},\ \Delta \dot{\mathbf{z}}^\mathbf{T}, \Delta \ddot{\mathbf{z}}^\mathbf{T}] \tag{9}$$

$$\mathbf{z} = \boldsymbol{z}_{\mathbf{MBS}} + \Delta \mathbf{z} \tag{10}$$

$$\dot{\mathbf{z}} = \dot{\mathbf{z}}_{\mathbf{MBS}} + \Delta \dot{\mathbf{z}} \tag{11}$$

$$\ddot{\boldsymbol{z}} = \ddot{\mathbf{z}}_{\mathbf{MBS}} + \Delta \ddot{\mathbf{z}} \tag{12}$$

donde $\boldsymbol{z}_{\mathbf{MBS}}$, $\dot{\mathbf{z}}_{\mathbf{MBS}}$ y $\ddot{\mathbf{z}}_{\mathbf{MBS}}$ son los valores de $\mathbf{z}$, $\dot{\mathbf{z}}$ y $\ddot{\boldsymbol{z}}$ estimadas por el modelo multicuerpo antes de aplicar las correcciones del observador.

Una vez aplicada la corrección de los sensores, se obtienen la estimación de los errores en posición, velocidad y aceleración sobre las coordenadas independientes del modelo y, por tanto, el valor corregido de las variables del modelo [8]. Sin embargo, para corregir el estado de todo el modelo multicuerpo, es necesario obtener los errores en el resto de variables que componen el modelo. Por tanto, es necesario aplicar las ecuaciones de restricción con el nuevo valor de las variables independientes, asegurando entonces que las posiciones, velocidades y aceleraciones de todas las variables del modelo cumplen las ecuaciones de restricción y son coherentes con los valores de las variables independientes.

Para cumplir las restricciones a nivel de posición, se resuelve un problema de posición, que es iterativo y conlleva un incremento del coste computacional. Sin embargo, como las correcciones en el error son pequeñas, es posible linealizar el problema de posición [8]. Si se define como $\mathbf{q}$ el conjunto de variables que definen el sistema (independientes y dependientes), será suficiente con resolver el siguiente sistema,

$$\boldsymbol{\Phi}_\mathbf{q} \Delta \mathbf{q} = \mathbf{0} \tag{13}$$

Las correcciones en velocidades y aceleraciones se realizan después de la corrección en posiciones, resolviendo los problemas de velocidad y aceleraciones correspondientes. Al ser un filtro de fuerzas, una vez que las aceleraciones se han corregido, es posible obtener el error en las fuerzas $\mathbf{Q}$ del modelo, consiguiendo una mayor precisión en la estimación. Sin embargo, como hay más coordenadas que grados de libertad, existen infinitas combinaciones de fuerzas que producen el miso efecto. Es por esto que en [8], se asume que la solución de fuerzas necesarias corregir el modelo serán aquellas que se obtienen de los errores en las aceleraciones de los grados de libertad (variables independientes), de forma que,

$$\Delta \mathbf{Q}_\mathbf{i}^{\mathrm{dep}} = \mathbf{0} \tag{14}$$

$$\Delta \mathbf{Q}_\mathbf{i}^{\mathrm{indep}} = \Delta \mathbf{Q}_{\mathbf{i-1}}^{\mathrm{indep}} + \mathbf{M} \Delta \ddot{\mathbf{z}} \tag{15}$$

donde el subíndice *i* se corresponde con el paso de tiempo. Este incremento de fuerzas se añade al vector de fuerzas Q del modelo y se aplica en la integración del siguiente paso de tiempo, de tal forma que el error esperado en las aceleraciones obtenidas del modelo multicuerpo es nulo.

3.3. Implementación en procesador

Durante la simulación a bordo del vehículo, es necesario establecer una comunicación entre los sensores y el procesador, así como una interfaz que permita analizar los datos que se obtienen de manera cómoda y sencilla. Esto significa que será necesario disponer de un sistema operativo en el procesador. Debido a que se trata de un procesador embebido, es importante que el sistema operativo no consuma demasiados recursos del procesador y únicamente realice las tareas necesarias para proporcionar la información de los sensores que necesita el observador para corregir el modelo. Es por esto se emplea en este trabajo una distribución de *Linux* para sistemas embebidos conocida como *PetaLinux*, desarrollada por *Xilinx*.

Como se ha comentado anteriormente, el procesador empleado es un procesador heterogéneo: posee dos procesadores de diferente tipo embebidos en un único chip. En este trabajo, el procesador Zynq que se emplea posee un procesador ARM y una FPGA. Por tanto, a la hora de implementar el modelo y el observador y con el fin de extraer el máximo rendimiento del procesador, surgen diferentes combinaciones posibles entre la FPGA y el ARM. Como se ha indicado en la introducción, el ARM es un procesador de arquitectura RISC, ideal para aplicaciones empotradas dado su bajo consumo de energía. Sin embargo, y con la finalidad de aumentar su rendimiento, se combina con una FPGA. La FPGA es un elemento que se puede programar para ejecutar únicamente una tarea determinada de las que se estén ejecutando en el procesador ARM, acelerándola y reduciendo los tiempos de ejecución.

Sin embargo, dado que la FPGA tiene espacio limitado, es importante seleccionar las partes del modelo que ralentizan en mayor medida la simulación y analizar la posibilidad de implementarlas en la FPGA. Para ello se ha realizado un *profiling* del código, donde se ha observado que la resolución del sistema de ecuaciones necesario

para determinar los incrementos de las variables del modelo es una de las partes de mayor coste computacional. Por otro lado, el cálculo de la matriz de masas que debe realizarse en cada iteración, y que es una de las principales desventajas del empleo de las coordenadas relativas, también consume una gran parte del tiempo de ejecución.

Para resolver el sistema de ecuaciones comentado, se recurre al algoritmo de Gauss-Jordan que se propone en [9]. Sin embargo, la FPGA de la que se dispone en este trabajo es mucho más pequeña que la empleada en [9], por lo que no se podrán alcanzar los mismos resultados para los tamaños de matriz que se encuentran en el modelo. En cuanto al cálculo de la matriz de masas en cada paso de tiempo, el número de variables que se necesitan es demasiado grande para poder almacenarlos en la FPGA para el cálculo, por lo que esta opción queda descartada.

Como alternativa a estas propuestas, surge también el empleo de *look up tables* (o tablas cinemáticas) que representan la cinemática de las suspensiones [10]. De esta forma, se reduce considerablemente el tamaño del modelo y se eliminan las variables dependientes del mismo, de forma que se simplifica enormemente el modelo y, consecuentemente, se reduce el tiempo de ejecución. El principal problema de esta solución es que se desprecian ciertos efectos inerciales de las suspensiones, ya que son eliminadas de la simulación y reemplazadas por relaciones cinemáticas.

4. Resultados

Para evaluar la capacidad del procesador heterogéneo y las ventajas de las diferentes alternativas propuestas, se ejecutarán diferentes maniobras de 10 s de duración y con un paso de tiempo de 4 ms. Como se ha comentado anteriormente, se comprobarán los resultados que se obtienen de derivar a la FPGA la resolución del sistema de ecuaciones y el empleo de las *look up tables*.

En la Tabla 1 se muestran los resultados obtenidos. Como referencia, se tomará la ejecución del conjunto modelo-observador en el ARM, utilizando en el ARM el mismo algoritmo Gauss-Jordan que se ha implementado en la FPGA para resolver el sistema de ecuaciones del modelo multicuerpo y obtener los incrementos de posiciones en cada paso de tiempo. De esta forma, se puede evaluar la mejora obtenida con el empleo de la FPGA con respecto a su implementación en el ARM.

La simulación de referencia, no obstante, puede ser mejorada (sin necesidad de emplear la FPGA) con el empleo de una librería como Eigen, que dispone de algoritmos optimizados que mejoran el rendimiento en el procesador ARM del algoritmo Gauss-Jordan implementado en este trabajo, con la ventaja de que su implementación es directa. Es por esto, que otra opción que se ha considerado es la ejecución del modelo multicuerpo y el observador utilizado la librería Eigen en el procesador ARM.

Tabla 1: Resultados obtenidos

	ARM	FPGA	Tiempo (s)
Referencia	Modelo + observador (Gauss-Jordan)	-	24.57
Opción 1	Modelo + observador (Eigen)	-	14.22
Opción 2	Modelo + observador	Resolución sistema de ecuaciones	13.94
Opción 3	Modelo (*look up tables*) + observador	-	8.39

Como se puede apreciar, el modelo multicuerpo combinado con el observador en el ARM no alcanza tiempo real, por lo que es necesario recurrir a las alternativas propuestas. Empleando la librería de Eigen, se logra reducir en gran medida el tiempo de ejecución, como era de esperar dado el nivel de optimización de la librería con respecto al algoritmo implementado. Sin embargo, el resultado muestra que el empleo de una librería optimizada no es suficiente para alcanzar tiempo real. Implementando el algoritmo de Gauss Jordan en la FPGA, mejora el tiempo con respecto a su implementación en el procesador ARM, pero no alcanza el rendimiento esperado y apenas supone una mejora de tiempo en la simulación con respecto a la librería Eigen. Esta situación puede ser debida a las limitaciones de espacio de la FPGA empleada, ya que no dispone de los recursos suficientes para implementar el algoritmo propuesto en [9] y alcanzar las prestaciones mostradas.

La última alternativa propuesta, que recurre al empleo de las *look up tables* para reducir la complejidad del modelo, dando lugar a una mejora del tiempo de la simulación, permitiendo que se pueda simular en tiempo real el observador de estados basado en modelos multicuerpo en un procesador embebido, utilizando en este caso únicamente el procesador ARM.

5. Conclusiones

La tendencia que está siguiendo la industria del automóvil lleva al desarrollo de algoritmos de control más sofisticados que requieren de una mayor información de sensores. Actualmente, se está comenzando a instalar cámaras, radares y sensores infrarrojos para ciertas aplicaciones que ayuden a la conducción del vehículo. Sin embargo, hay cierta información relativa a la dinámica del vehículo que no puede ser obtenida con el empleo de

sensores prácticos. Combinando un modelo multicuerpo del vehículo con un observador de estados, pueden estimarse variables que no pueden obtenerse a partir de sensores.

En la implementación de esta solución, resulta clave el desarrollo de los procesadores heterogéneos, que empiezan a formar parte de las ECUs de los vehículos comerciales y de los nuevos sistemas de adquisición de datos. Con el aumento de la potencia computacional que ofrecen estos procesadores, surge la posibilidad de implementar observadores de estados basados en modelos multicuerpo a bordo de un vehículo en tiempo real.

En este trabajo, se ha modelizado un vehículo real empleando coordenadas relativas y una formulación semi-recursiva combinado con un observador del tipo *errorEKF* y se ha implementado en un procesador Zynq7020. Se han explorado diferentes posibilidades de implementación tratando de aprovechar la presencia de un procesador ARM combinado con una FPGA. De los resultados obtenidos, se puede concluir que el tamaño de la FPGA no es suficiente para conseguir mejorar en una cantidad apreciable el tiempo de ejecución. Actualmente, existen FPGAs embebidas en procesadores heterogéneos con 100 veces más recursos que la empleada en este trabajo, lo que amplía las posibilidades de acelerar el algoritmo que se implemente en la FPGA. Sin embargo, como alternativa, si se reemplazan las suspensiones por tablas cinemáticas de las mismas, la mejora de tiempo es notable.

En trabajos futuros, se explorará la implementación del modelo y el observador en un procesador con una FPGA de mayor capacidad, para evaluar la posibilidad de alcanzar el tiempo real sin necesidad de emplear tablas cinemáticas, de forma que se aumente la precisión de los datos obtenidos durante la simulación. También resulta interesante evaluar la posibilidad de combinar el modelo basado en tablas cinemáticas con la FPGA, buscando aumentar más aún el rendimiento de observador de estados basado en el modelo multicuerpo del vehículo. Es importante tener en cuenta que, en una implementación a bordo, habrá que añadir el tiempo que se consuma en recibir y procesar los datos de los diferentes sensores, por lo que es importante lograr la mayor reducción de tiempo posible a fin de garantizar la estimación en tiempo real.

6. Agradecimientos

Este trabajo se ha podido realizar gracias al apoyo del Ministerio de Economía y Competitividad Español (MINECO), bajo el proyecto TRA 2014-59435-P, cofinanciado por la Unión Europea a través del programa EFRD.

7. Referencias

[1] Chokor A., Talj R., Charara A., Shraim H., Francis C., "Active Suspension Control to Improve Passengers Comfort and Vehicle's Stability". *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC).* Rio de Janeiro, Brasil (2016)

[2] Wenzel T., Burnham K., Blundell M., Williams R., "Dual extended Kalman filter for vehicle state and parameter estimation", *Vehicle System Dynamics*, 44, 153-171 (2006)

[3] Cuadrado J., Dopico D., Perez J. A., Pastorino R., "Automotive observers based on multibody models and the extended Kalman filter". *Multibody System Dynamics*, Vol. 27, No. 1, pp. 3–19 (2012)

[4] Sanjurjo E., Naya M. A., Blanco-Claraco J. L., Torres-Moreno J. L., Giménez-Fernández A., "Accuracy and efficiency comparison of various nonlinear Kalman filters applied to multibody models", *Nonlinear Dyn*, 88, 1935-1951 (2017)

[5] Rodríguez A. J., Pastorino R., Naya M. A., Sanjurjo E., "Virtual sensing on automotive embedded heterogeneous platforms". *15th European Automotive Congress (EAEC)*. Madrid, España (2017)

[6] García de Jalón J., Bayo E., *Kinematic and Dynamic Simulation of Multibody Systems: The Real Time Challenge*, Springer-Verlag (1994)

[7] Cuadrado J., Dopico D., Gonzalez M., Naya M. A., "A Combined Penalty and Recursive Real-Time Formulation for Multibody Dynamics", *Journal of Mechanical Design*, vol. 126 (2004)

[8] Sanjurjo E., Dopico D., Luaces A., Naya M. A., "State and force observers based on multibody models and the indirect Kalman filter", *Mechanical Systems and Signal Processing*, 106, 210-228 (2018)

[9] David J.P., "Low latency and division free Gauss-Jordan solver in floating point arithmetic", *Journal of Parallel and Distributed Computing*, Vol. 106, 185-193 (2017)

[10] Cuadrado J., Vilela D., Iglesias I., Martin A. Peña A., "A Multibody Model to Assess the Effect of Automotive Motor In-Wheel Configuration on Vehicle Stability and Comfort", *ECCOMAS Thematic Conference Multibody Dynamics*, 457-458, Zagreb, Croacia (2013)