Título:

Integración y Documentación de un Flujo de Diseño de Circuitos Integrados utilizando Software Libre.

Descripción:

El objetivo inmediato del trabajo es producir documentación con el mayor nivel de detalle posible sobre el flujo de diseño de un sistema digital utilizando herramientas de software libre. La motivación del trabajo es lograr la base de conocimiento necesaria (con respecto a las herramientas de software) para diseñar circuitos integrados con tecnología CMOS con herramientas flexibles que permitan la posibilidad de modificarlas y/o mejorarlas, de ser necesario. El objetivo de este trabajo es facilitar el acceso a los diseñadores de circuitos integrados las herramientas de diseño disponibles, por medio de una guía paso a paso y con ejemplos de cómo utilizar las herramientas.

Parte I – Introducción

Microelectrónica en la Universidades Argentinas: El objetivo de esta sección será brindar información sobre el estado actual de la microelectrónica en nuestro país, obtener información sobre cuantos laboratorios realizan trabajos en microelectrónica, cuántas materias de grado y postgrado existen.

También se brindará un panorama sobre los posibles escenarios de desarrollo de la disciplina, en función de las licencias de software que las universidades adopten.

Parte II - Posibilidades que brinda la programación funcional al diseño de circuitos digitales semi-custom.

Utilizar un lenguaje[0] de alto nivel embebido en Haskell[1] para el flujo de diseño, que permite: Descripción parametrizable de los circuitos. Circuitos que se adaptan al entorno[2]. Simulación simbólica. Verificación formal de propiedades de los circuitos descriptos, y la utilización de otras técnicas disponibles para el chequeo de software.

Objetivo: Generar desde la descripción de alto nivel en Lava[Lava], una descripción a nivel de compuertas (en VHDL) de forma automática.

Metodología: Se elije un circuito de complejidad media a implementar, un sumador rápido extraído del paper de Brent-Kung[3], se buscan los algoritmos que lo resuelva, y se implementa en el lenguaje. Luego se crea un modelo de referencia simple (ripple carry adder), se especifican las propiedades matemáticas que debe cumplir dicho circuito, se verifican automáticamente, y luego se realiza un chequeo de equivalencia con el modelo de mayor complejidad, utilizando un SAT solver[4].

Parte III – Place and Route (buscar el equivalente en castellano)

Con la descripción RTL en VHDL usar Electric VLSI[6] para compilar al silicio, es decir hacer place and route con celdas estandar.

Generación automática de los CMOS pads para conectar el sistema diseñado a los pads de salida del die.

Parte IV – Analog Front End y simulación a nivel de transistores.

Se realiza una interfase analógica simple, como pueden ser los CMOS pads y los buffers para conectar las salidas digitales a los CMOS pads. Utilizar gnucap[5] para simular circuitos analógicos con modelos BSIM3V3 ó BSIM4 (según la tecnología de fabricación que se elija). Simulaciones de punto de operación, de continua, alterna, transitoria, transformada de fourier, ruido, etc.

Realizar el post procesamiento de los datos de la simulación y su representación gráfica. Curvas de caracterización de transistores, ganancia y margen de fase, diagramas de ojo, etc.

Objetivos: Brindar una documentación detallada y didáctica de todo el flujo de diseño analógico integrado.

Entregables: Esquemáticos y Layouts del diseño analógico, y la documentación de como se realizaron con las herramientas utilizadas (Electric VLSI, Gnucap, Matplotlib[], Numpy[], Octave). Además todos los scripts que se utilizan para simular, documentados con el mayor nivel de detalles.

Parte V - Layout Digital y Analógico

Realizar el layout del circuito anterior, buscando las posibilidades de automatización de este proceso, partiendo de la descripción a nivel de compuertas que obtengo del lenguaje de descripción de hardware. También se realiza el layout de la interfase analógica. Extracción de los parásitos para simulación.
Objetivos: Brindar una documentación detallada y didáctica de todo el flujo integrado.
Entregables: Una librería con todo el circuito integrado, las celdas estandars y el archivo en formato gds2 listo para enviar a fabricación.

Parte VI - Back End

Realizar todo el trabajo final del chip para integrar la parte analógica y digital, y evitar problemas del tipo de caida de tensión por IR, violación a la electromigración, y otros temas referente al diseño físico del chip. Eso implica hacer los power grids, las celdas de ESD (Electromagnetic Static Discharge), los CMOS pads, etc.

Parte VI – Distribución GNU/Linux ad-hoc

Realizar una distribución con todo el software, los scripts, y la documentación realizada.

Parte VII - Conclusiones

Conclusiones. Líneas de trabajo a futuro.

[0]: Lava, http://www.cs.chalmers.se/~koen/Lava/ (LINK ROTO)
[0']: http://www.cse.chalmers.se/edu/course/TDA956/Papers/lava-tutorial.ps
[1]: http://haskell.org/
[2]: M. Sheeran, “Parallel prefix network generation: an application of functional programming In Hardware Design and Functional Languages,” in *Hardware design and Functional Languages (HFL07), Braga, Portugal*, March 2007
[3]: R. P. Brent and H. T. Kung, “A Regular Layout for Parallel Adders,” IEEE Transaction on Computers, vol. C-31, Issue: 3, pp. 260–264, 1982.
[4]: Minisat, http://minisat.se/
[5]: http://gnucap.org
[6]: http://www.staticfreesoft.celom/