

# Hacia el uso de ASIPs en la Computación Aproximada

Daniel Moya Sánchez

Área de Ingeniería en Computadores  
Instituto Tecnológico de Costa Rica

Proyecto de Diseño de Ingeniería en Computadores

# Agenda

- 1 Contenido
- 2 Presentación del problema
- 3 Estado del arte del problema
- 4 Objetivos
- 5 Descripción de la solución desarrollada
- 6 Resultados obtenidos
- 7 Conclusión
- 8 Referencias

# Problemas en el desarrollo de procesadores

- Área
- Potencia
- Tiempo de ejecución
- Características eléctricas de los CMOS
- Pared de memoria
- Pared de utilización

**Computación Aproximada** como posible solución, implica:

- Identificar secciones, funciones u operaciones aproximables
- Diseñar implementación en hardware o en software
- Evaluación de la calidad

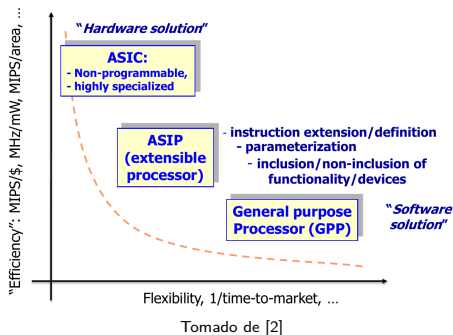
# Propuestas entorno a la Computación Aproximada

## Soluciones en hardware (ASICs):

- SALSA: síntesis de circuito combinacional
- ASLAN: síntesis de circuito secuencial
- ABACUS: síntesis a partir de descripción de comportamiento

## Soluciones en software (GPP):

- Perforación de ciclos
- Calendarización de tareas
- Uso de red neuronal



# Objetivos

## Objetivo principal:

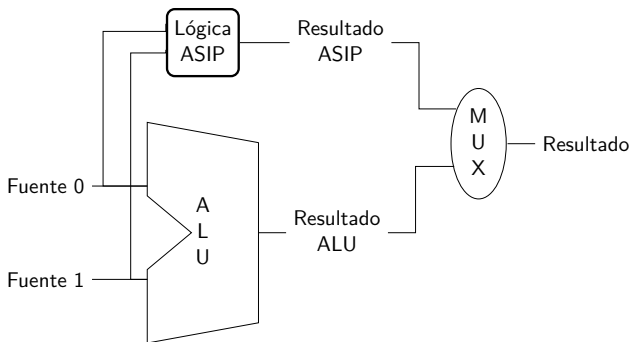
- Evaluar el desempeño de Procesadores de Conjunto de Instrucciones para Aplicaciones Específicas (ASIPs) en aplicaciones tolerantes a errores.

## Objetivos específicos:

- Seleccionar tres aplicaciones tolerantes a errores.
- Desarrollar, para cada aplicación encontrada, una instrucción especial que refleje una operación recurrente.
- Evaluar el desempeño e impacto de cada optimización contra la versión original.

# Solución a nivel de hardware

- Se utiliza procesador con instrucciones comunes (sumas, multiplicaciones, etc.)
- Se agrega el hardware especializado según la aplicación
- Para las instrucciones especiales se utiliza el hardware adicional y no la ALU



# Solución a nivel de software

- Instrucción *euch*:

$$rd = (rs0 - rs1)^2$$

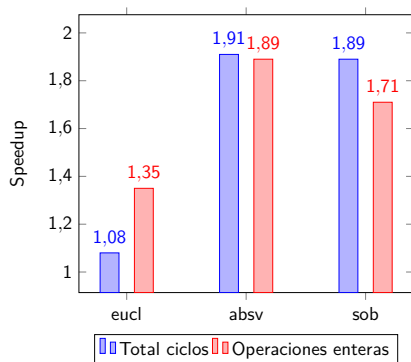
- Instrucción *absv*:

$$rd = rs0 > rs1 ? rs0 - rs1 : rs0 - rs1$$

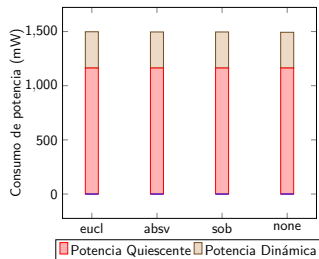
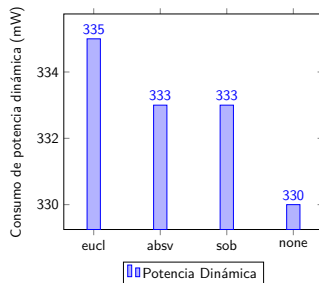
- Instrucción *sob*:

$$rd = rs0^2 + rs1^2$$

# Resultados en ciclos, área y potencia







Métrica	Instr. eucl.	Instru. absv.	Instru. sob.	No Instru. especial
# Slices	3998	4058	4223	3990
% Slices	5 %	5 %	6 %	5 %
# LUTs	6384	6465	6079	6199
% LUTs	9 %	9 %	8 %	8 %





# Conclusión

# Referencias

-  Qiang Xu, Todd Mytkowicz, and Nam Sung Kim. Approximate computing: A survey. *IEEE Design & Test*, 2018.
-  Jörg Henkel. Closing the soc design gap. *Computer*, 36(9):119-121, 2003.
-  Jörg Henkel. Design and architectures for embedded systems (esii), 2006.
-  Swagath Venkataramani, Amit Sabne, Vivek Kozhikkottu, Kaushik Roy, and Anand Raghunathan. Salsa: systematic logic synthesis of approximate circuits. In *Proceedings of the 49th Annual Design Automation Conference*, pages 796-801. ACM, 2012.

# Referencias



Ashish Ranjan, Arnab Raha, Swagath Venkataramani, Kaushik Roy, and Anand Raghunathan. Aslan: Synthesis of approximate sequential circuits. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*, pages 1-6. IEEE, 2014.



Kumud Nepal, Yueting Li, R Iris Bahar, and Sherief Reda. Abacus: A technique for automated behavioral synthesis of approximate computing circuits. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*, pages 1-6. IEEE, 2014.



Stelios Sidiroglou-Douskos, Sasa Misailovic, Henry Hoffmann, and Martin Rinard. Managing performance vs. accuracy trade-offs with loop perforation. In *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, pages 124-134. ACM, 2011.

# Referencias



Cheng Tan, Thannirmalai Somu Muthukaruppan, Tulika Mitra, and Lei Ju. Approximation-aware scheduling on heterogeneous multi-core architectures. In *Design Automation Conference (ASP-DAC), 2015 20th Asia and South Pacific*, pages 618-623. IEEE, 2015.



Hadi, Esmailzadeh, Adrian Sampson, Luis Ceze, and Doug Burger. Neural acceleration for general-purpose approximate programs. *IEEE Micro*, 33(3):16-27, 2013