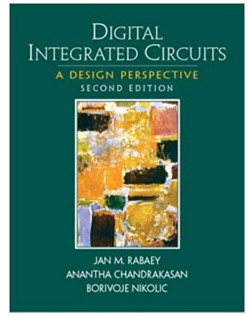
Microeletrônica

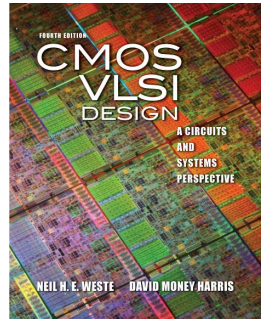
Aula #11 → Metodologias de projetos

- Professor: Fernando Gehm Moraes
- Livro texto:

  Digital Integrated Circuits a Design Perspective - Rabaey

  C MOS VLSI Design - Weste





Revisão das lâminas: 01/junho/2023

Sumário

O início e o estado-da-arte

Introdução

Métodos de Projeto

- **Full Custom**
- **Standard Cells**
- **Geração Automática**
- **Pré-Difundidos**
- **Componentes Programáveis**
  - **Passado – PLAs**
  - **Atualidade - FPGAs**

O início – um pouco de história

(in the start of Intel…) “There was absolutely **no computer assistance** for design rule verification or for logic vs. layout wiring correctness. Physical layout proceeded as highly skilled mask designers drew lines with pencils on very large sheets of gridded Mylar (Figure 3). By **1974**, the result was being digitized on a Calma GDS I system so repeated cells could be handled automatically, instead of being hand drawn every time” (70’s)

“We were still doing it this way for the 8086 first stepping in 1977. That part had 20,000 transistors, and it took two weeks for each of the two design engineers who performed the final task. Both engineers (Peter and Chun-Kit Ng) found 19 of the same 20 errors, which was considered quite a good detection rate for this particular technique. ”

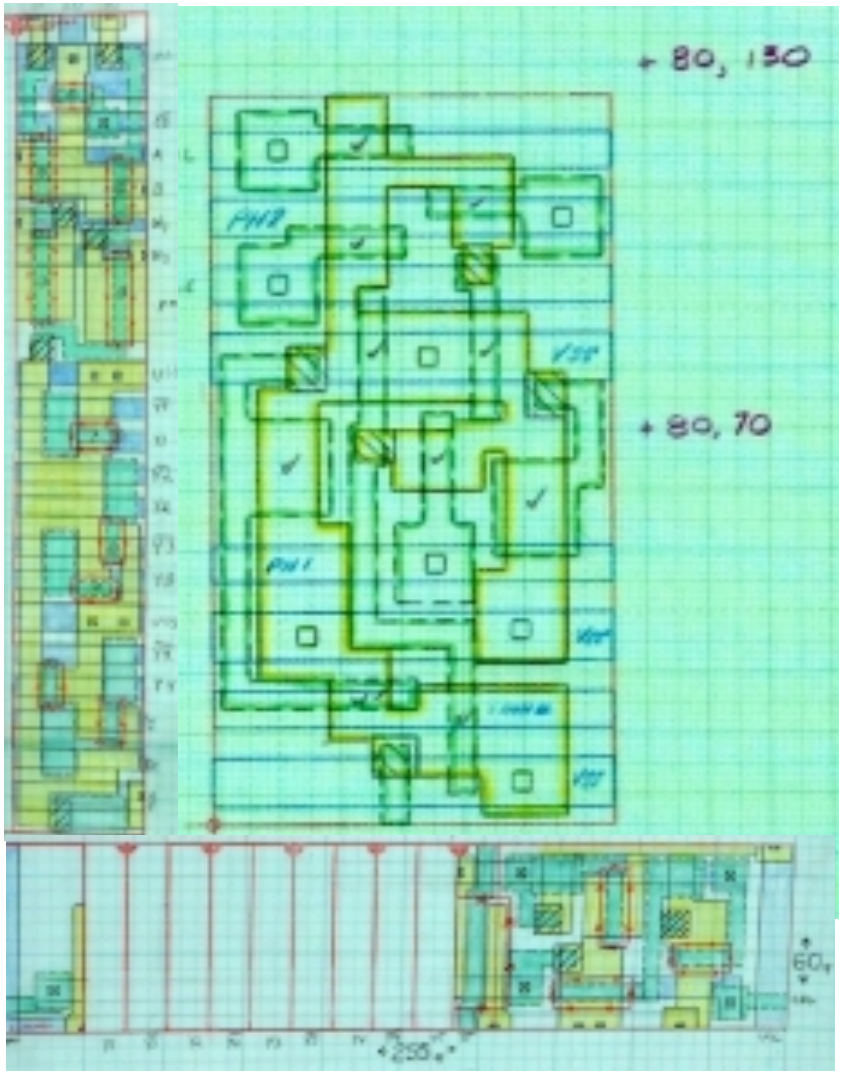


Figure 3: Hand-drawn cell layout on Mylar

From “Volk, Stoll & Metrovich. Recollections of Early Chip Development at Intel, ITC, Q1’ 2001.

Um pouco de história

"The first masks were made by transferring the drawings on the Mylar to "rubylith." Rubylith is a two-layered material, which comes in huge sheets. The base layer is heavy transparent dimensionally stable Mylar. A thin film of deep red cellophane-like material covers the base layer. The first chips at Intel used a machine called a "Coordinatograph" to guide cutting of the ruby layer. The coordinates and lengths had to be measured and transferred by hand to the cutter."

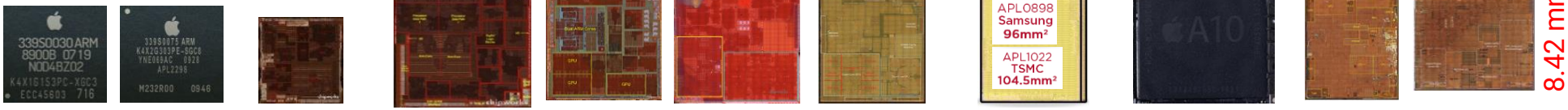
Figure 4: Technicians transferring layout to rubylith

From "Volk, Stoll & Metrovich. Recollections of Early Chip Development at Intel, ITC, Q1' 2001.

fernando.moraes@pucrs.br

Transistor size continues to shrink

2007	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018
iPhone	iPhone 3G	iPhone 4	iPhone 4S	iPhone 5	iPhone 5S	iPhone 6(+)	iPhone 6S(+)	**iPhone 7 (+)** (dual lens design)	iPhone 8,8+,X	iPhone XS,XR
APL0098	APL2298	A4	A5	A6	A7	A8	A9	A10	A11	A12
90nm	**65nm**	**45nm**		**32nm**	**28nm**	**20nm**	**14/16nm**	**16nm+**	**10nm**	**7nm**
XT:12X0 Shrink Add functions	*XT:14X0 Shrink* Add functions	*XT:17X0I Shrink* Add functions	Add functions	*XT:19X0I Shrink* Add functions	*NXT:1950Ai Shrink* Add functions	*NXT:1960Bi Shrink* Add functions	*NXT:1970Ci Shrink* Add functions	Add functions	*NXT:1980Di Shrink* Add functions	*NXT:1980Di Shrink* Add functions
72mm²	**71.8mm²**	**53.3mm²**	**122.2mm²**	**97mm²**	**102mm²**	**87.4mm²**	**96-104mm²**	**125mm²**	**89mm²**	**83mm²**
16 bit 1 Core 1 GPU 412 MHz	32 bit 1 Core 1 GPU 412 MHz	32 bit 1 Core 1 GPU 0.8 GHz	32 bit 2 Core 2 GPU 0.8 GHz	32 bit 2 Core 3 GPU 1.2 GHz	64 bit 2 Core 4 GPU 1.3 GHz	64 bit 2 Core 4 GPU 1.4 GHz	64 bit 2 Core 6 GPU 1.85 GHz L14&N16FF	64 bit 4 Core 6 GPU 2.37 GHz **N16FF+ InFO** wafer-level pckg	4.3B transistors 2 +4 Core CPU (4 + 1)GPU + NPU 2.4 GHz **N10FF**	6.9B transistors 2 +4 Core CPU (6)GPU +8 NPU 2.5 GHz **N7FF**



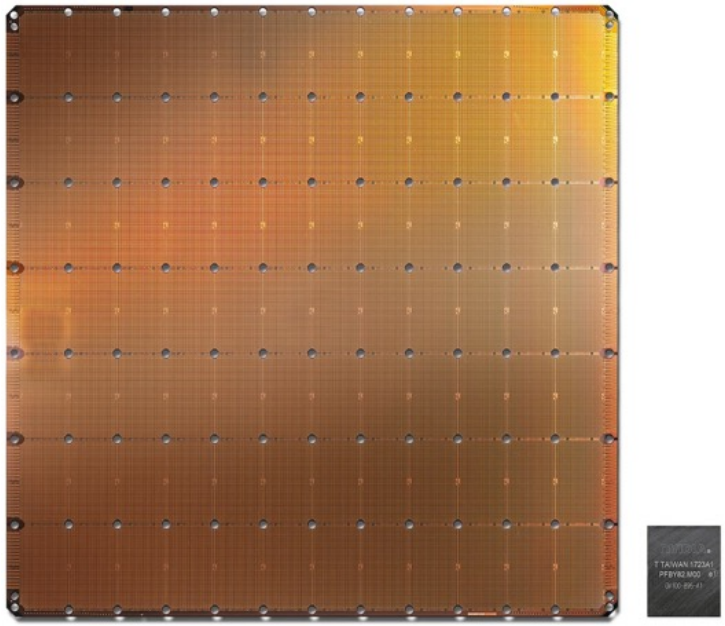
Source: ASML Market Research

TWINSCAN NXT:1980Di
193-nm Step and Scan (Resolution: ≤ 38 nm)

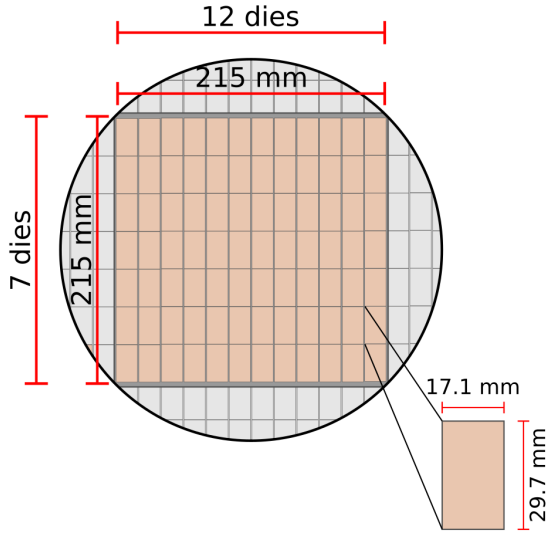
https://en.wikichip.org/wiki/apple/ax/a12

https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&ved=2ahUKEwjA6o_AxP_oAhWNGLkGHaeID5kQFjABegQIAhAB&url=https%3A%2F%2Fwww.asml.com%2F-%2Fmedia%2Fasml%2Ffiles%2Finvestors%2Fpast-events-and-presentations%2Fasml_20190319_2019-03-20_baml_taiwan_mar_2019_v1_final.pdf&usg=AOvVaw3rEbc4uaCfGlHnU6z4CY5H

Cerebras – https://www.cerebras.net



- TSMC 16nm, 84 dies
- The WSE (Wafer Scale Engine) is 215 mm by 215 mm



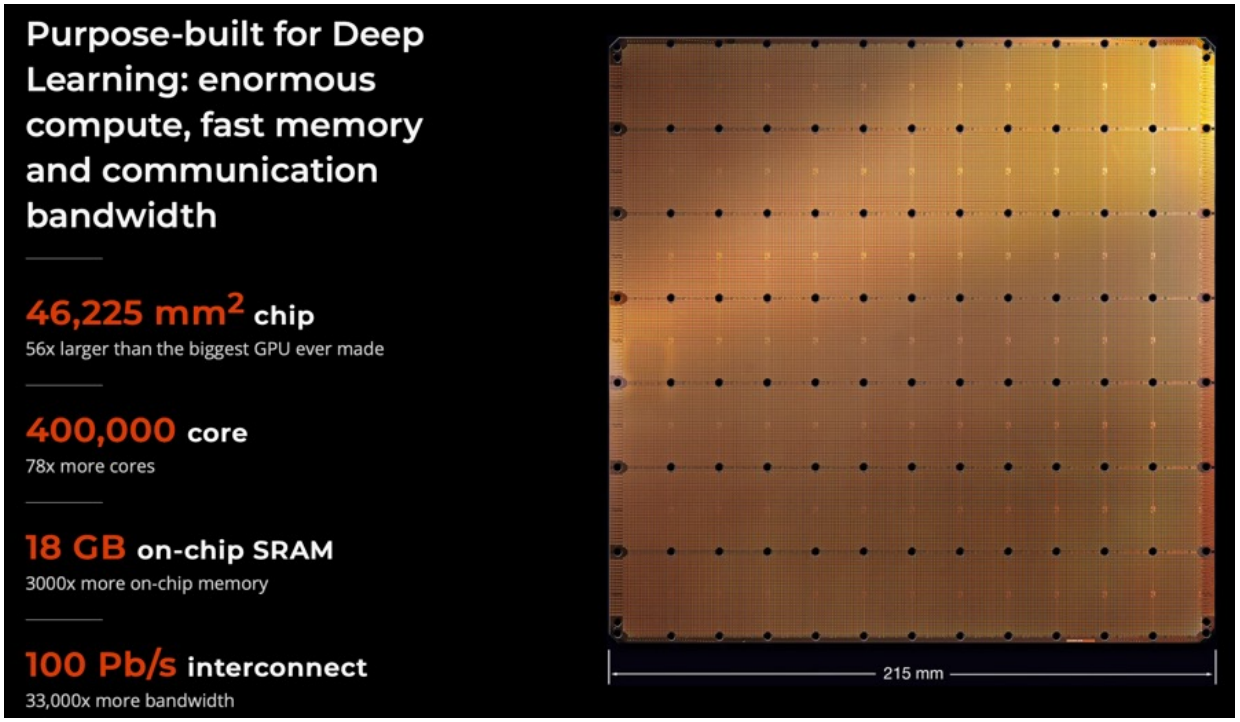
CS-1 is powered by the Cerebras Wafer Scale Engine - the largest chip ever built

56x the size of the largest Graphics Processing Unit

The Cerebras Wafer Scale Engine is 46,225 mm^2 with 1.2 Trillion transistors and 400,000 AI-optimized cores.

By comparison, the largest Graphics Processing Unit is 815 mm^2 and has 21.1 Billion transistors.

Consumo de potência máxima: 20 kW



https://fuse.wikichip.org/news/3010/a-look-at-cerebras-wafer-scale-engine-half-square-foot-silicon-chip

Sumário

O início e o estado-da-arte

Introdução

Métodos de Projeto

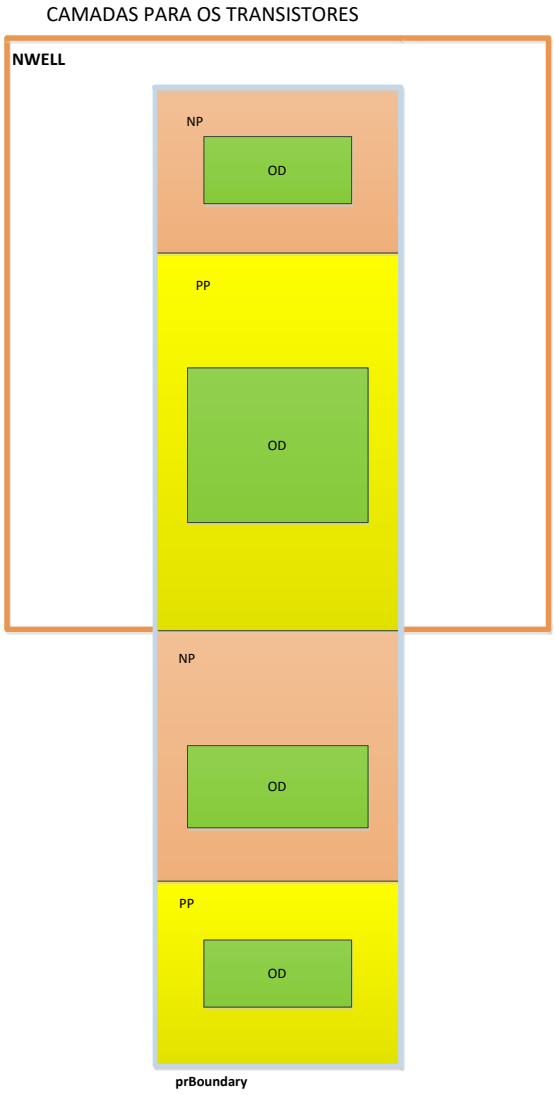
- **Full Custom**
- **Standard Cells**
- **Geração Automática**
- **Pré-Difundidos**
- **Componentes Programáveis**
  - **Passado – PLAs**
  - **Atualidade - FPGAs**

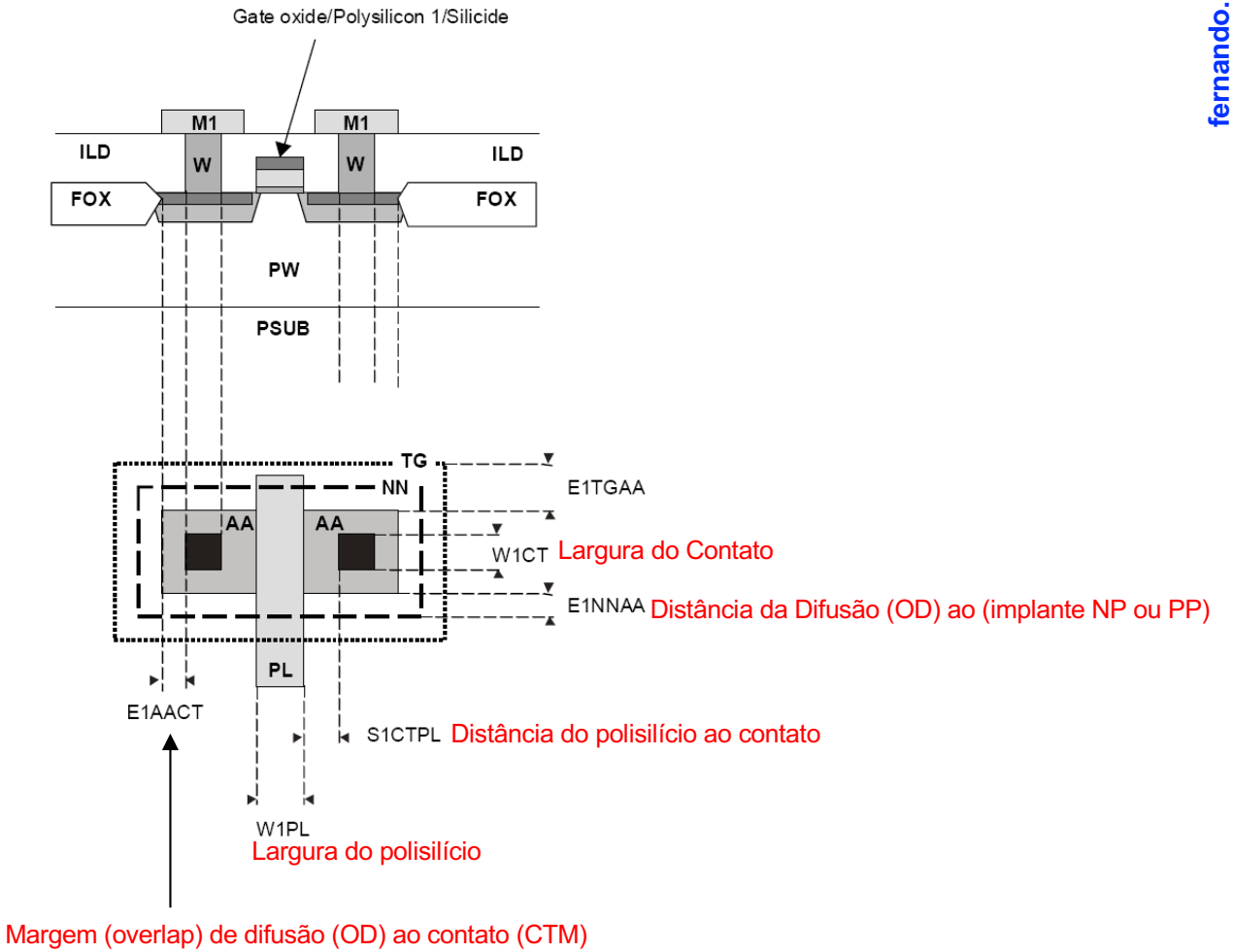
Em: 45 anos

De: 10.000
para: 21.000.000.000
transistores

Layout

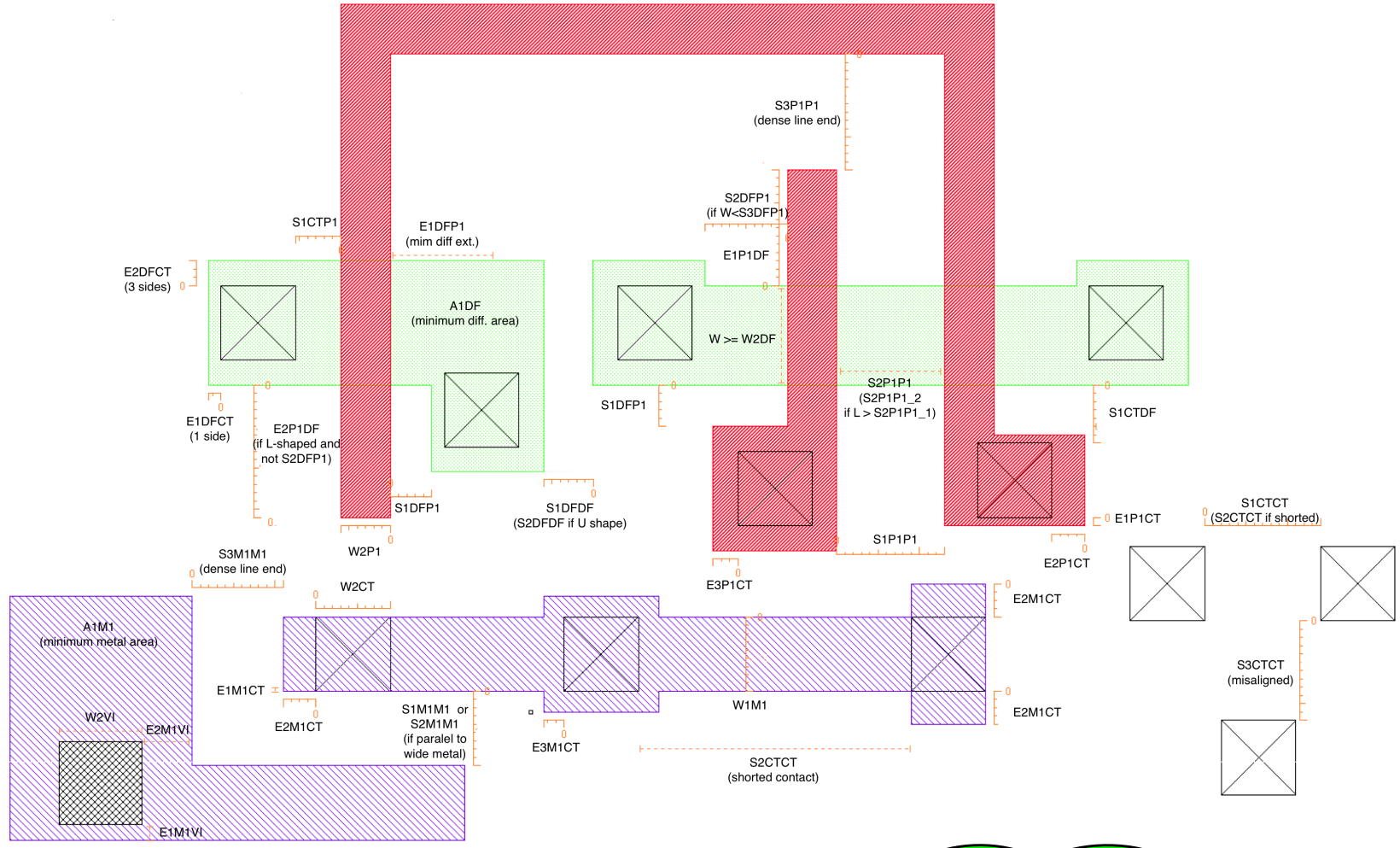
Refere-se às geometrias dos transistores e conexões que compõem o circuito





fernando.moraes@pucrs.br

Layout – regras de desenho e layout



E1INDF -> NPLUS enclosure of NDIFF
E1IPDF -> PPLUS enclusure of PDIFF
E1WNDP -> NWELL enclosure of PDIFF
S1DNWN -> NDIFF spacing to NWELL

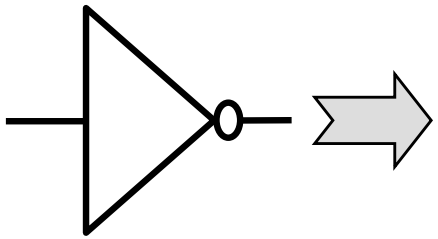
Dá trabalho fazer o layout de um circuito integrado!

Fonte: Adriel Ziesemer (Astran)

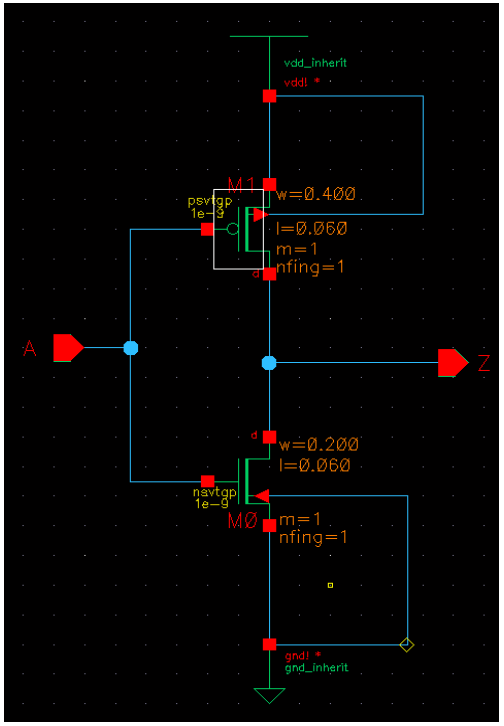
fernando.moraes@pucrs.br

Célula

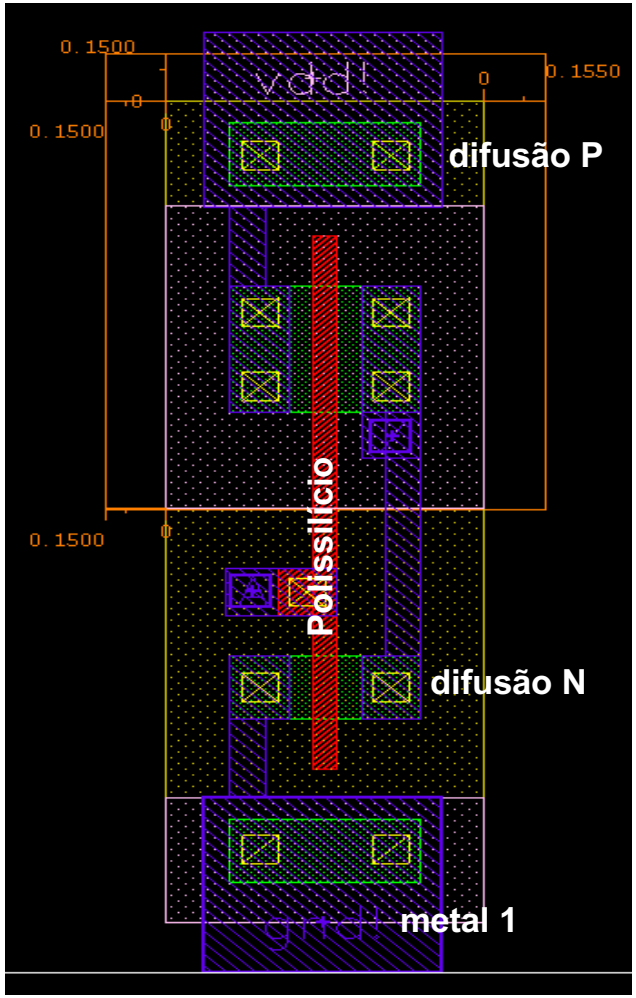
Refere-se ao layout de uma parte do circuito contendo transistores e conexões que implementam alguma função lógica básica (e.g., portas lógicas, multiplexadores, latches, flip-flops etc)



Símbolo lógico



Esquemático no nível de transistores



Layout

fernando.moraes@pucrs.br

Métodos de projeto

Como implementar um sistema eletrônico?

µprocessador
µcontrolador

- Chip fabricado e encapsulado ou projeto validado que pode ser "embarcado"
- Funcionalidade definida por programação

FPGA, CPLD, PLA, PAL, PROM, EPROM, EEPROM

- Chip fabricado e encapsulado
- Programação por fusíveis, transistores especiais, flash ou SRAM

pré-difundidos (gate array/sea-of-gates)

- Faltam algumas ou todas etapas de fabricação
- Programação = definição das máscaras de metal e contato

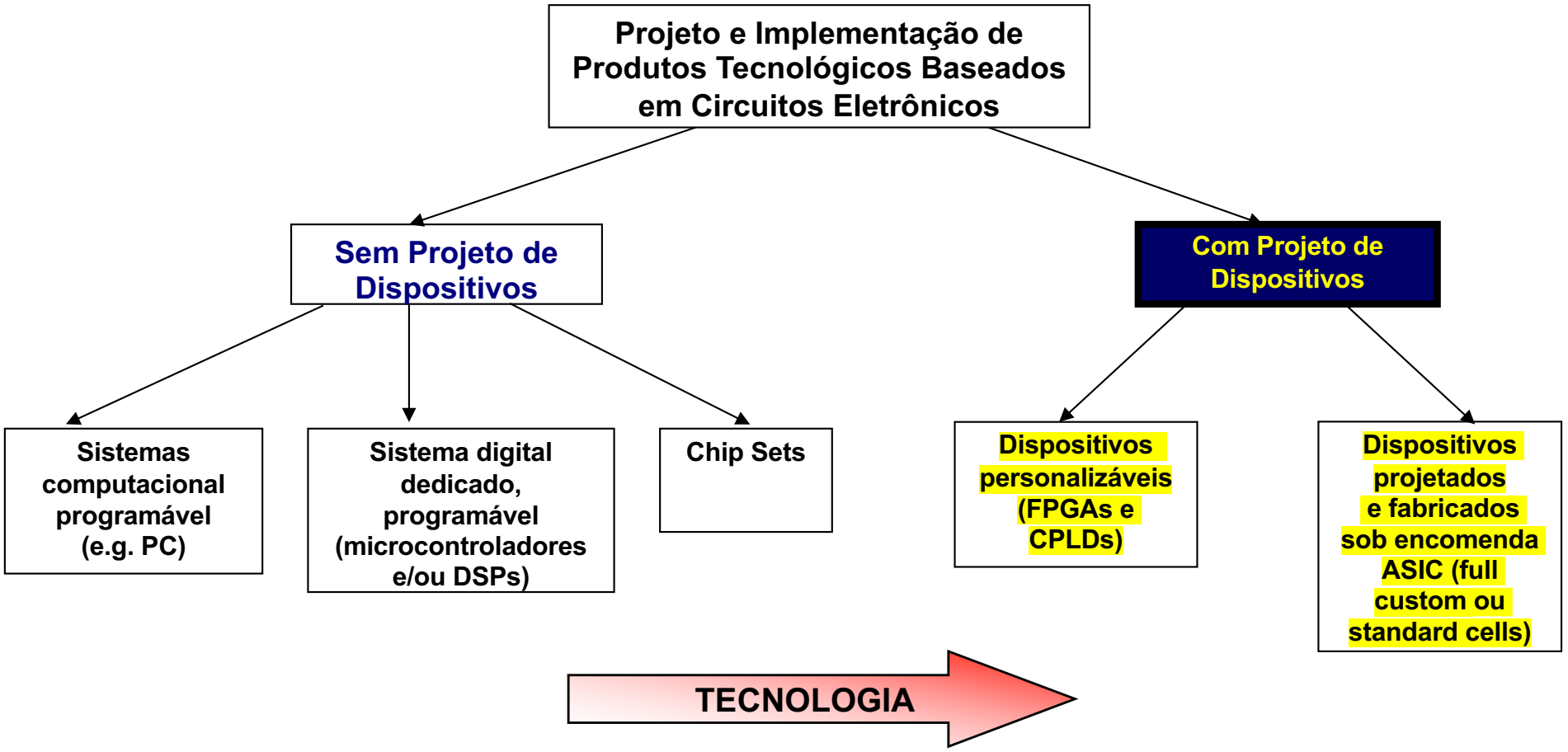
Full Custom/Standard-Cells (ASIC)

Menor custo para baixo volume

Menor custo para alto volume

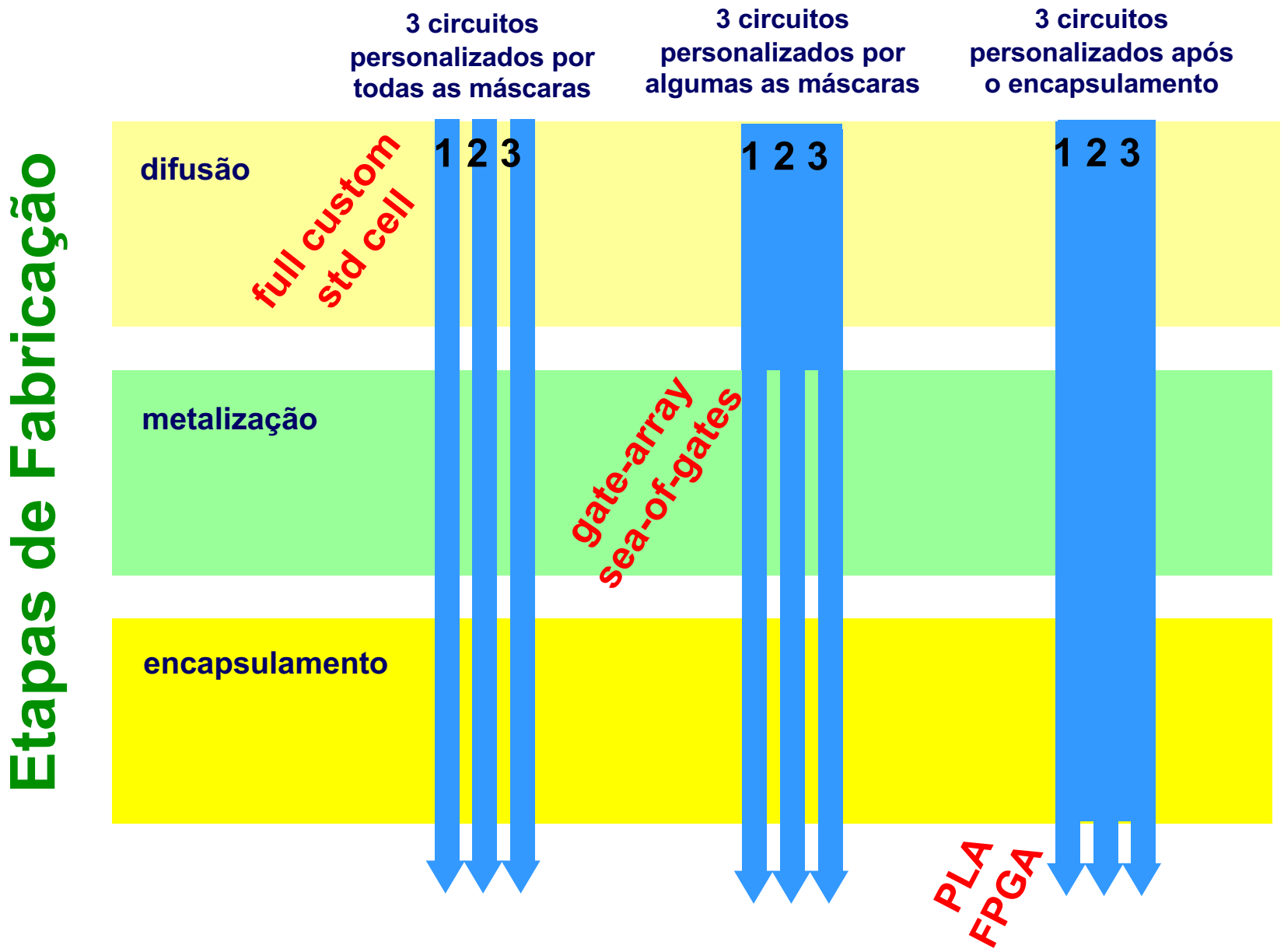
++ Proteção à Propriedade Intelectual
++ Densidade lógica
++ Desempenho
-- Tempo de projeto
-- Time to Market

Métodos de projeto



fernando.moraes@pucrs.br

Métodos de projeto com projeto de dispositivos



fernando.moraes@pucrs.br

Sumário

O início e o estado-da-arte

Introdução

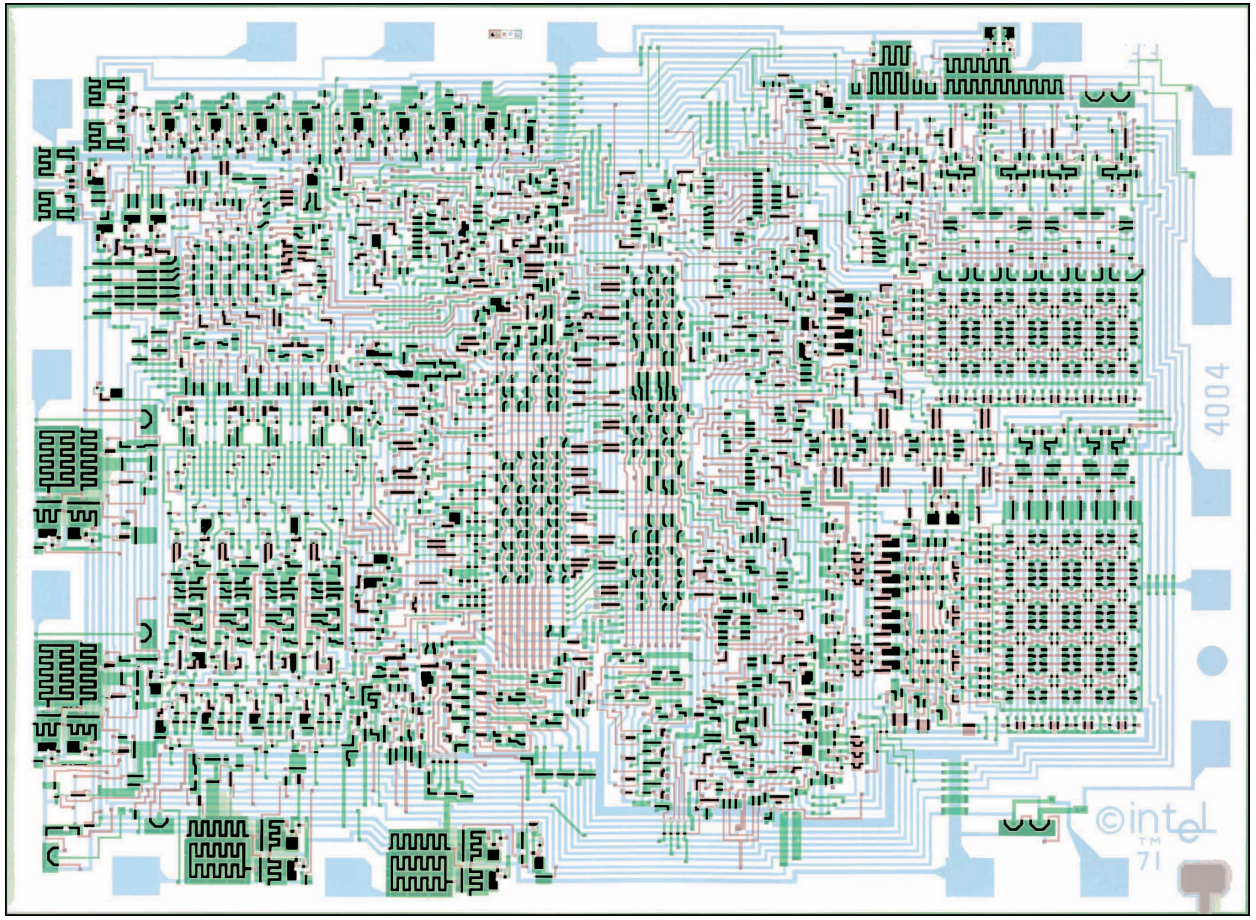
Métodos de Projeto

- **Full Custom**
- **Standard Cells**
- **Geração Automática**
- **Pré-Difundidos**
- **Componentes Programáveis**
  - **Passado – PLAs**
  - **Atualidade - FPGAs**

Projeto Full Custom

- O projetista define cada elemento (transistor, conexão)
- Pouca ou nenhuma regularidade no layout
- Uso de editor de layout
- Edição de layout é muito passível de erros

4004: 2.300 transistors (1971)

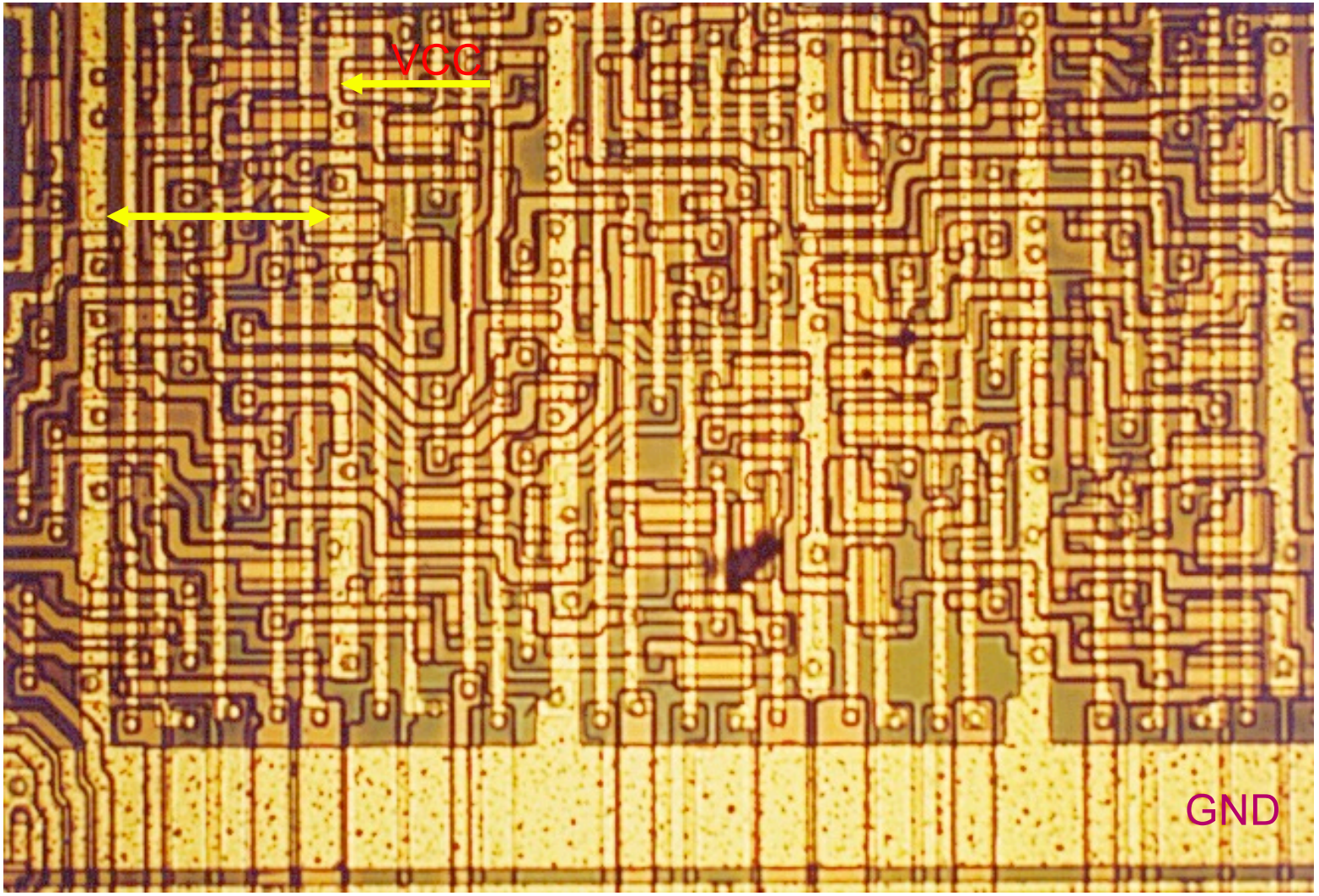


http://www.4004.com

Full Custom

Detalhe da parte controle do TMS7000 (Microcontrolador da Texas Instrument)

Estrutura em bandas (linhas de células)

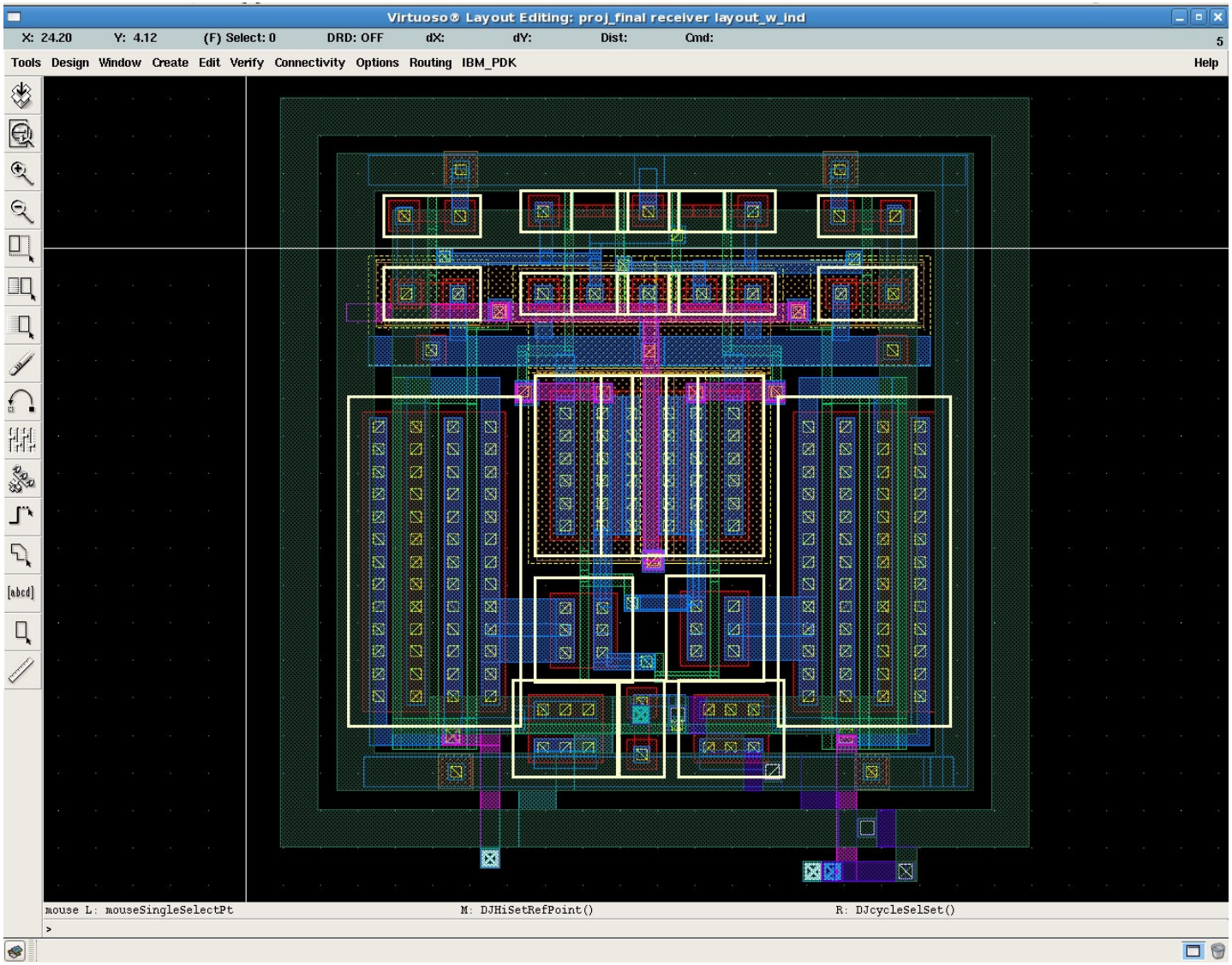


fernando.moraes@pucrs.br

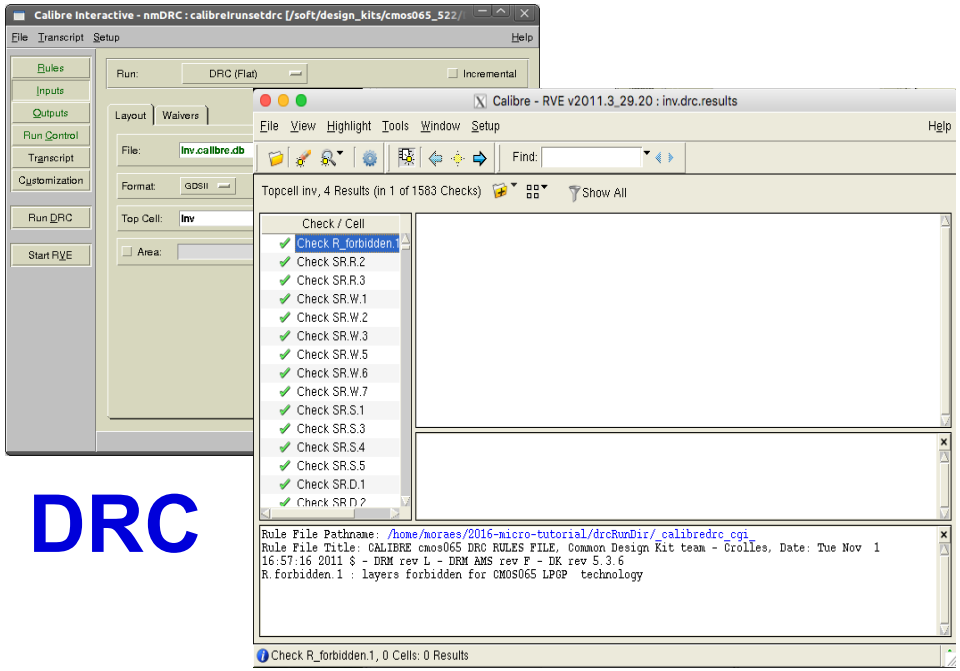
Full Custom

Edição de Layout

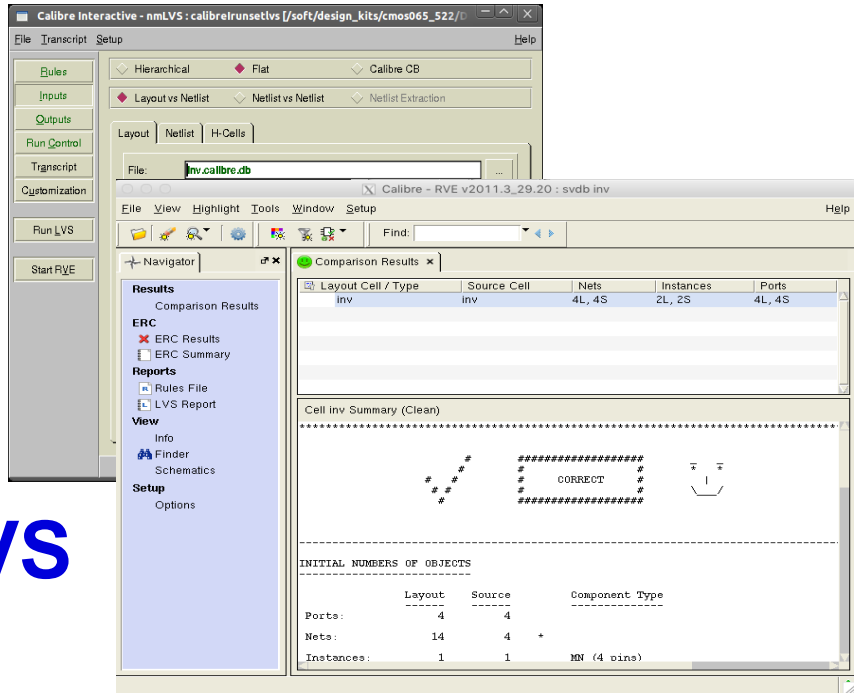
Editor de layout Virtuoso (Cadence)



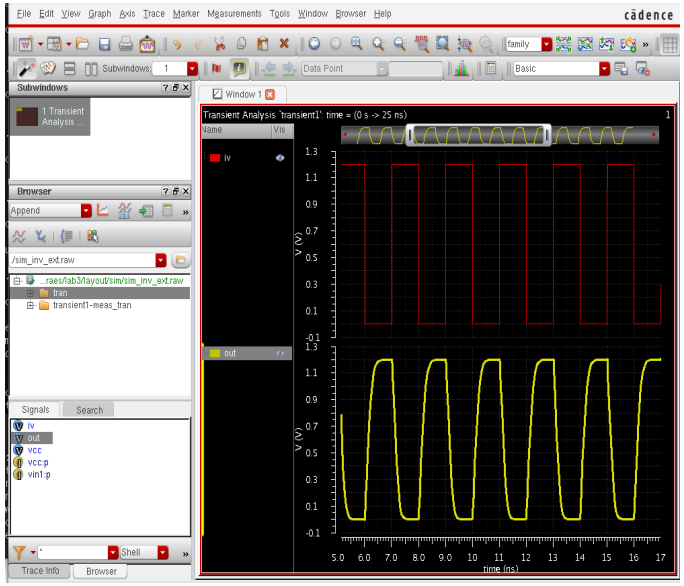
Full Custom – ferramentas de verificação



DRC



LVS



Extração elétrica e simulação

Full Custom

fernando.moraes@pucrs.br

Necessidade de Maior Automatização

- Número de componentes muito grande para desenhar um a um
- Regras de projeto se tornam cada vez mais complexas

Uso do método hoje

- aplicado ao projeto de biblioteca de células (também muito automatizado hoje)
- partes específicas de circuitos de alto desempenho

Sumário

O início e o estado-da-arte

Introdução

Métodos de Projeto

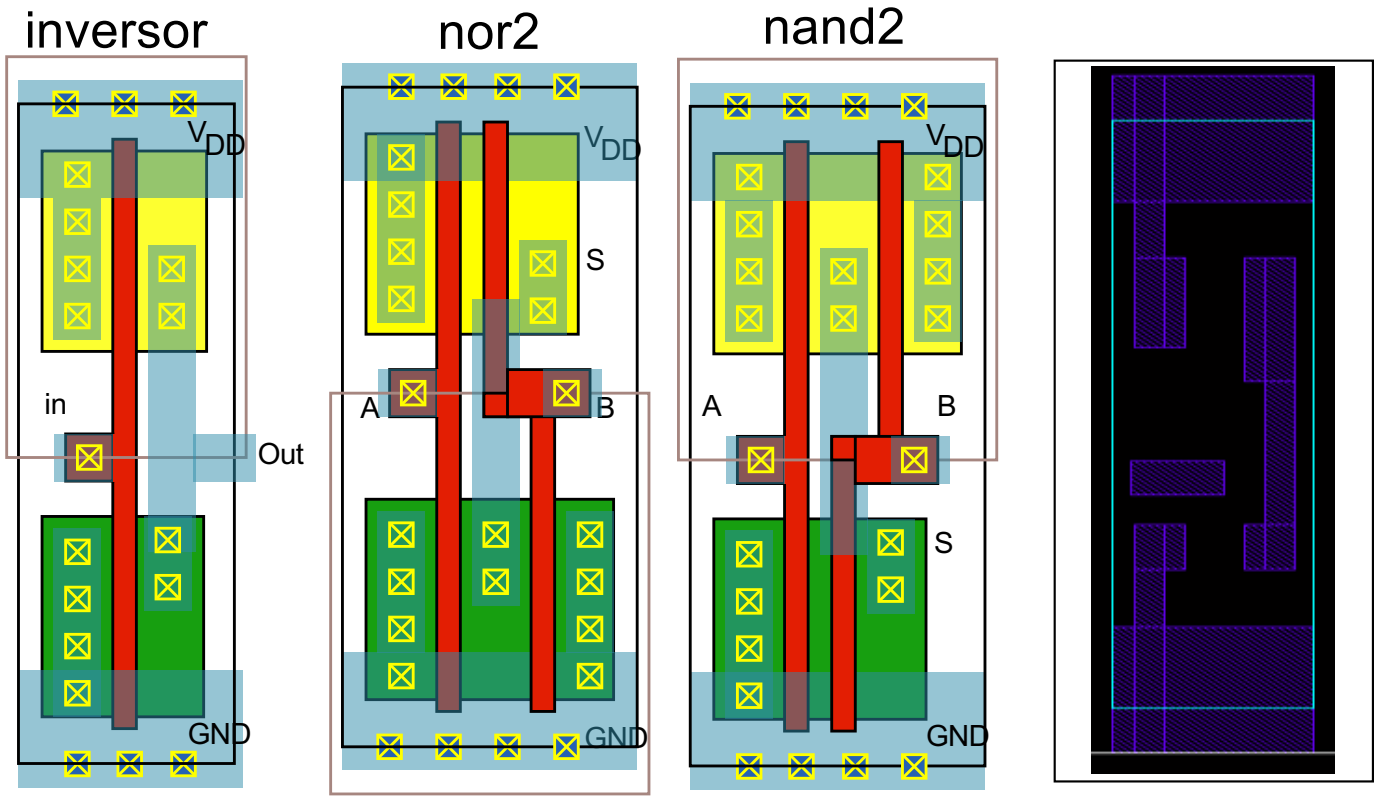
- **Full Custom**
- **Standard Cells**
- **Geração Automática**
- **Pré-Difundidos**
- **Componentes Programáveis**
  - **Passado – PLAs**
  - **Atualidade - FPGAs**

Standard Cells

- **Reduz a complexidade para o projeto**

  → *full custom* só no nível de células

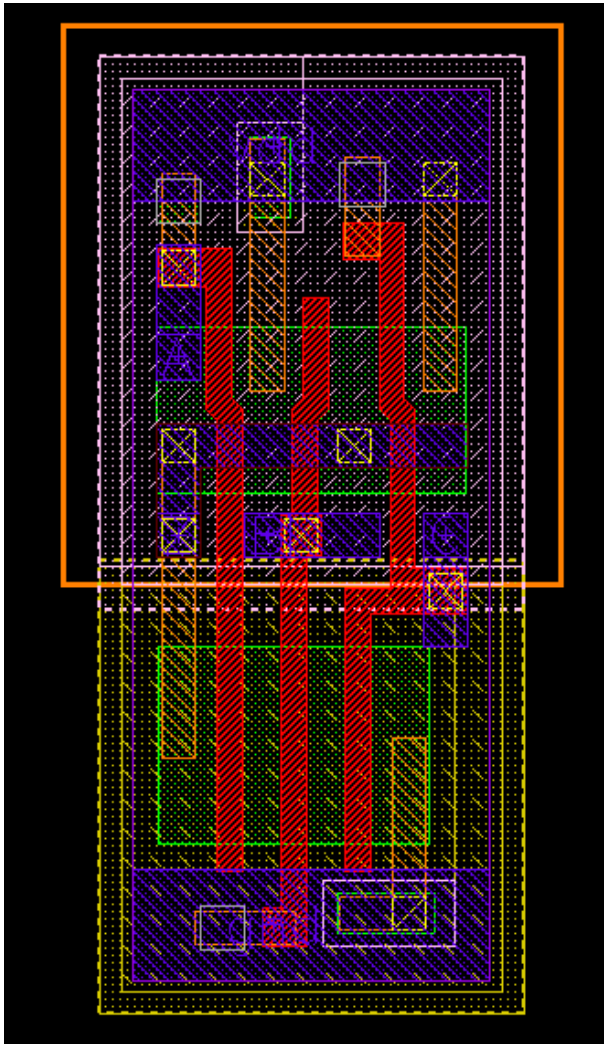
- **Layout de um circuito é montado usando células disponíveis em uma biblioteca de células**
- **Todas as camadas são fabricadas na *foundry***



"Vista"
LEF

Library
Exchange
Format

Standard Cells



3-input NAND cell
(from ST Microelectronics)

Biblioteca de células

- Diversos layouts para cada tipo de célula
- Cada layout prevê uma certa carga capacitiva máxima (*logic-effort*)
- Todas as células têm a mesma altura e são pré-caracterizadas
- Conexão da alimentação por justaposição

Path	1.2V - 125°C	1.6V - 40°C
$In1—t_{pLH}$	$0.073+7.98C+0.317T$	$0.020+2.73C+0.253T$
$In1—t_{pHL}$	$0.069+8.43C+0.364T$	$0.018+2.14C+0.292T$
$In2—t_{pLH}$	$0.101+7.97C+0.318T$	$0.026+2.38C+0.255T$
$In2—t_{pHL}$	$0.097+8.42C+0.325T$	$0.023+2.14C+0.269T$
$In3—t_{pLH}$	$0.120+8.00C+0.318T$	$0.031+2.37C+0.258T$
$In3—t_{pHL}$	$0.110+8.41C+0.280T$	$0.027+2.15C+0.223T$

C = Load capacitance T = input rise/fall time

Vista: LIB

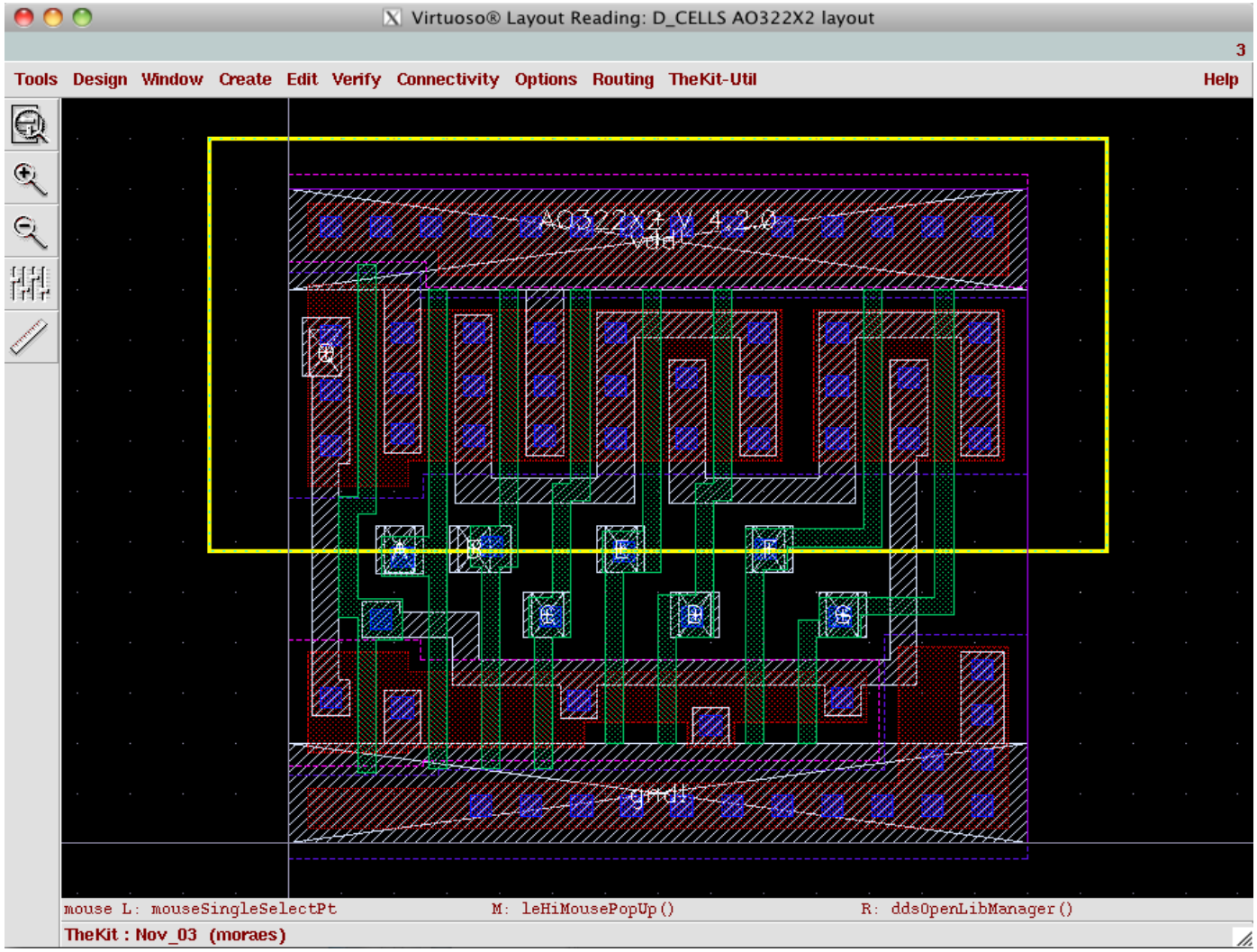
Standard Cells

HS65_GL_IVX2
HS65_GL_IVX4
HS65_GL_IVX7
HS65_GL_IVX9
HS65_GL_IVX13
HS65_GL_IVX18
HS65_GL_IVX22
HS65_GL_IVX27
HS65_GL_IVX31
HS65_GL_IVX35
HS65_GL_IVX40
HS65_GL_IVX44
HS65_GL_IVX49
HS65_GL_IVX53
HS65_GL_IVX62
HS65_GL_IVX71
HS65_GL_IVX106
HS65_GL_IVX142
HS65_GL_IVX213
HS65_GL_IVX284

HS65_GL_NAND2X2
HS65_GL_NAND2X4
HS65_GL_NAND2X5
HS65_GL_NAND2X7
HS65_GL_NAND2X11
HS65_GL_NAND2X14
HS65_GL_NAND2X21
HS65_GL_NAND2X29
HS65_GL_NAND2X43
HS65_GL_NAND2X57

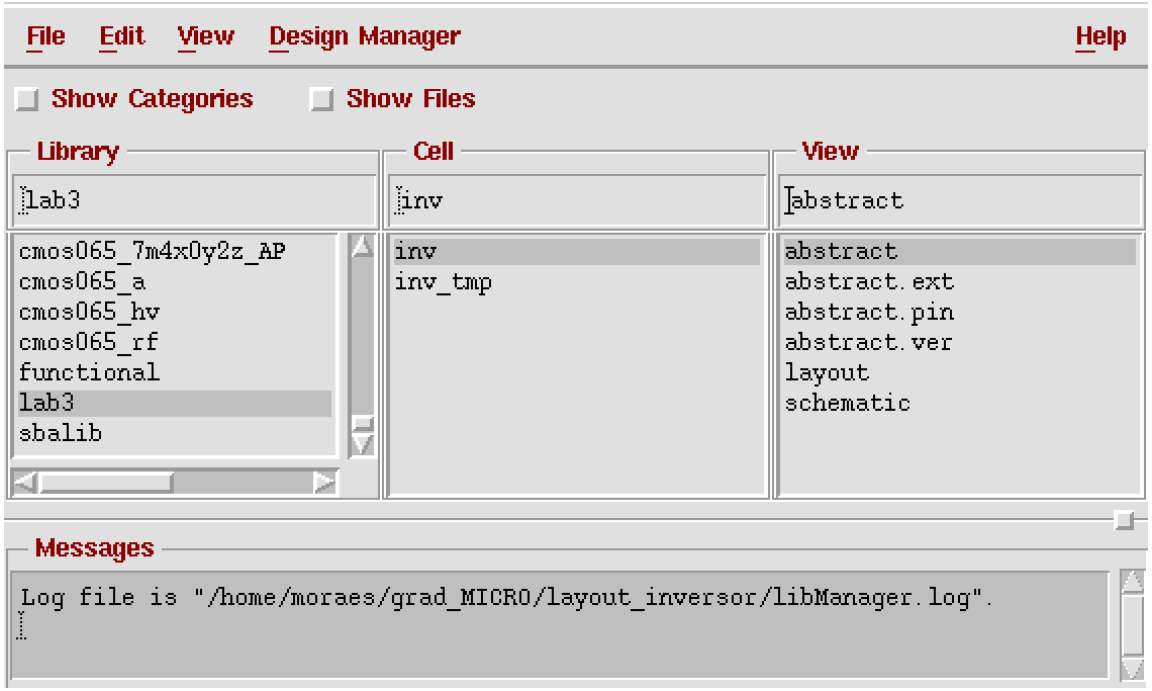
Diversos layouts para cada tipo de célula

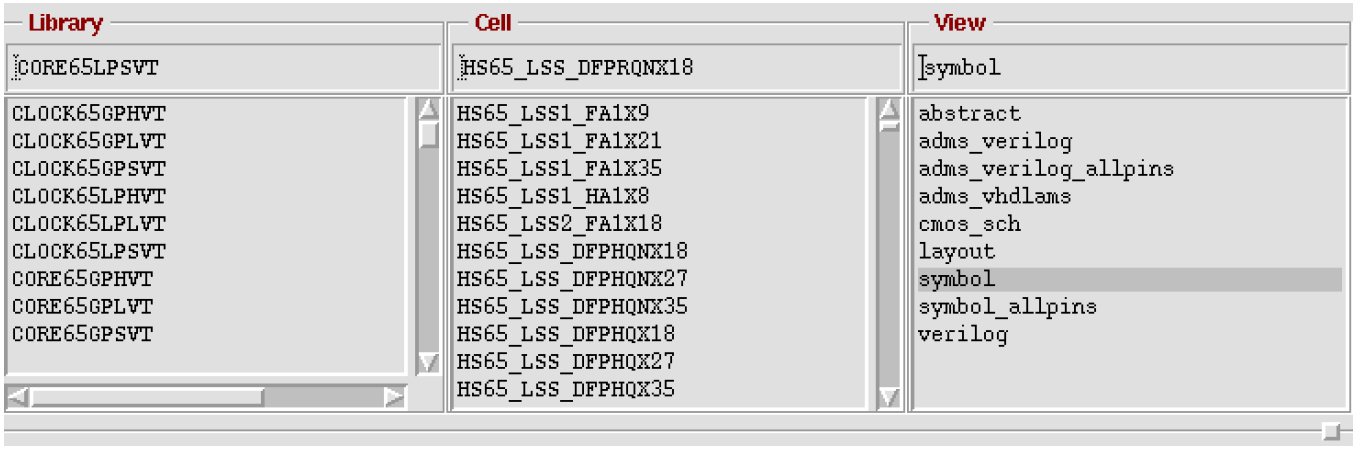
- diferentes *drive strengths*

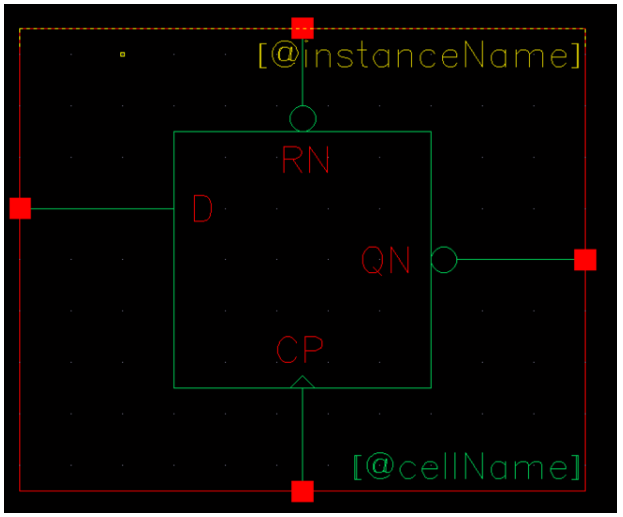


Standard Cells

- "Vistas" de uma célula
  - LEF (abstract)
  - LIB
  - layout
  - symbol
  - esquemático
  - verilog
  - ...







Standard Cells

PUCRS

fernando.moraes@pucrs.br

Automatização corresponde a montar o layout

- Particionamento
- Posicionamento
- Roteamento

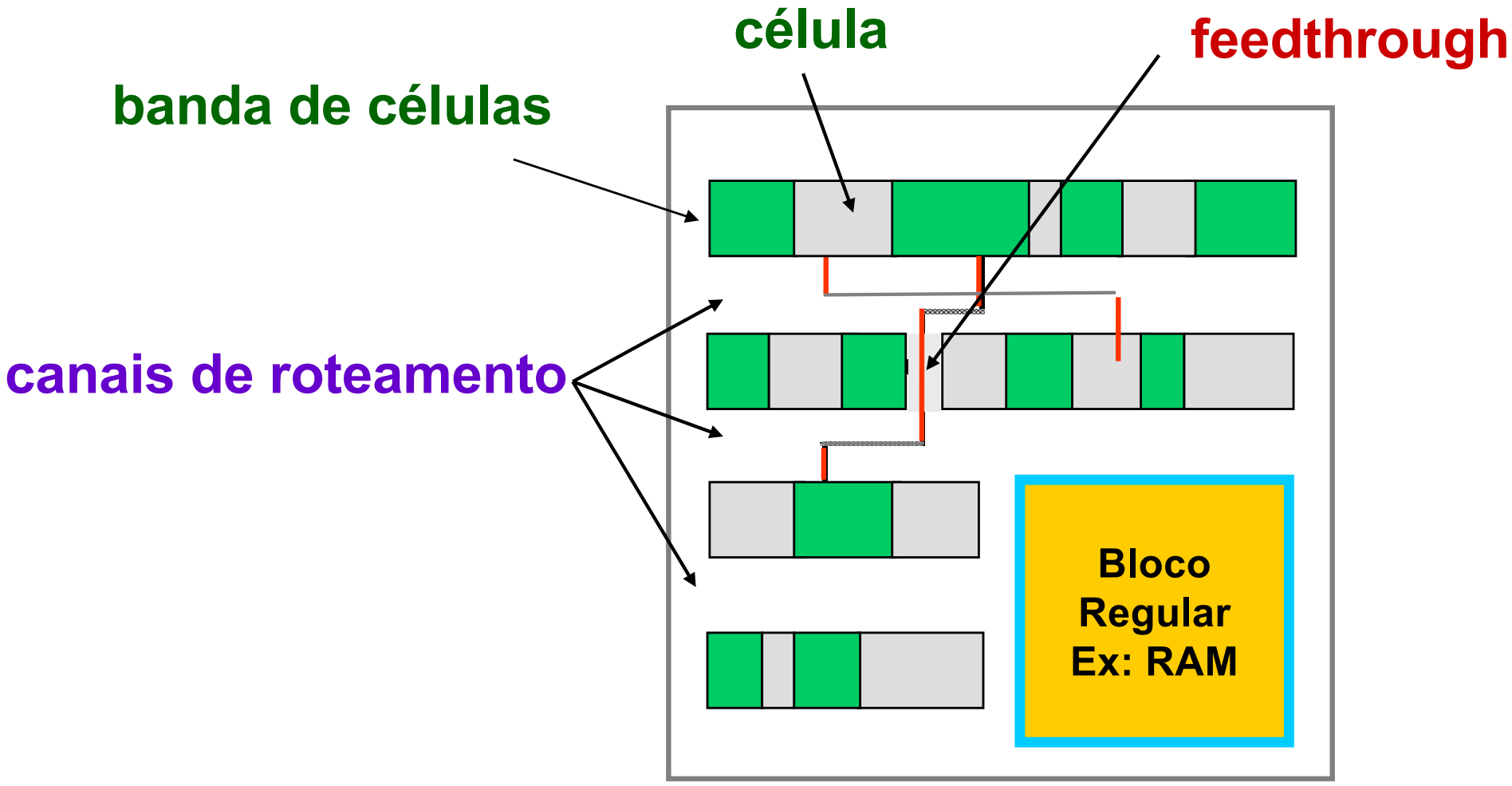
} Etapas da síntese física

Muito usado na indústria por permitir prever

- Área
- Potência
- Atraso

Mas esta previsibilidade começou a falhar à medida que as tecnologias CMOS se tornavam submicrônicas!

Standard Cells “antiga”

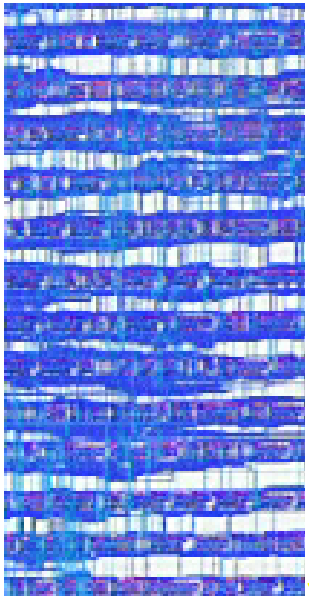


fernando.moraes@pucrs.br

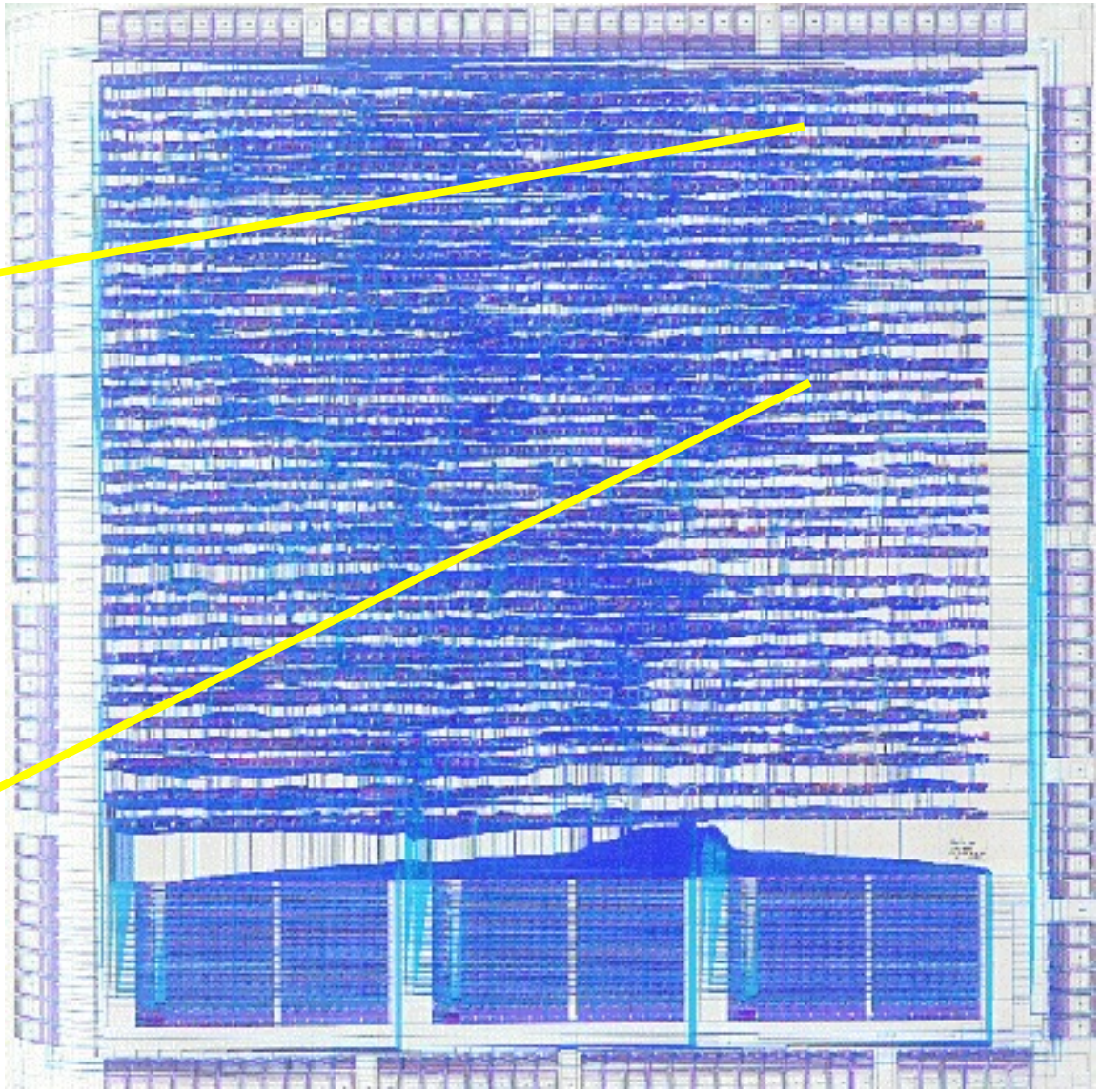
Standard Cells “antiga”

CPU HP
(1987, 80.000 trans.)

Ainda com canais de roteamento



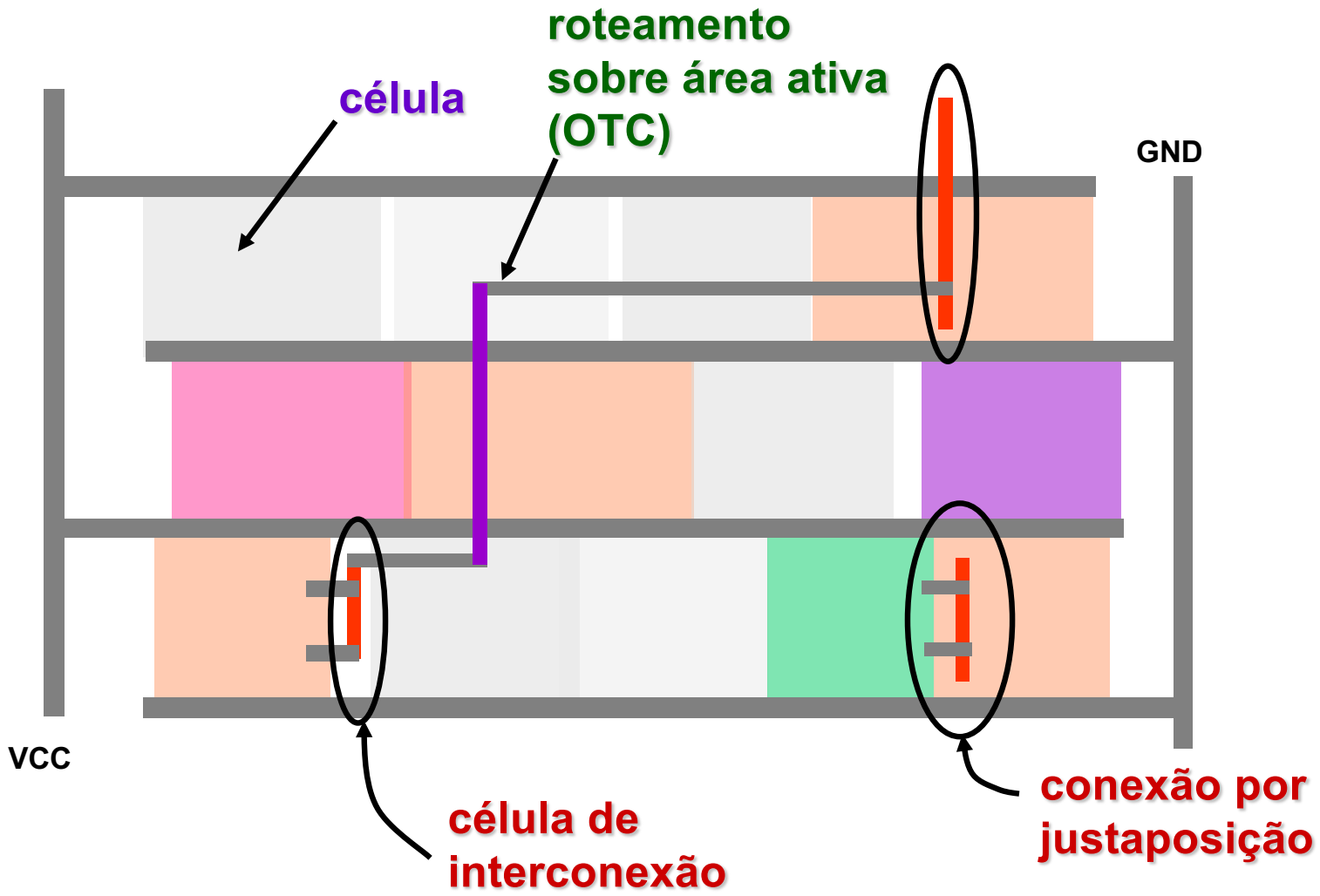
As zonas em azul escuro representam o roteamento em metal1 e metal2



Parte inferior: bloco regular Ex: RAM

Standard Cells atual

- Ausência de canais de roteamento (*over the cell routing*)
- Uso de múltiplas camadas de metal



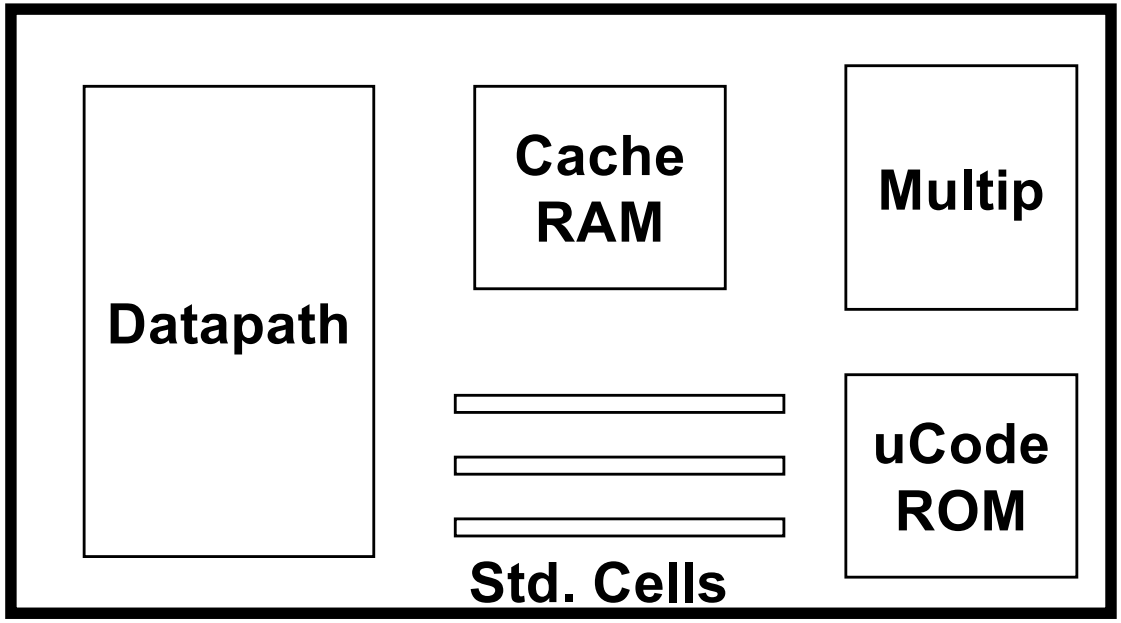
fernando.moraes@pucrs.br

General Cell Design

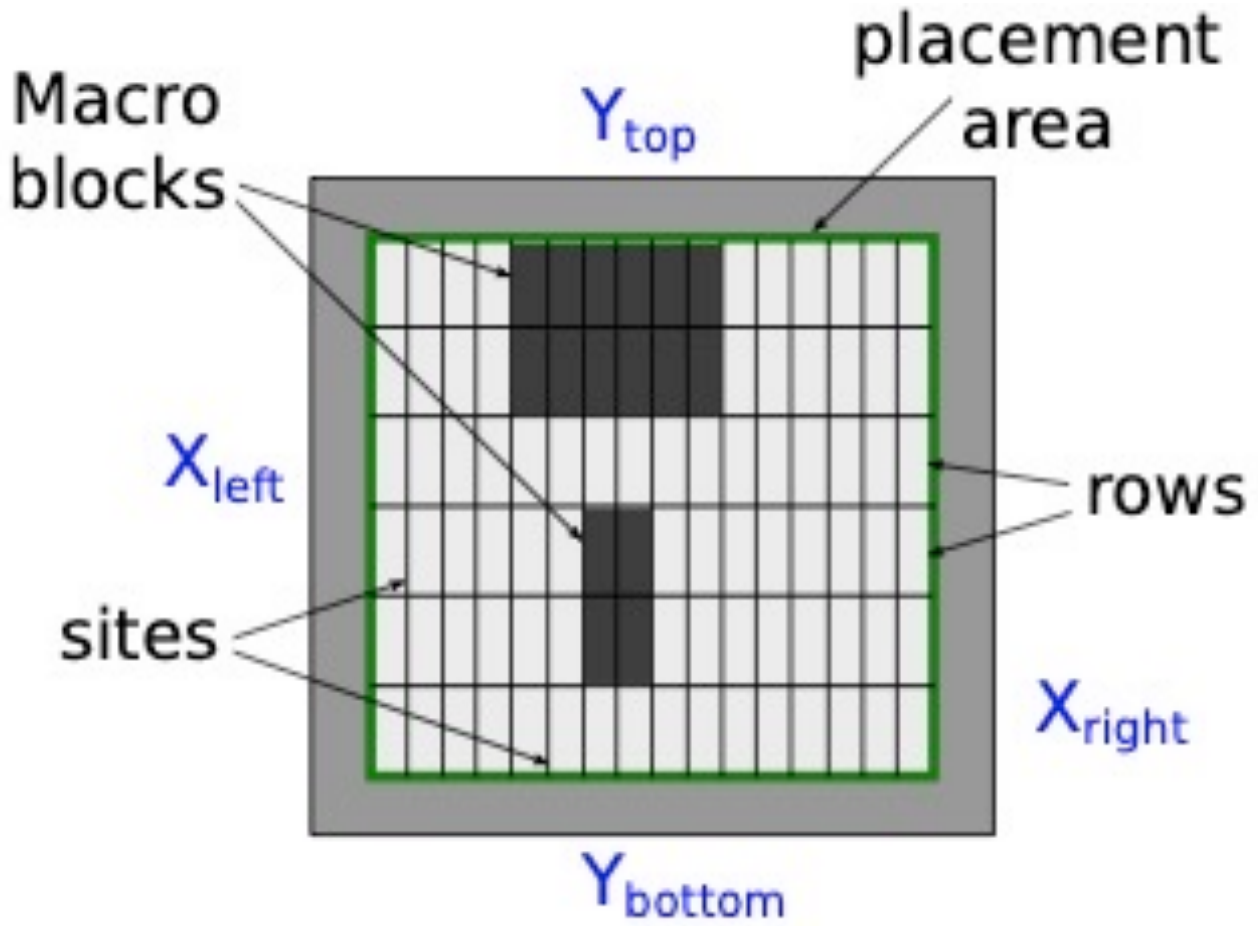
fernando.moraes@pucrs.br

Generalização de standard cells

- Células implementando portas lógicas são providas pelas empresas que fornecem a tecnologia (e.g. ARM, TSMC)
- Empresas também fornecem geradores de módulos regulares
- Utilizado em projetos de alta complexidade (e.g. CPUs, GPUs)
- Problema de CAD: posicionamento e roteamento de formas irregulares é muito difícil, por isso células possuem dimensões regulares

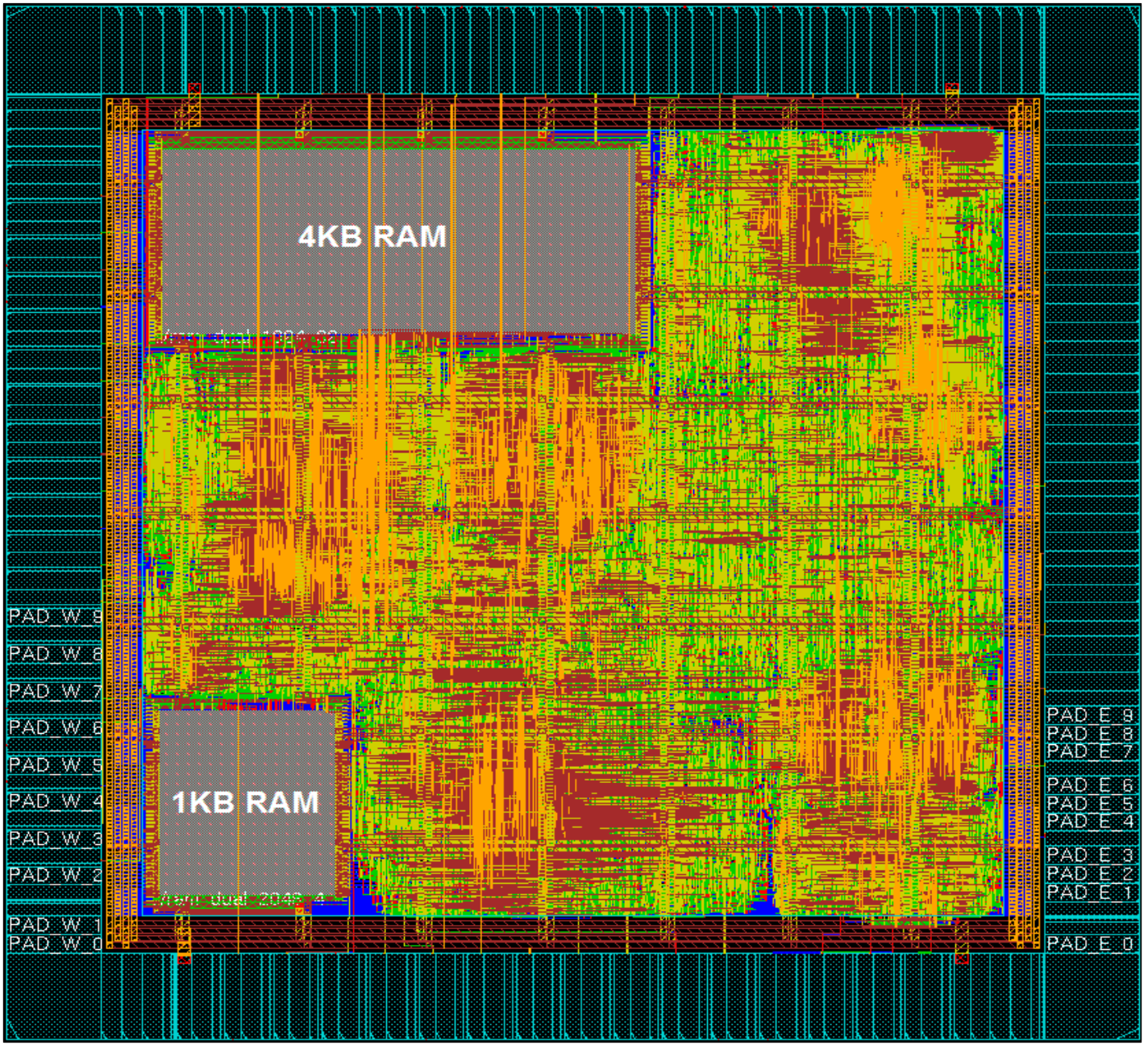


General Cell Design

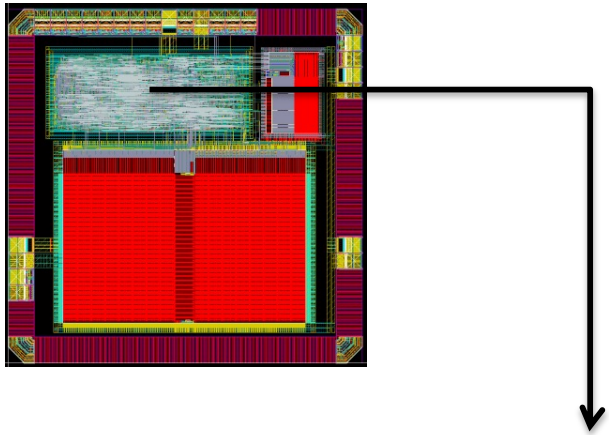


fernando.moraes@pucrs.br

General Cell Design - Exemplo

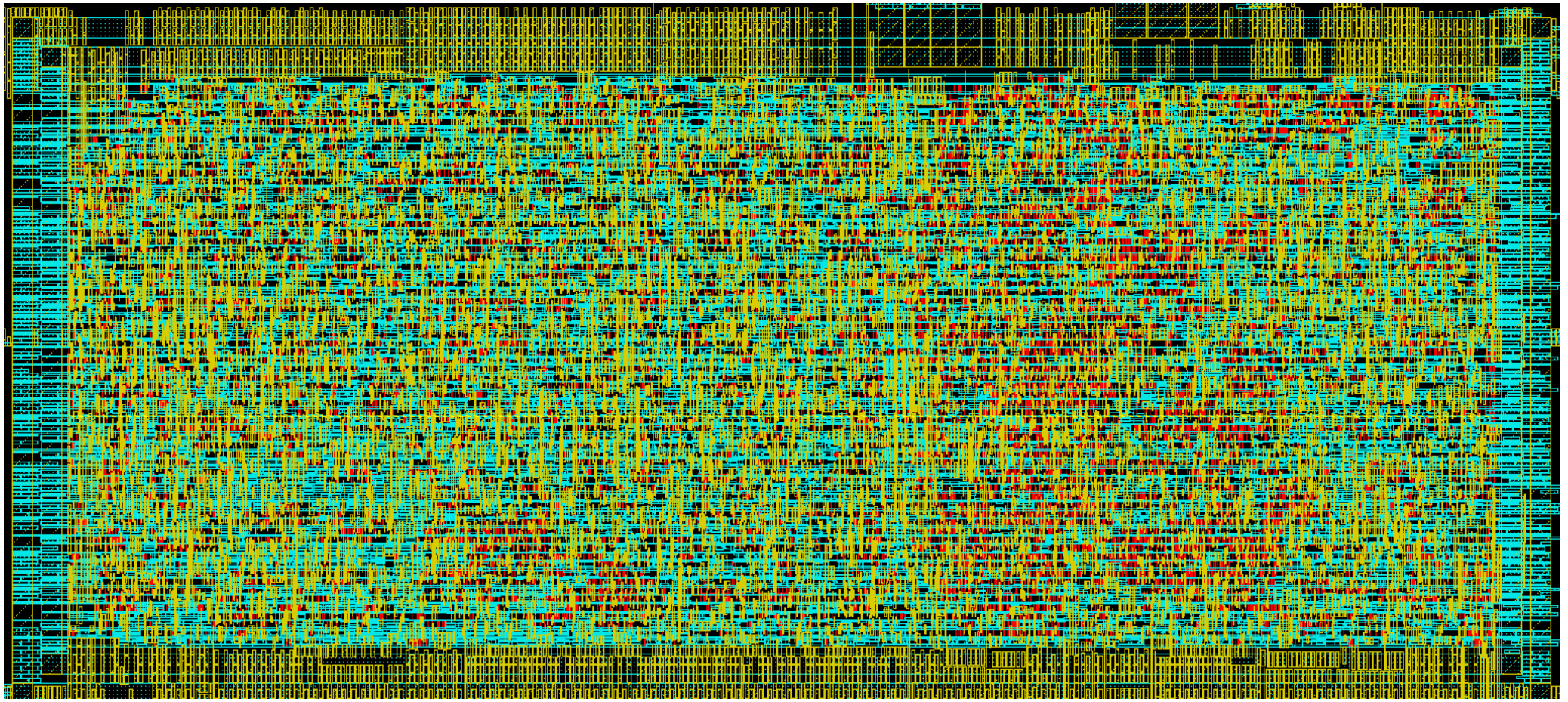


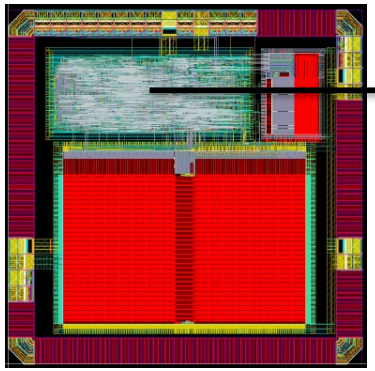
fernando.moraes@pucrs.br



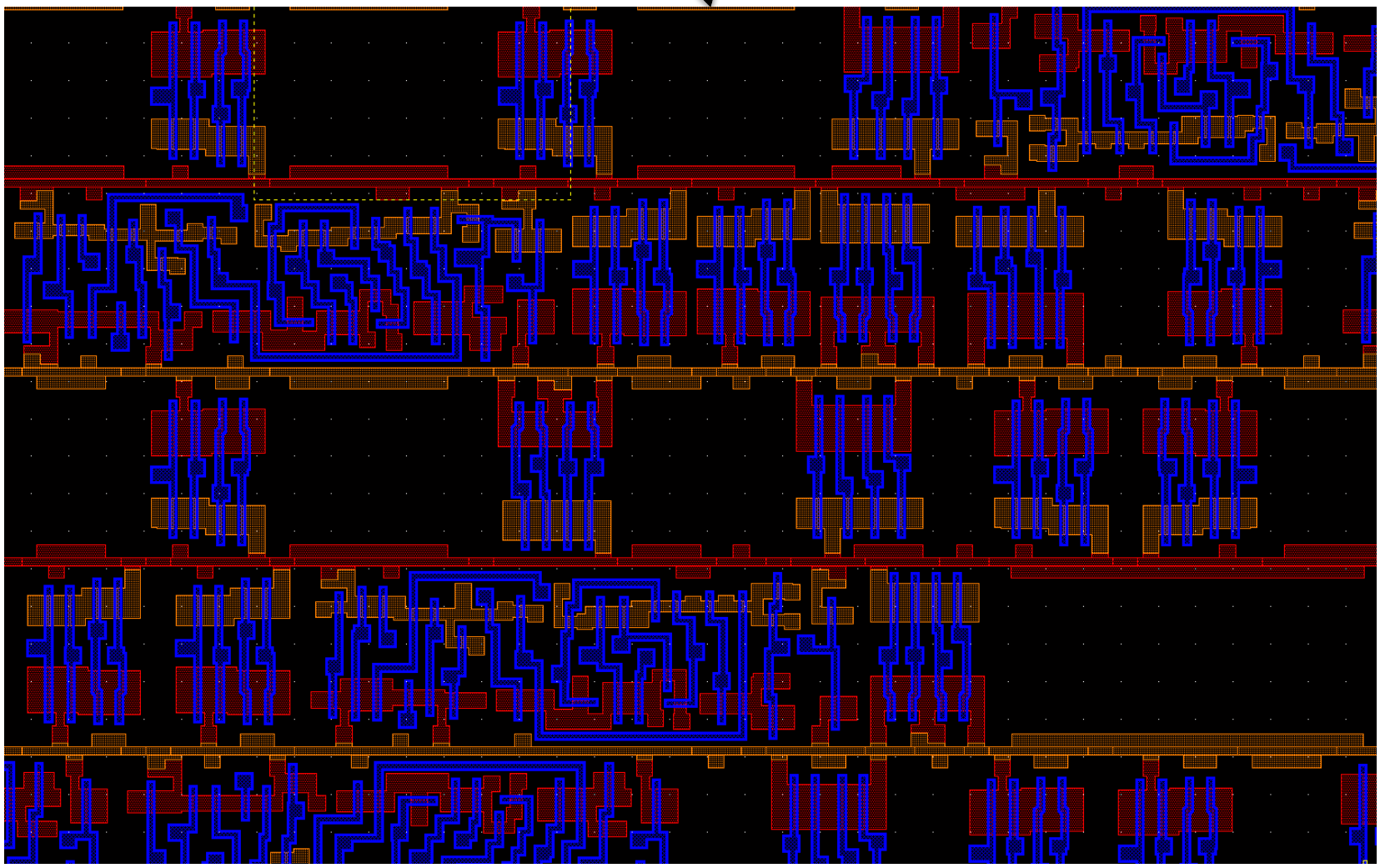
Standard-cells

- Células ocultas sob camadas de interconexão
- Roteamento sobre a célula

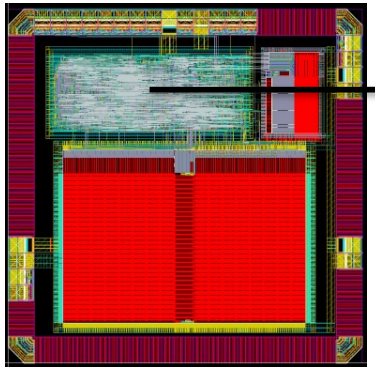




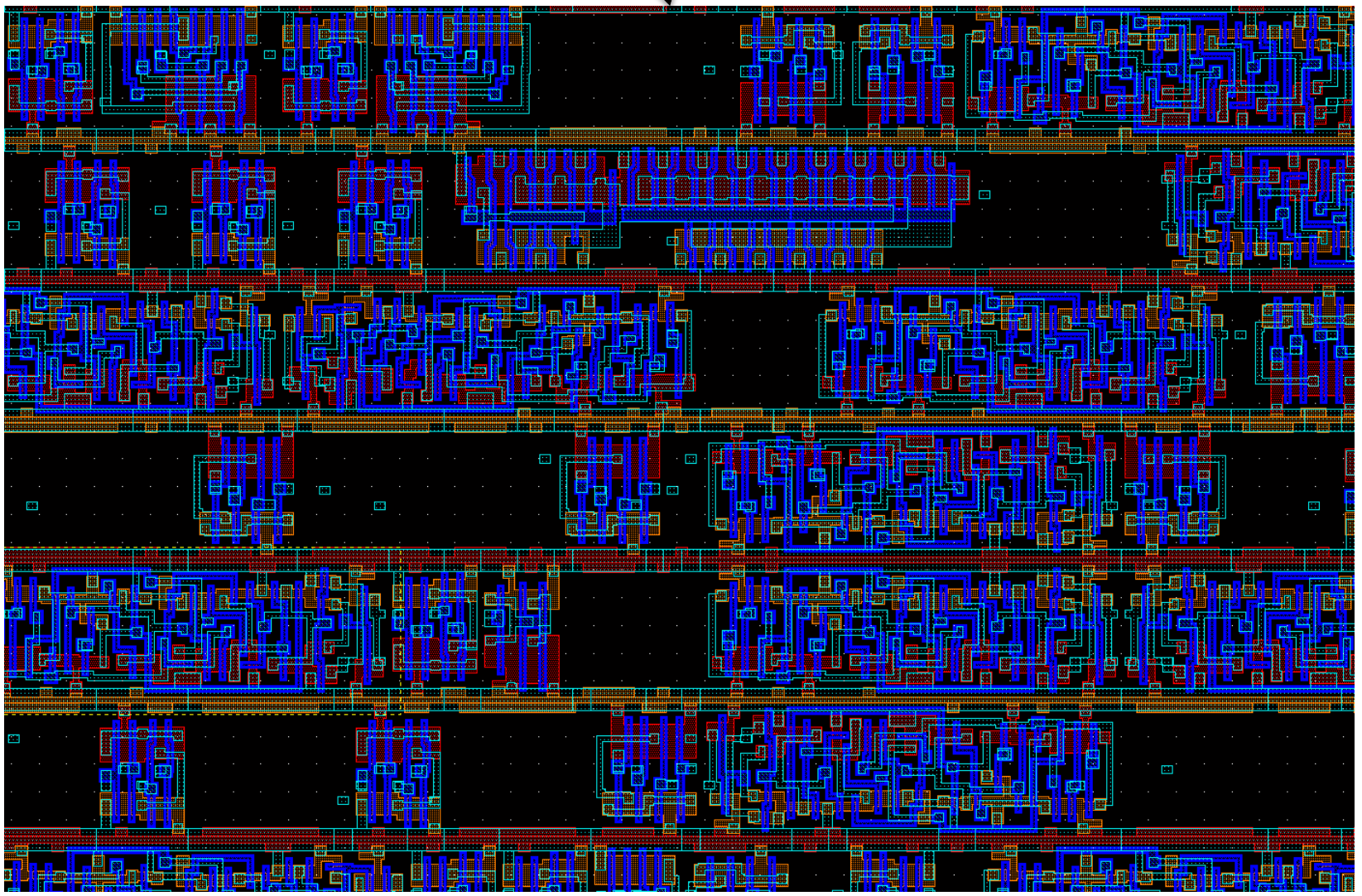
Standard-cells (só transistores)



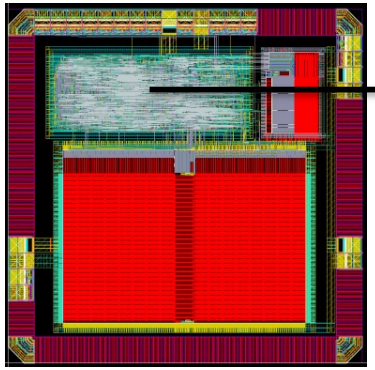
fernando.moraes@pucrs.br



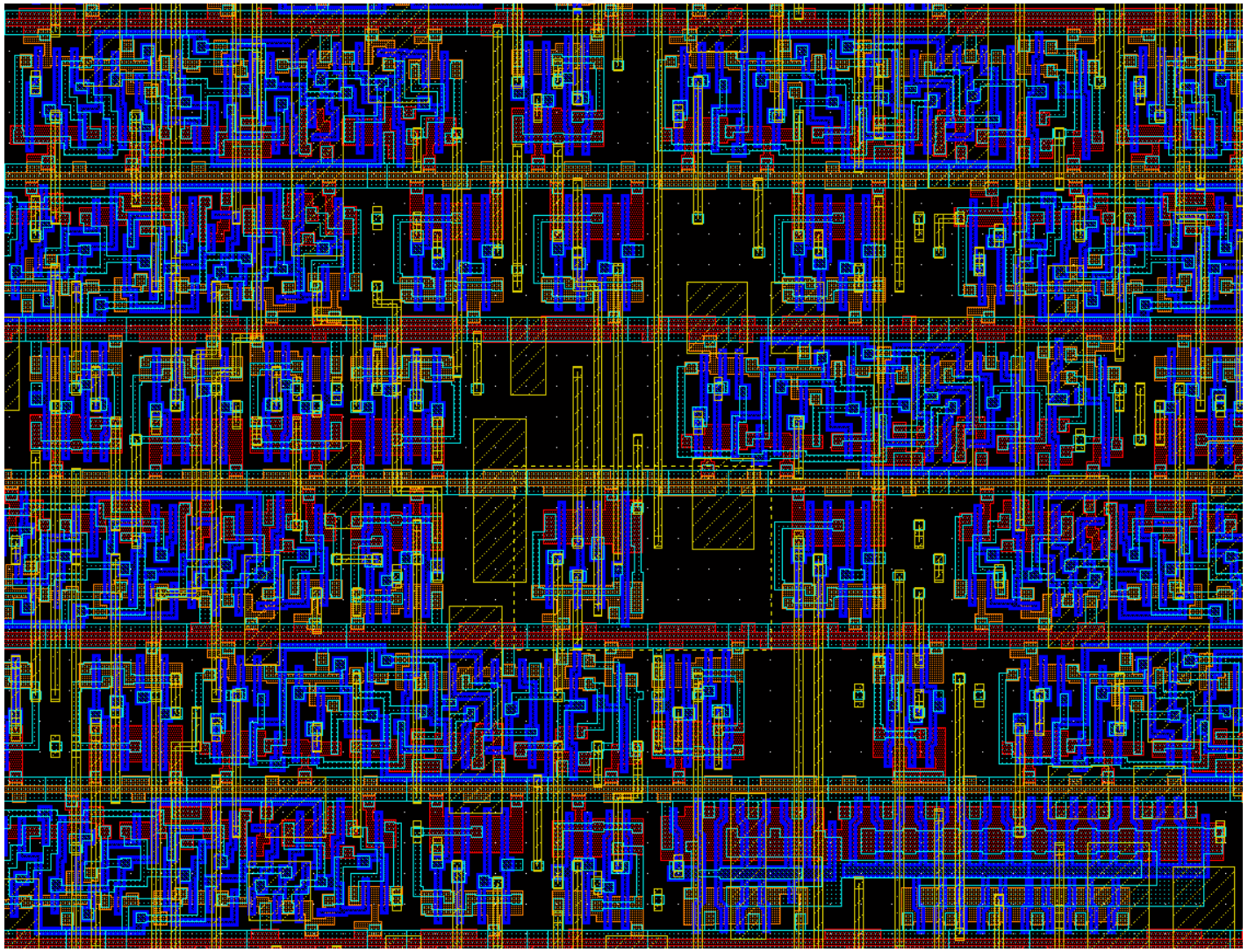
Standard-cells (com metal1 agora)



fernando.moraes@pucrs.br



Standard-cells (agora com zoom no roteamento entre células)



fernando.moraes@pucrs.br

Geração Automática

Automatiza a geração das bibliotecas de células ou blocos

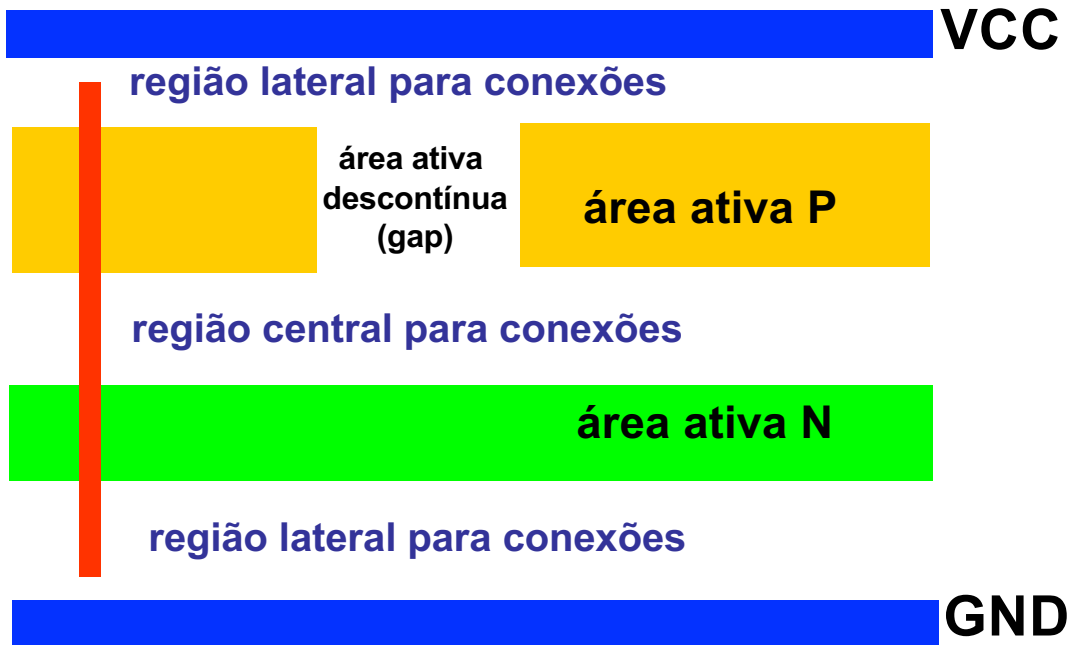
Costuma-se dividir os blocos funcionais em

1. **Lógica aleatória (ou randômica)**
   - **O layout não segue um padrão rígido**
   - **Gerado a partir de linguagens de descrição de hardware, como VHDL ou Verilog**
2. **Lógica regular**
   - **Repetição de padrões: aritméticos, memórias RAM e ROM**

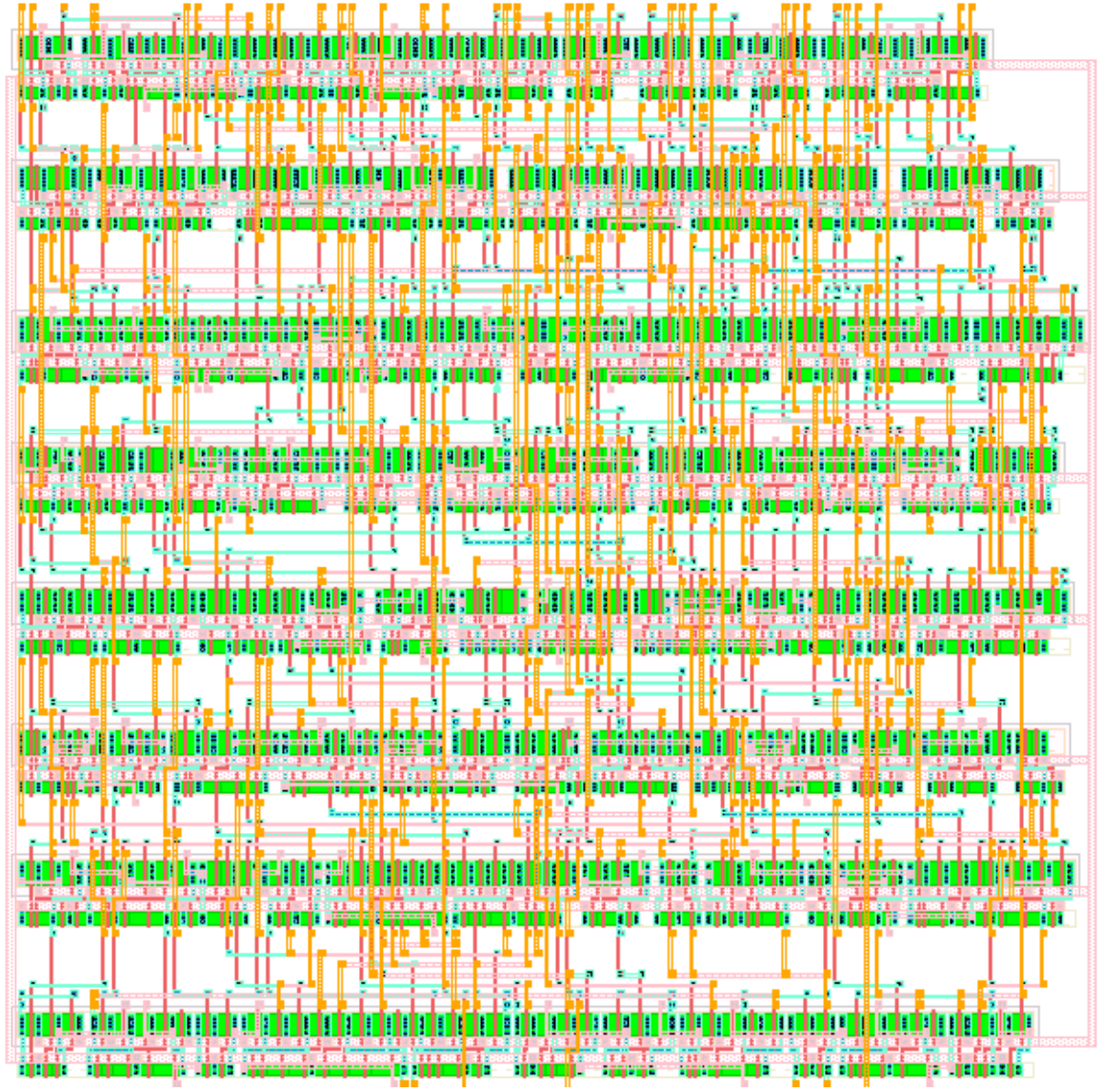
Geração Automática – Lógica Aleatória

Segue uma topologia básica para o posicionamento dos transistores – linhas horizontais

Estratégias: gate matrix (Lopez, 1980) **e linear matrix** (Uehara e van Cleemput, 1981)



Geração Automática – Lógica Aleatória



Exemplo de Linear Matrix: Tropic
(Moraes, 1997)

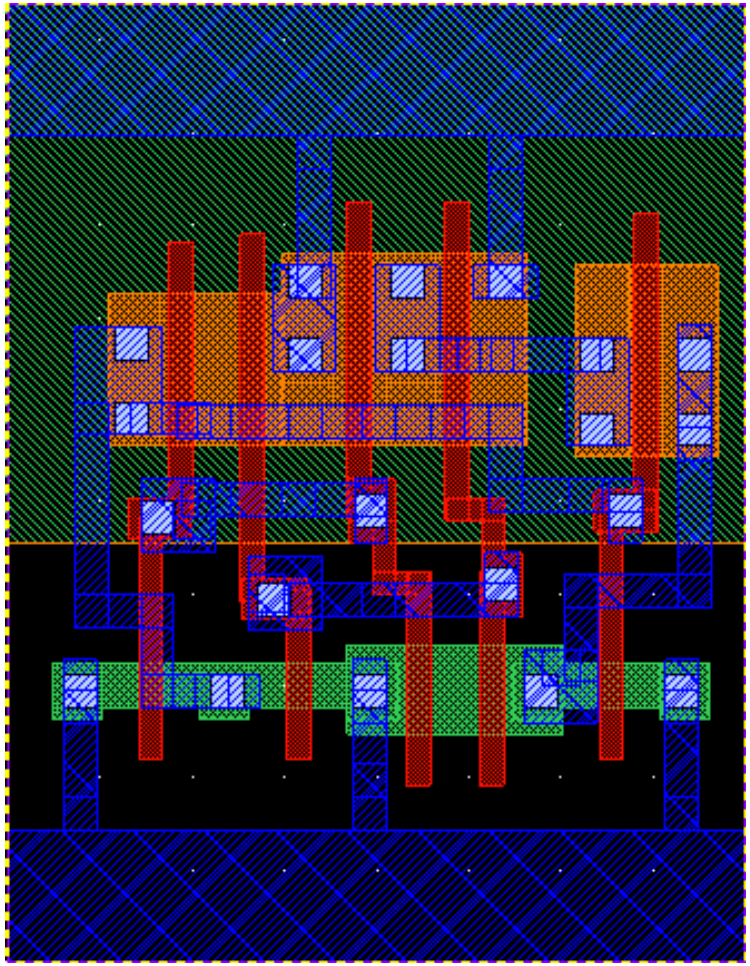
Antigo, buscava gerar um circuito completo

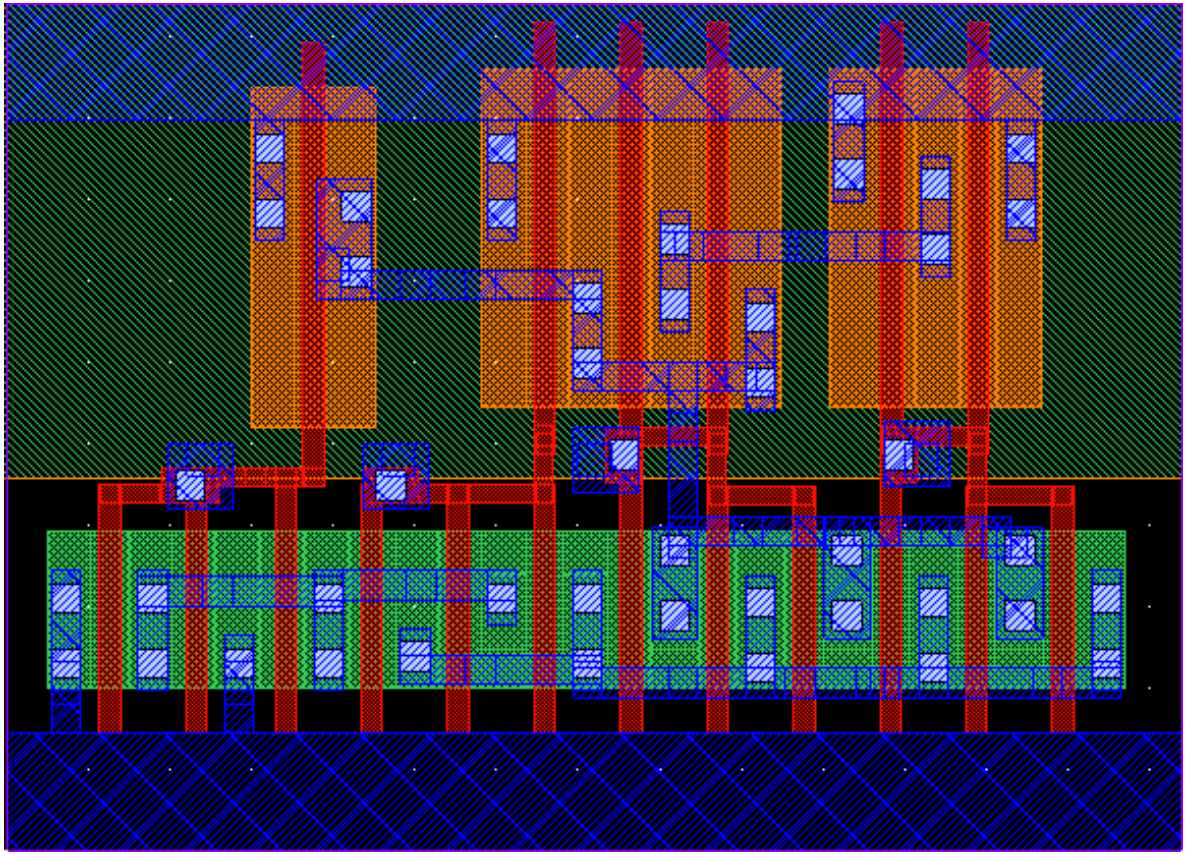
Uso de canais de roteamento

Geração Automática – Lógica Aleatória

ASTRAN

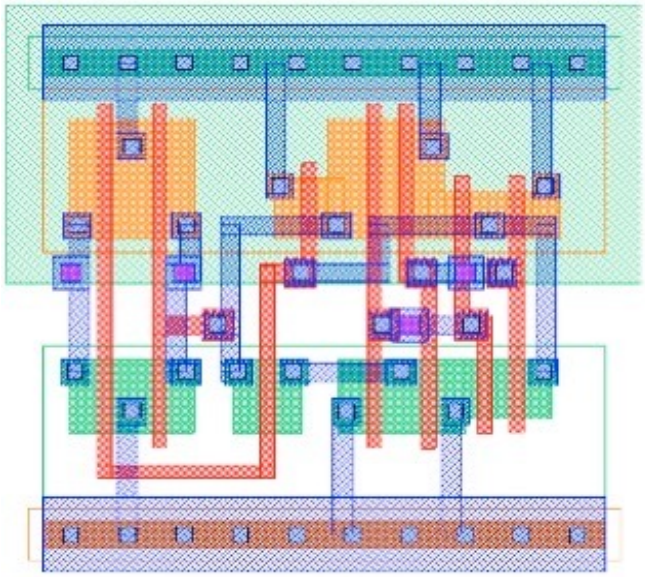
- Gerador de automático de células (2015)
- http://aziesemer.github.io/astran



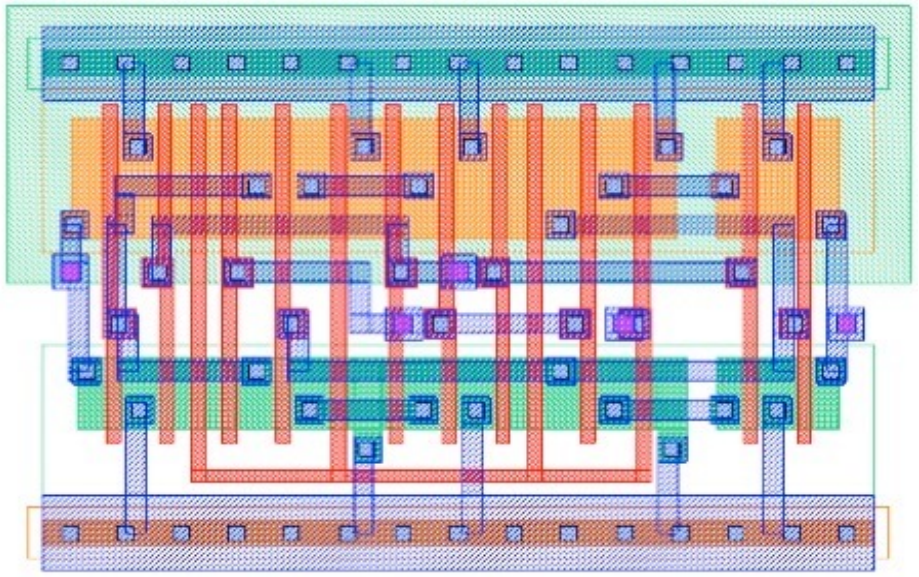


Geração Automática – Lógica Aleatória

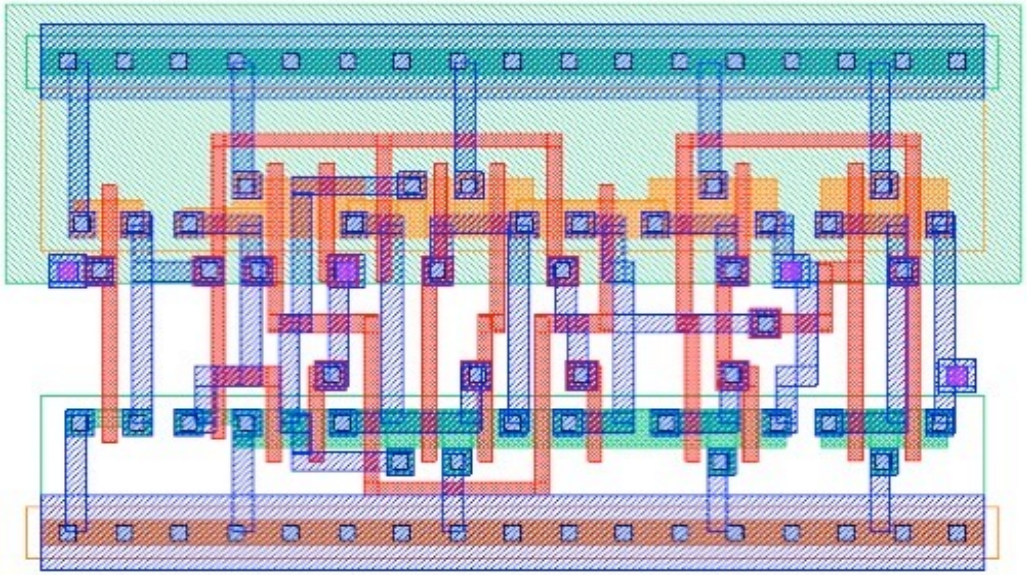
Células geradas pelo Astran



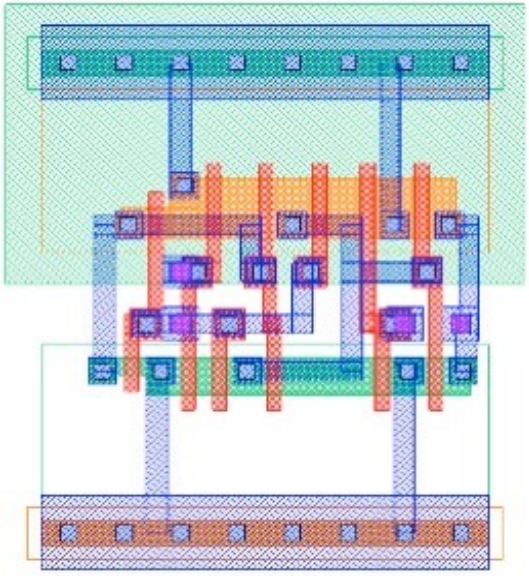
ADD22



ADD32

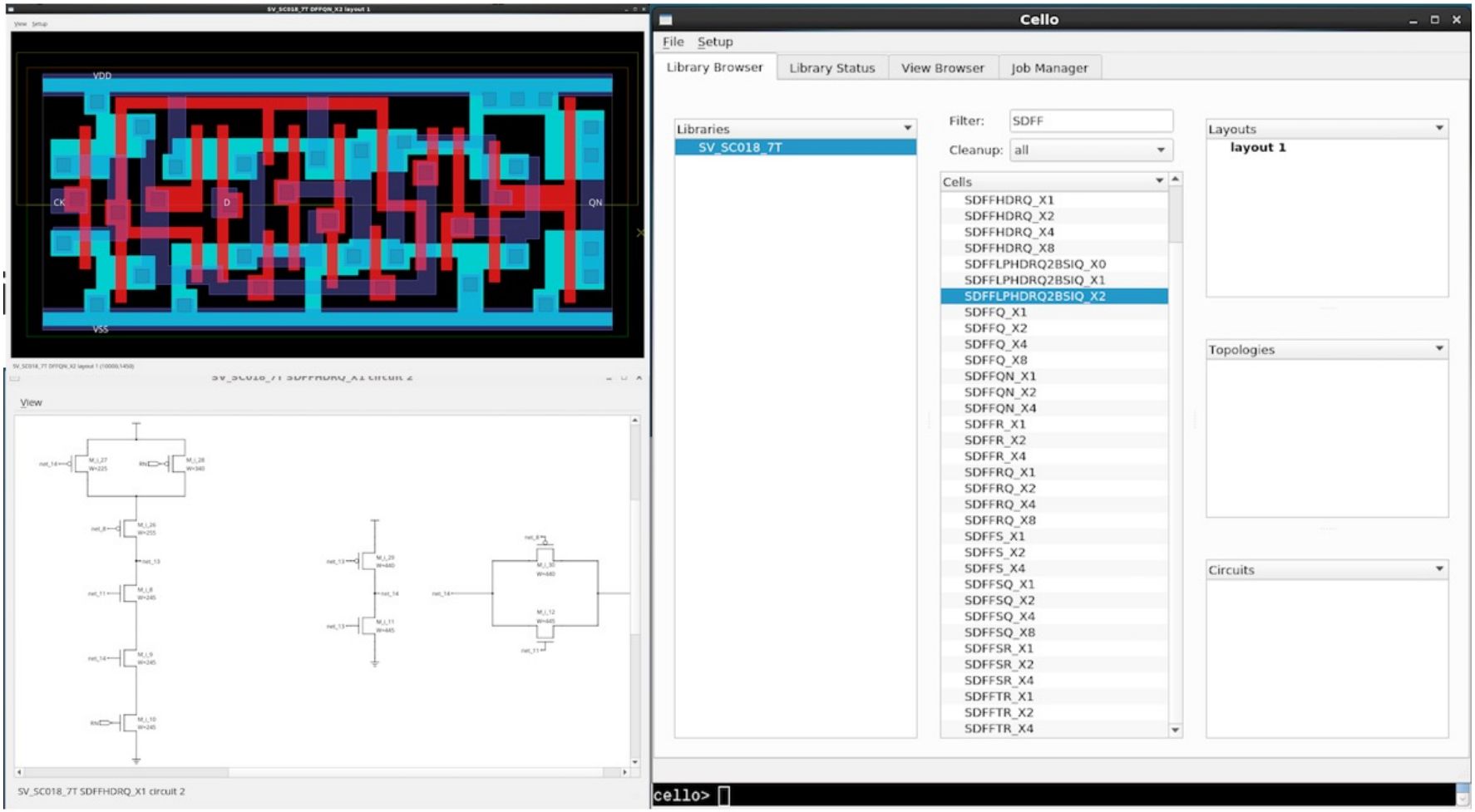


DF1



MUX21

Geração Automática – Indústria – Cello



Silvaco Cello™ is the industry's versatile, integrated, and easyto-use solution for digital cell library creation and optimization. It enables designers of digital CMOS ICs to custom-tailor digital cell libraries and explore the impact of alternate device models, design rules, and cell architectures, as well as process migration.

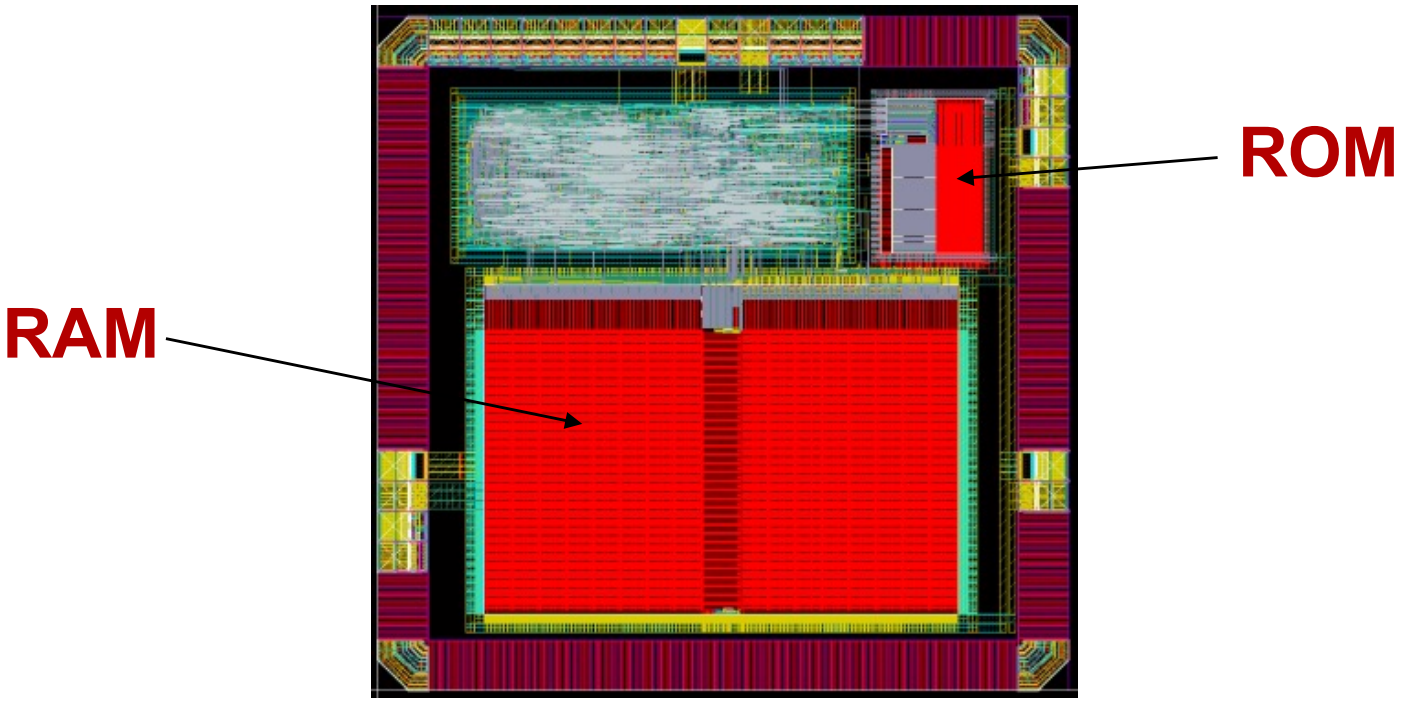
https://www.silvaco.com/products/nangate/

Geração Automática – Lógica Regular

fernando.moraes@pucrs.br

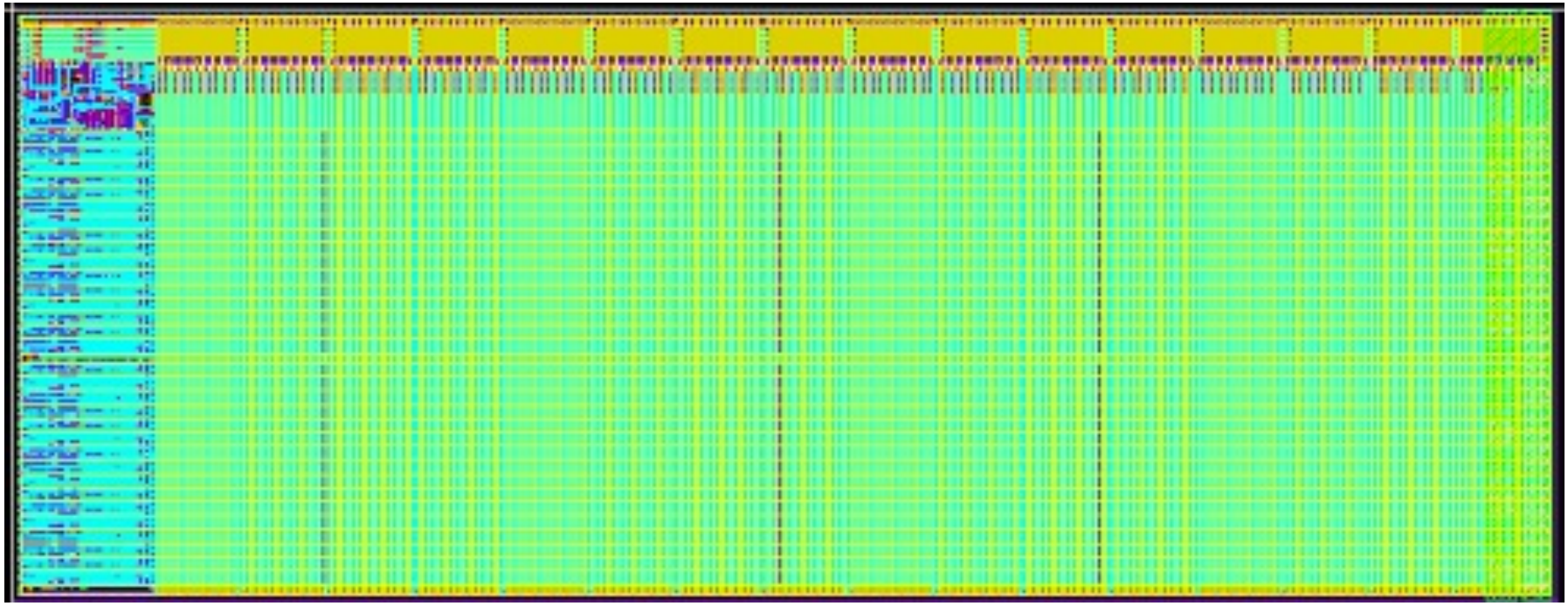
Blocos com funcionalidade específica

- Memórias RAM
- Memórias ROM
- Registradores (de armazenamento, contadores, deslocadores)
- Somadores, subtratores, multiplicadores e ULAs



Geração Automática – Lógica Regular

Exemplo de bloco de memória



256 x 32 (or 8192 bit) SRAM
Generated by hard-macro module generator

fernando.moraes@pucrs.br

Geração Automática – Lógica Regular

Exemplo de multiplicador

Ziesemer Jr., 2004

fernando.moraes@pucrs.br

Sumário

O início e o estado-da-arte

Introdução

Métodos de Projeto

- **Full Custom**
- **Standard Cells**
- **Geração Automática**
- **Pré-Difundidos**
- **Componentes Programáveis**
  - **Passado – PLAs**
  - **Atualidade - FPGAs**

Pré-Difundidos

Apresentam organização em forma de matriz de elementos

- Transistores
- Portas lógicas

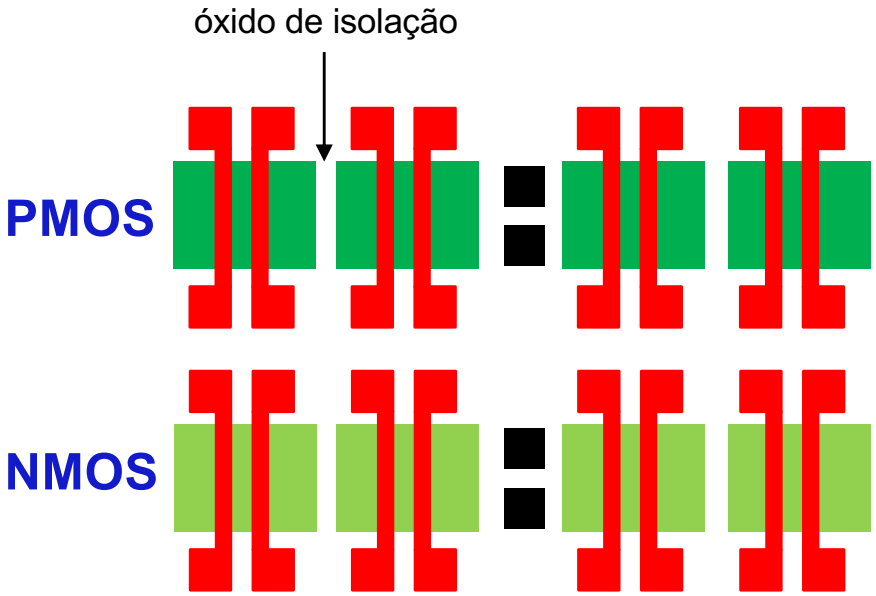
Matrizes são pré-processadas e armazenadas para posterior personalização

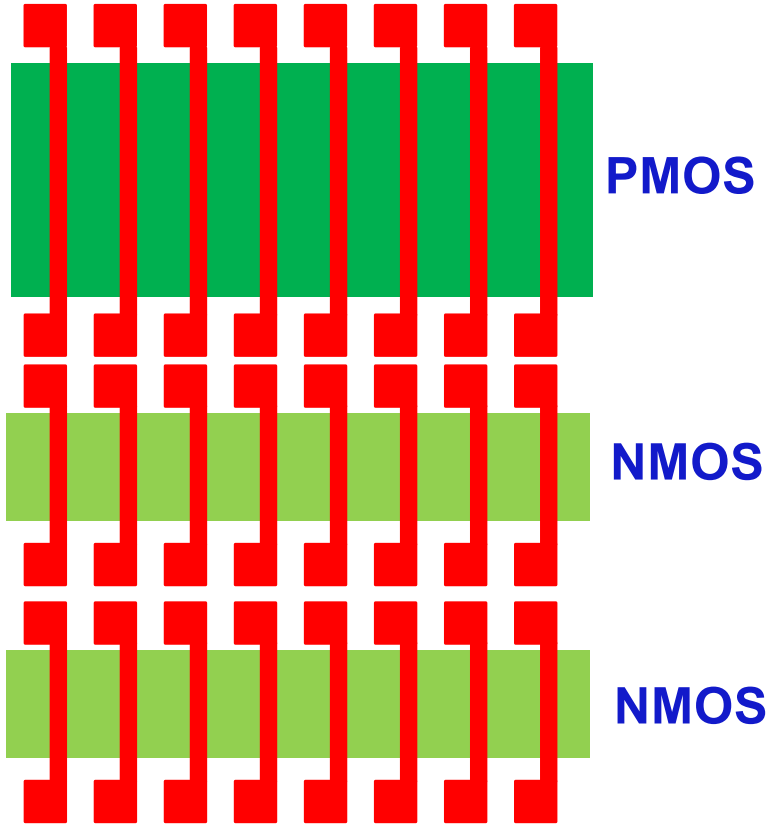
Vantagens

- redução do custo de fabricação
- diminuição do tempo para mercado

Pré-Difundidos

- **Microarquitetura**
  - **Célula de base (CB)**



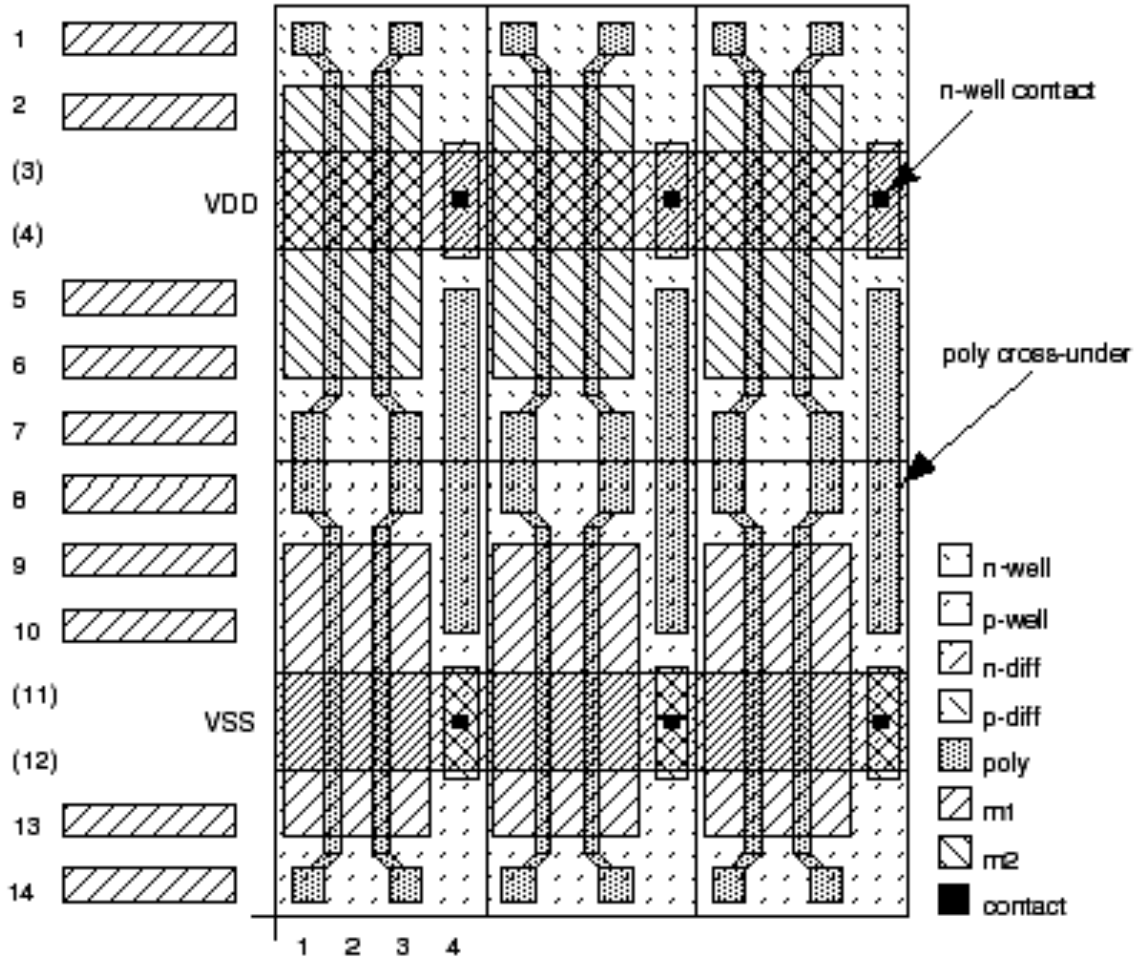


fernando.moraes@pucrs.br

Pré-Difundidos

Transistores já fabricados

- Definição da função por fabricação das camadas de metal e contato



Problema: "grão" muito fino - transistor

fernando.moraes@pucrs.br

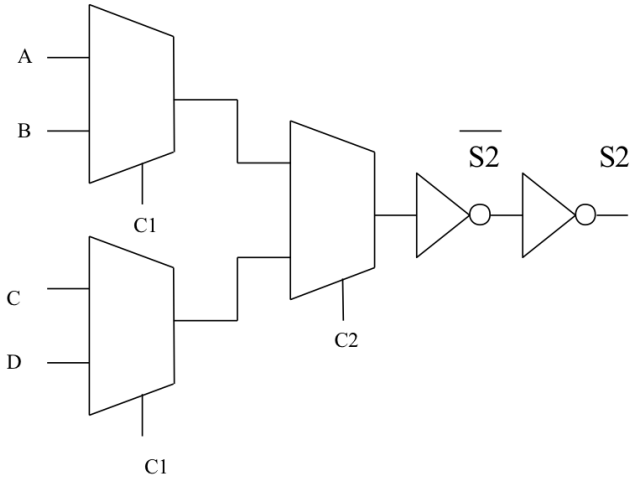
Pré-Difundidos

Sea-of-Gates

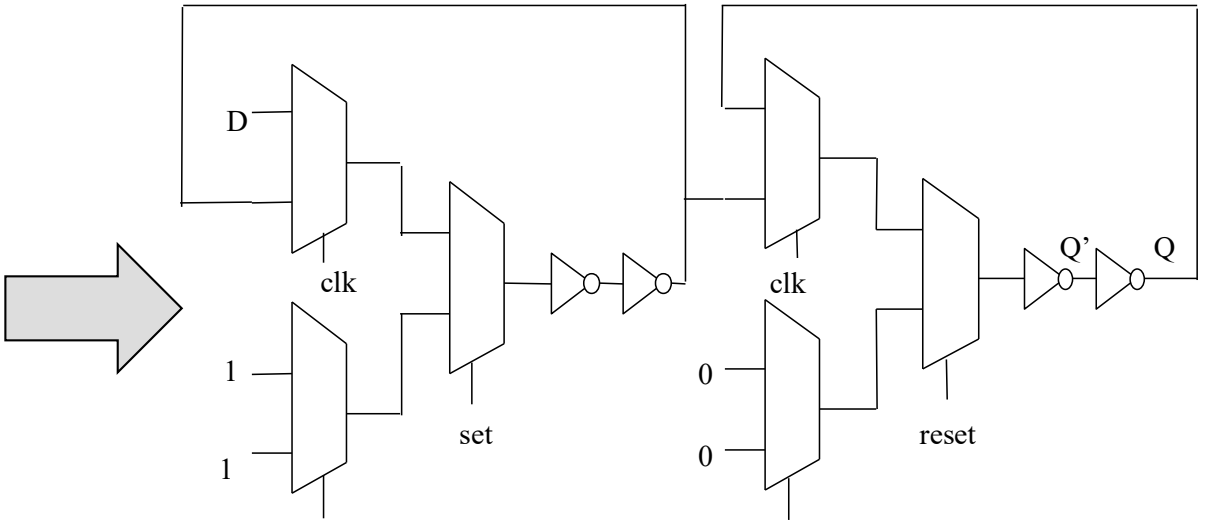
- Evolução do gate-array
- Ao invés de transistores temos “portas lógicas universais”
- Foi o método que deu origem aos FPGAs

fernando.moraes@pucrs.br

Exemplos:



Arranjo de 3 multiplexadores e inversores



Implementação de FLIP-FLOP MESTRE-ESCRAVO com SET e RESET

Pré-Difundidos

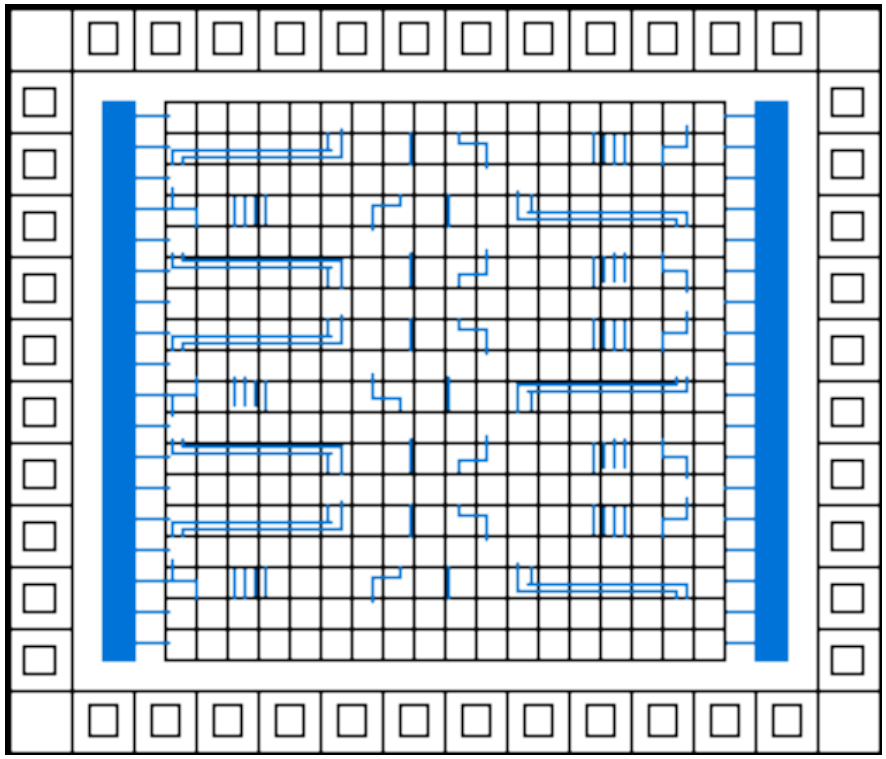
Exemplo de Macroarquitetura: Sea-of-Gates

O roteamento é realizado sobre os transistores

A matriz é completamente preenchida por “portas lógicas universais”

Não há canais explícitos de roteamento

Necessita de tecnologia CMOS com três ou mais níveis de metal



fernando.moraes@pucrs.br

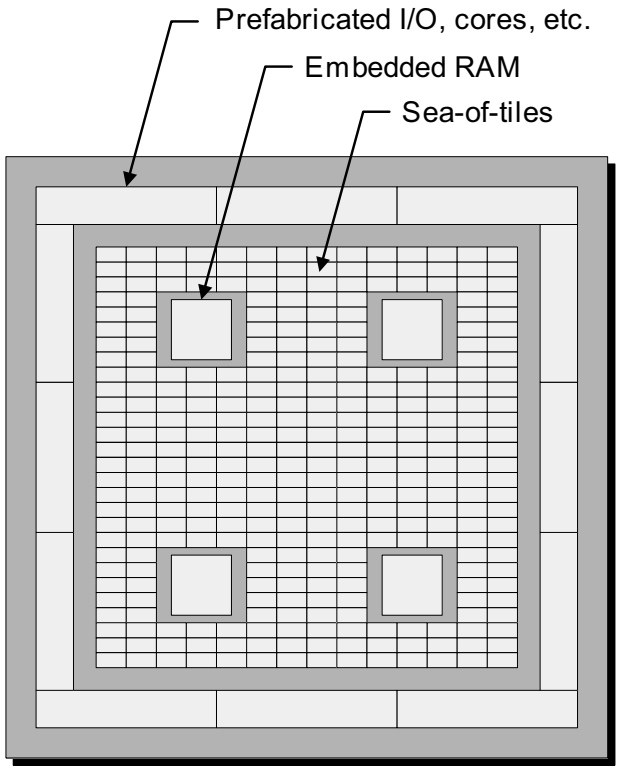
Structured ASICs – evolução

fernando.moraes@pucrs.br

Um “ASIC estruturado” fica entre um FPGA e um ASIC padrão baseado em célula

ASICs estruturados são usados principalmente para projetos em volume médio de produção

A lógica aleatória é mapeada na área “sea-of-gate” e os demais módulos são mapeados nos IPs disponíveis



Artigo: Structured ASICs: Opportunities and Challenges

Sumário

O início e o estado-da-arte

Introdução

Métodos de Projeto

- **Full Custom**
- **Standard Cells**
- **Geração Automática**
- **Pré-Difundidos**
- **Componentes Programáveis**
  - **Passado – PLAs**
  - **Atualidade - FPGAs**

Componentes Configuráveis

Configuração pelo projetista

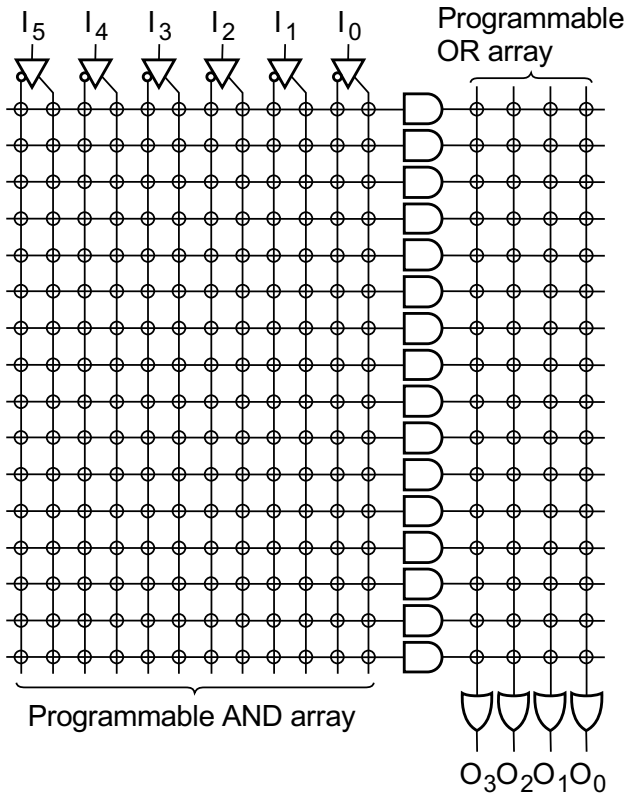
Programação “em campo” (field-programmable)

O CI já foi fabricado e encapsulado

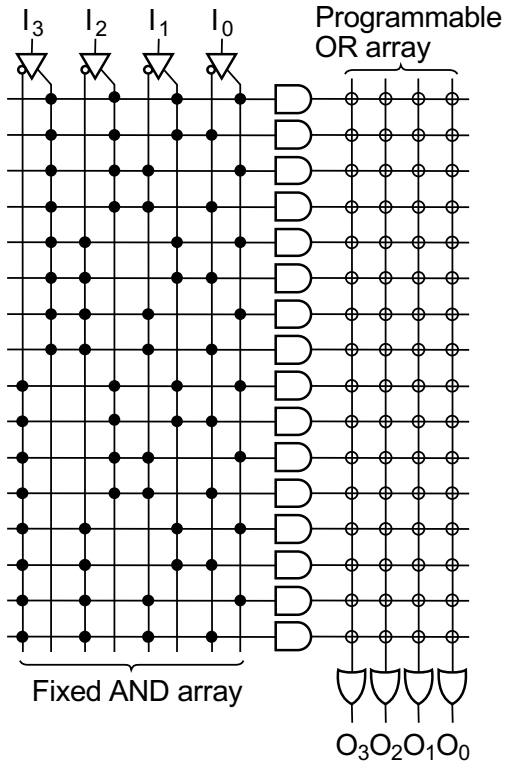
Pode ser feita com um equipamento especial ou na própria placa em que irá o componente

Array-Based Programmable Logic

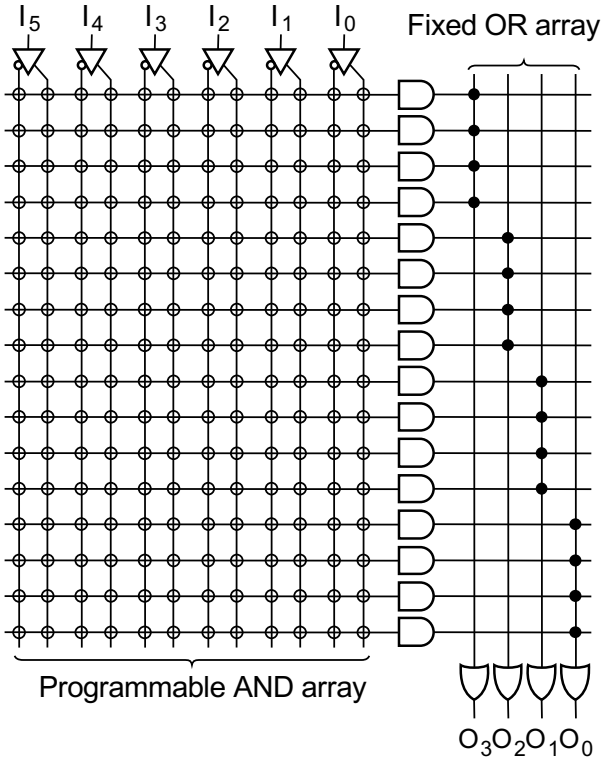
fernando.moraes@pucrs.br



PLA



PROM

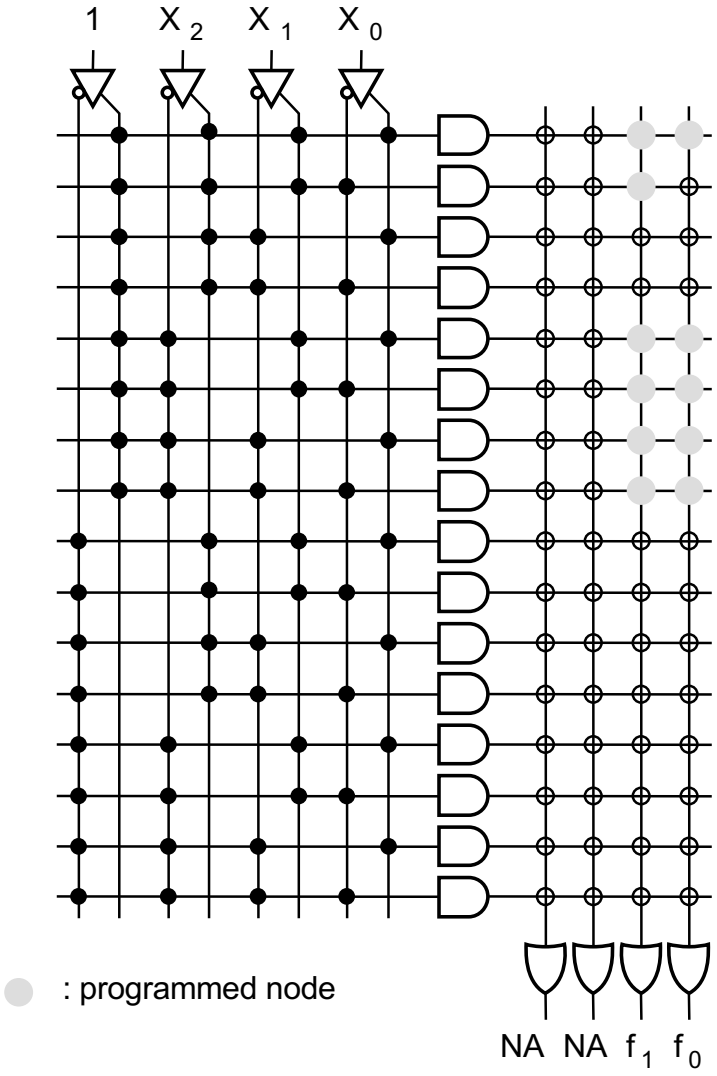


PAL

Indicates programmable connection

Indicates fixed connection

Programming a PROM



$x_0x_1x_2$

$x_0x_1{\sim}x_2$

● : programmed node

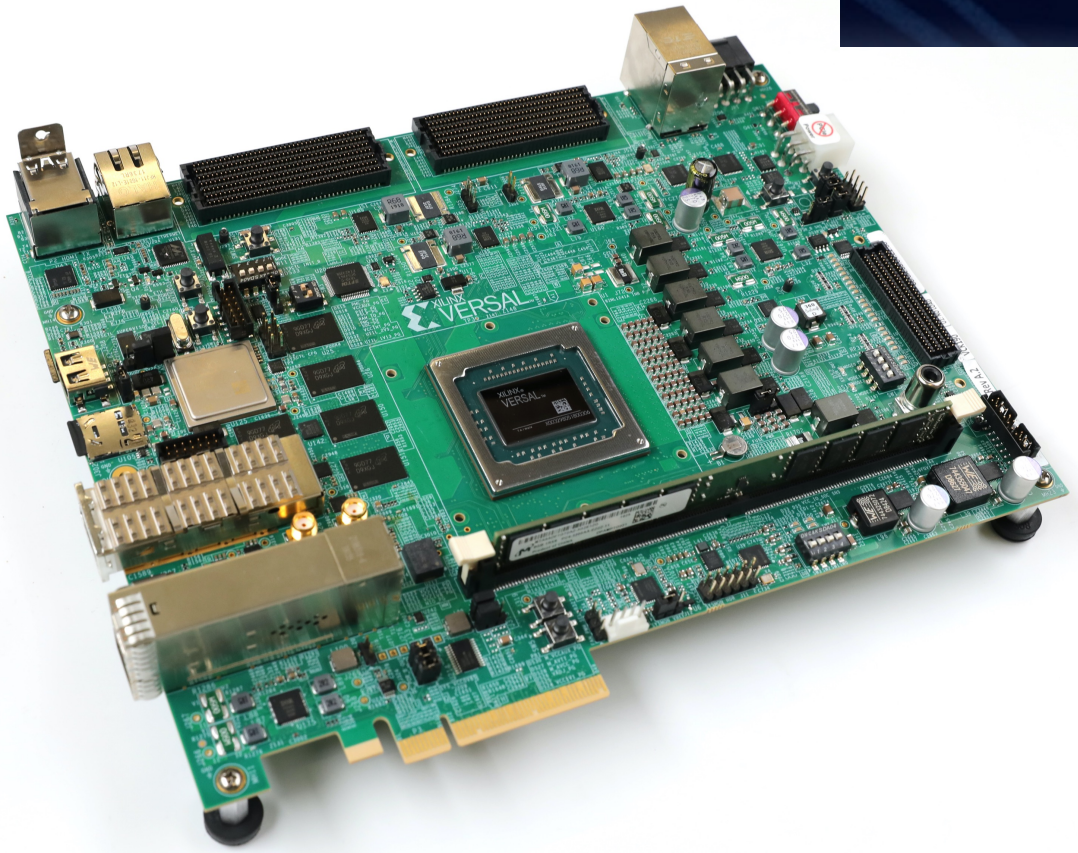
$$f_0 = x_0x_1 + \overline{x_2}$$

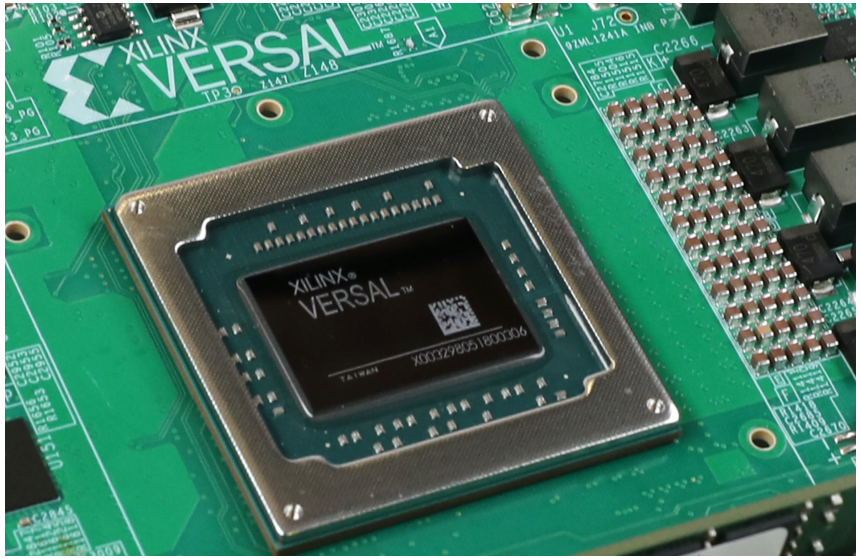
$$f_1 = x_0x_1x_2 + \overline{x_2} + \overline{x_0}x_1$$

fernando.moraes@pucrs.br

FPGA

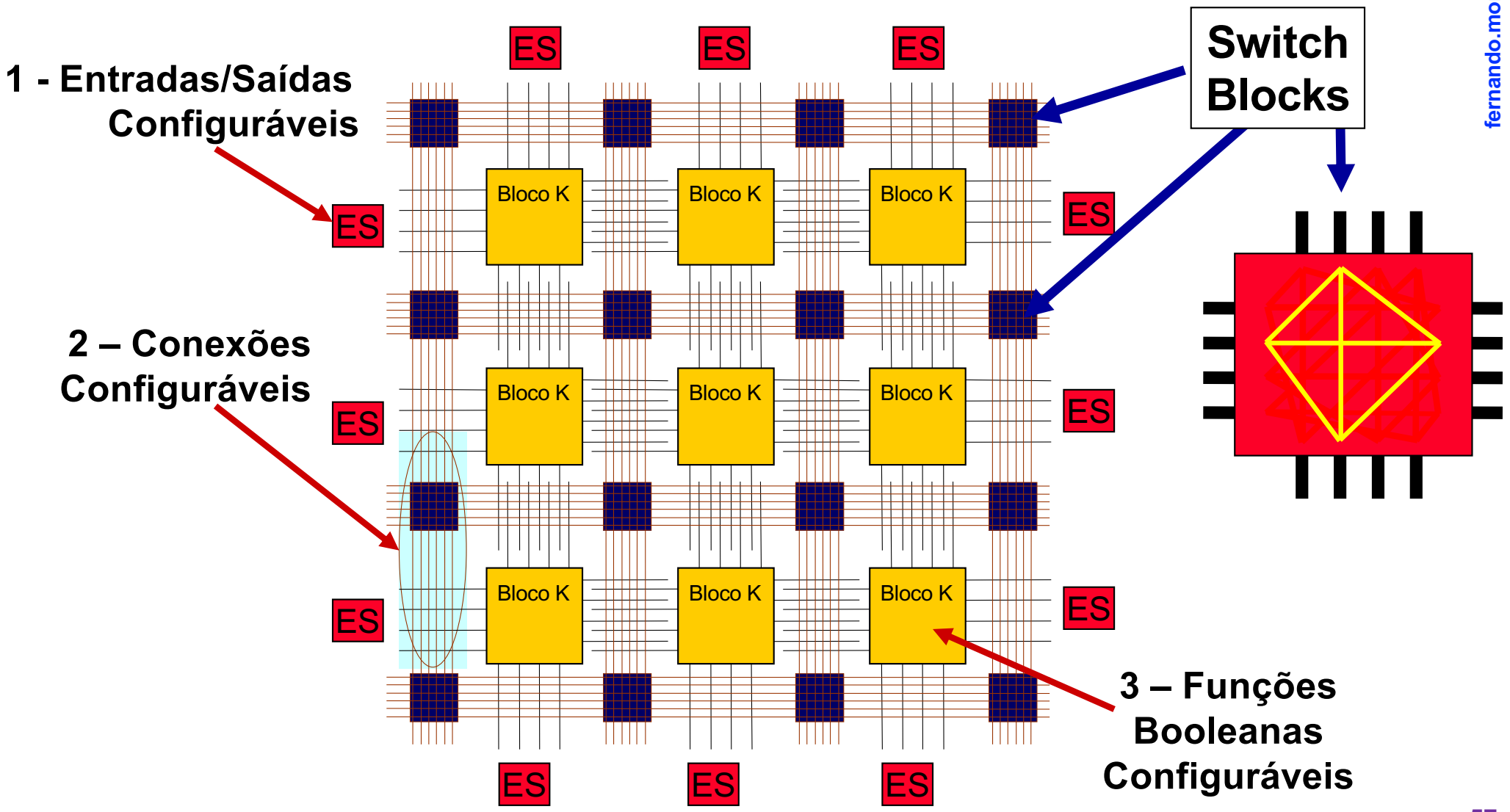
Field-**p**rogrammable
gate **a**rray





FPGAs – conceitos básicos

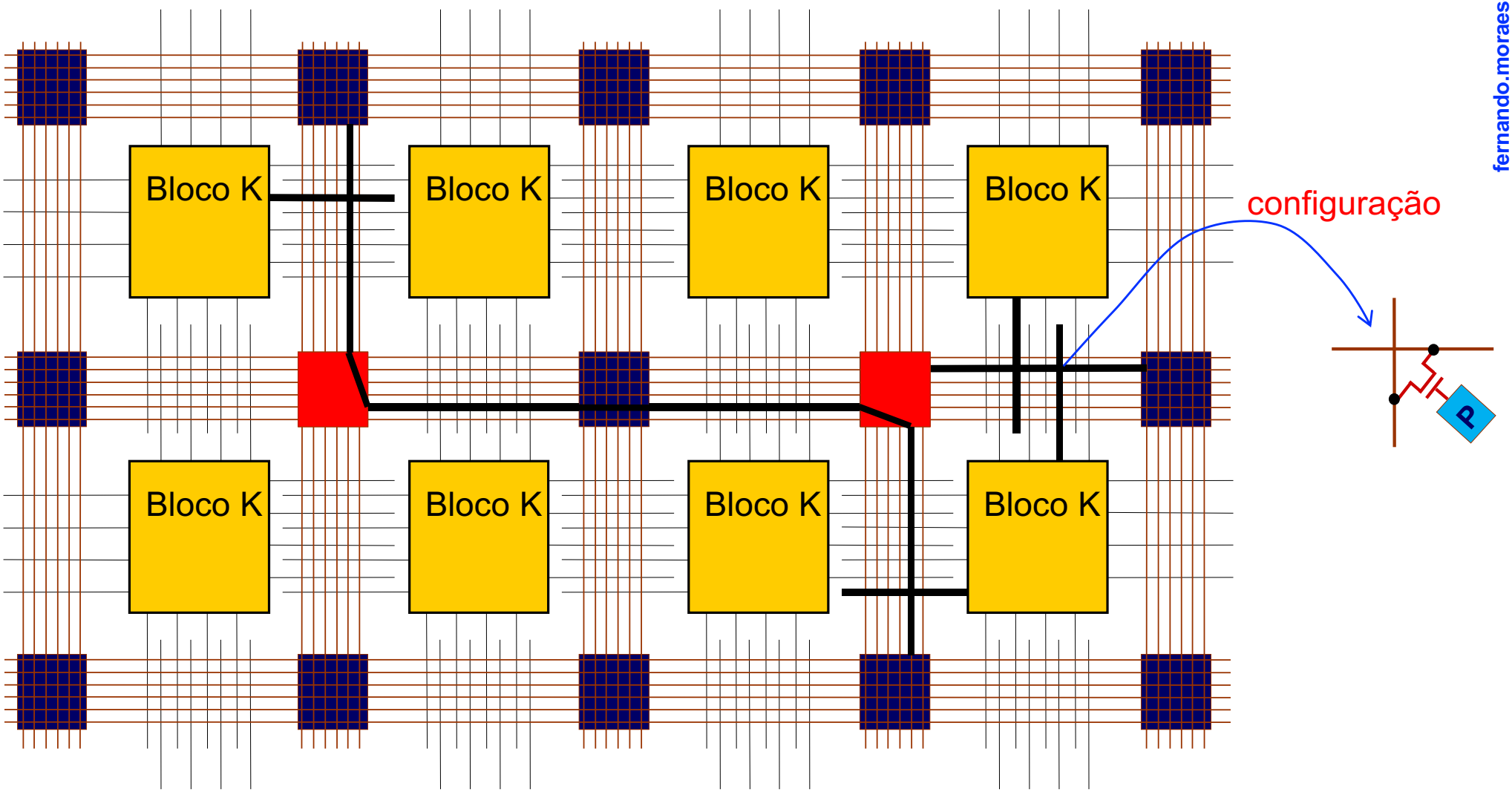
Matriz de CLBs (configurable logic blocks) interconectados por roteamento configurável



fernando.moraes@pucrs.br

FPGAs – Conceitos Básicos

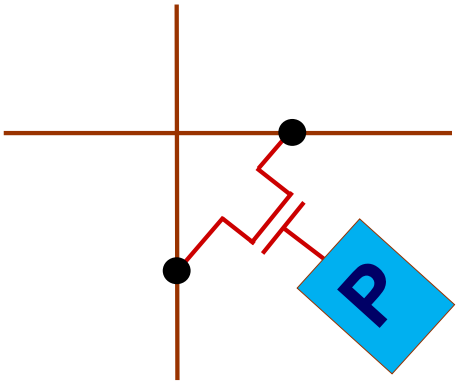
- **Exemplo de conexão entre duas redes**



fernando.moraes@pucrs.br

Tecnologias de Configuração

Diferentes tecnologias utilizadas para realizar a configuração de um FPGA:



Antifusível: configuração única (Quicklogic)

Flash: configuração mantida com o chip desconectado da alimentação (Microsemi – família IGLOO2)

SRAM: configuração deve ser realizada cada vez que o FPGA for alimentado (Xilinx, Altera)

fernando.moraes@pucrs.br

FPGAs – Configuração (RAM-based)

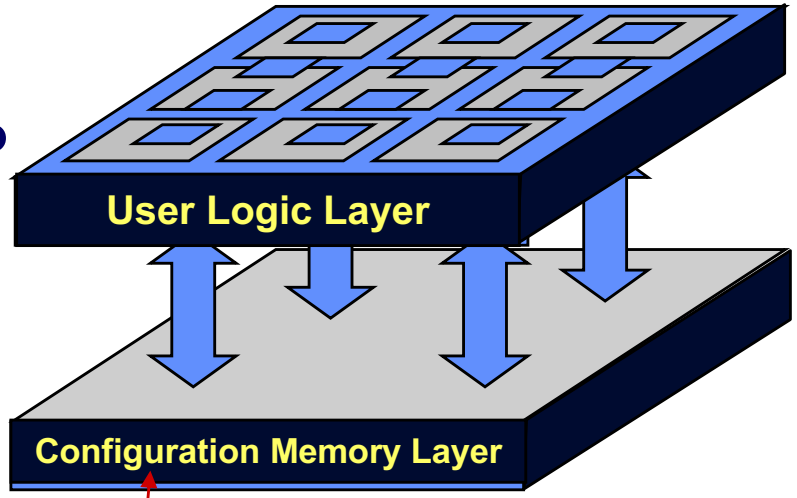
fernando.moraes@pucrs.br

FPGA deve ser visto como "duas camadas"

- Memória de configuração
- Lógica do usuário

Memória de configuração define

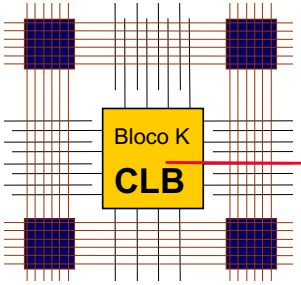
- Configuração dos pinos de E/S
- Toda o roteamento da lógica do usuário
- Definição das funções lógicas
- Configuração e conteúdo de blocos de memórias



memória de configuração: dezenas de MB

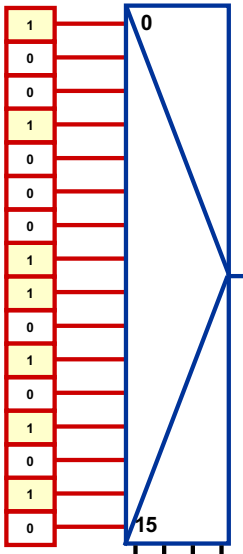
LUT – Gerador Universal de Funções

- **LUT – Look-Up Table – função booleana reconfigurável**
  - **Uma porção de hardware configurável/reconfigurável capaz de implementar qualquer tabela verdade de n entradas**
  - **Para n=4:** $2^{(2)^4}$ **= 65.536 funções implementáveis**
  - » Altamente flexível
  - » Método mais utilizado (Xilinx, Altera e outros)



Várias LUTs em uma CLB

A tabela verdade da função é armazenada em uma memória durante a configuração do FPGA



As entradas (variáveis Booleanas) controlam um multiplexador 2^n:1

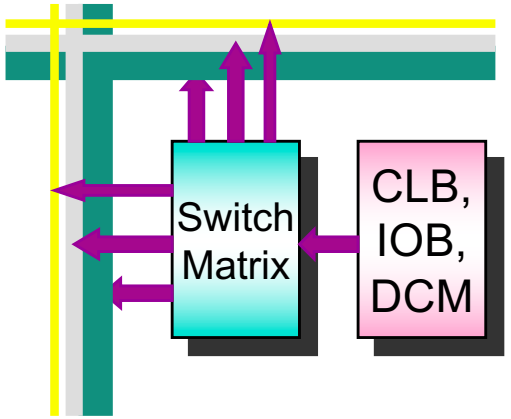
A B C D

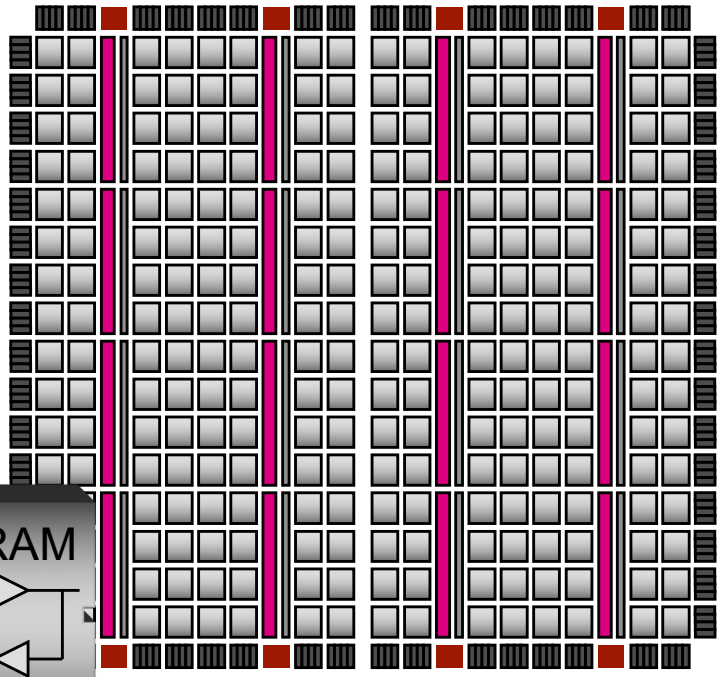
$$F = \bar{A}.\bar{B}.\bar{C}.\bar{D} + A.\bar{D} + \bar{A}.C.D$$
$$F = \sum(0, 3, 7, 8, 10, 12, 14)$$

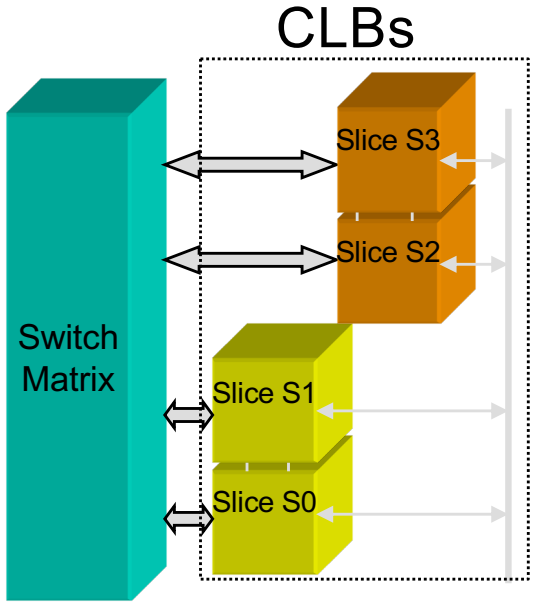
Considerando 150 transistores / LUT4, para 50.000 LUTS ➔ 7.500.000 transistores!!!

fernando.moraes@pucrs.br

Arquitetura Virtex





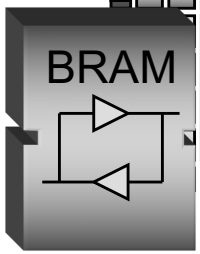


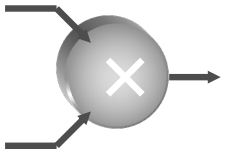
8 LUTs

- **Lógica (uso primário)**
- **128b RAM distribuída**
- **Shift registers**
- **Wide Input functions (32:1)**

Block RAM

18KBits True Dual Port





Multiplicadores

- 18b x 18b mult
- 200MHz pipelined

fernando.moraes@pucrs.br

Arquitetura Virtex

Conexões diretas entre CLBs vizinhas

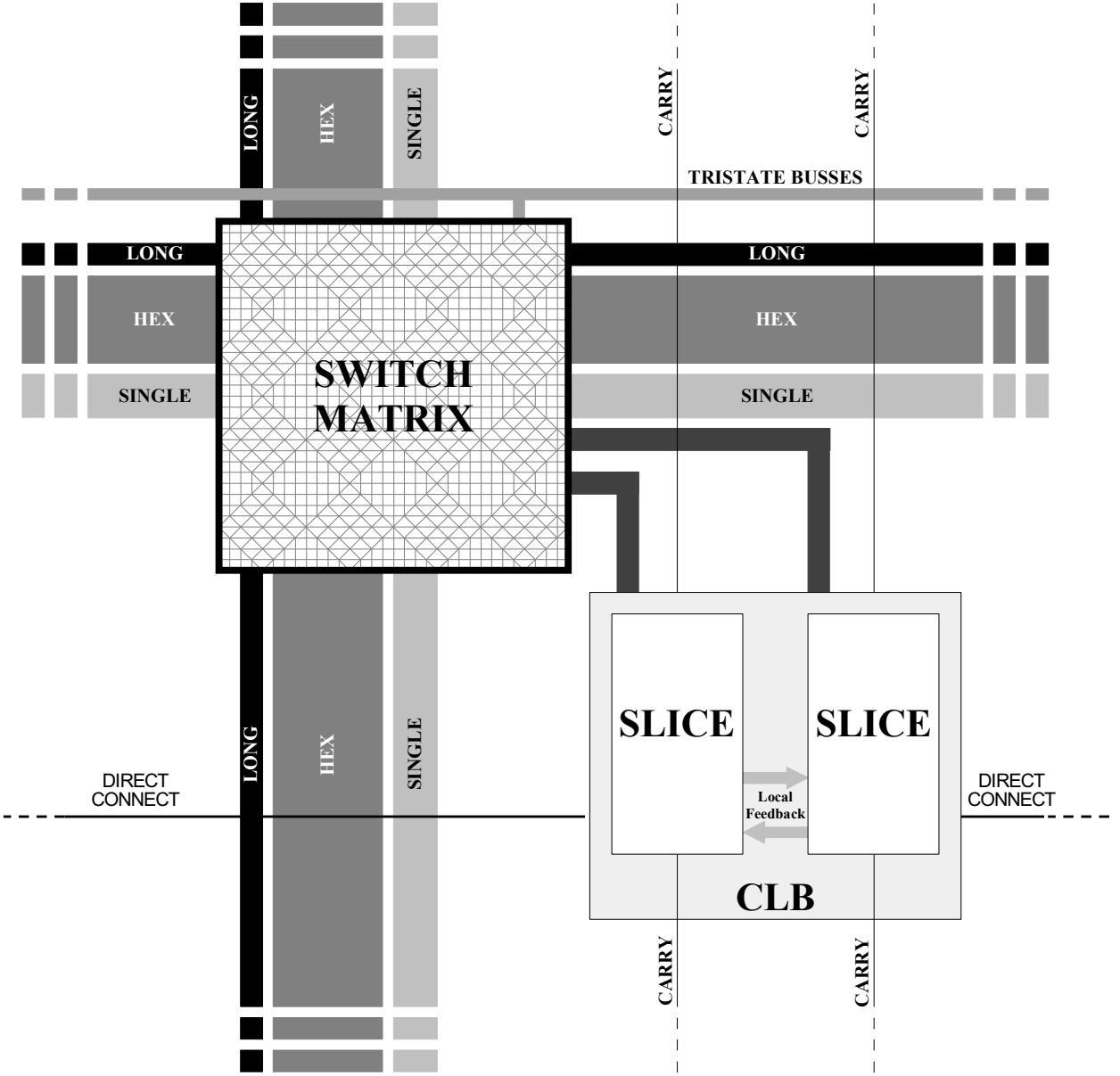
- Lógica de *vai-um*

Matrix de conexão

- CLB às linhas de roteamento

Linhas de roteamento

- Simples
- Hexas
- Longas
- Tri-state



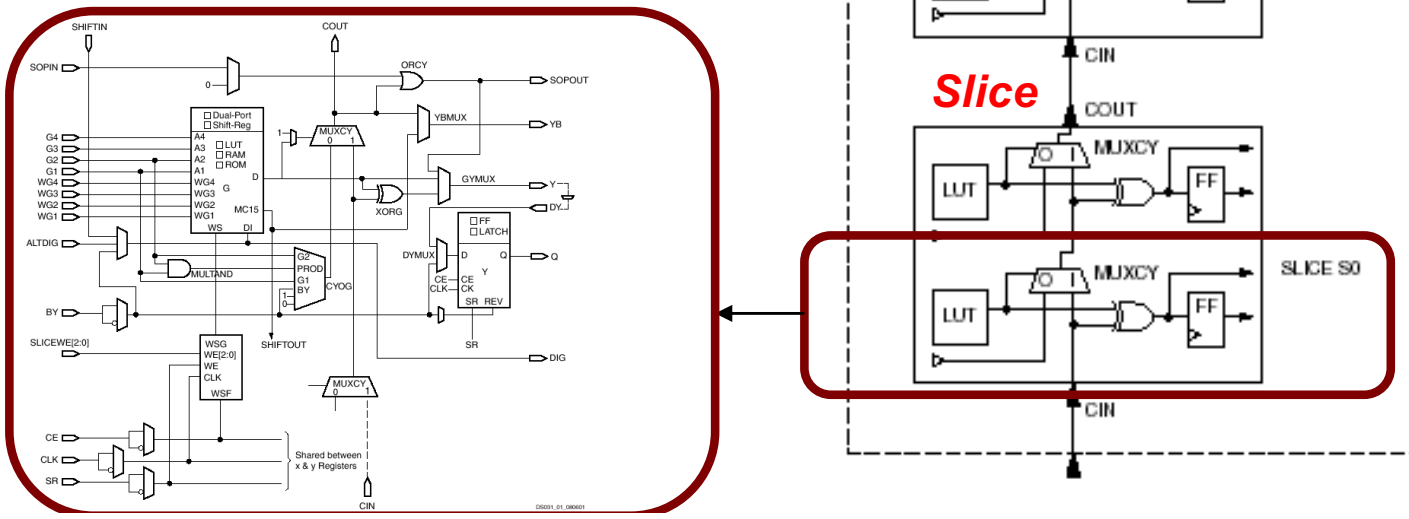
fernando.moraes@pucrs.br

Arquitetura do CLB do Dispositivo VIRTEX

- Fast Carry Logic Path
- Provides fast arithmetic add and sub

RESUMINDO O CLB

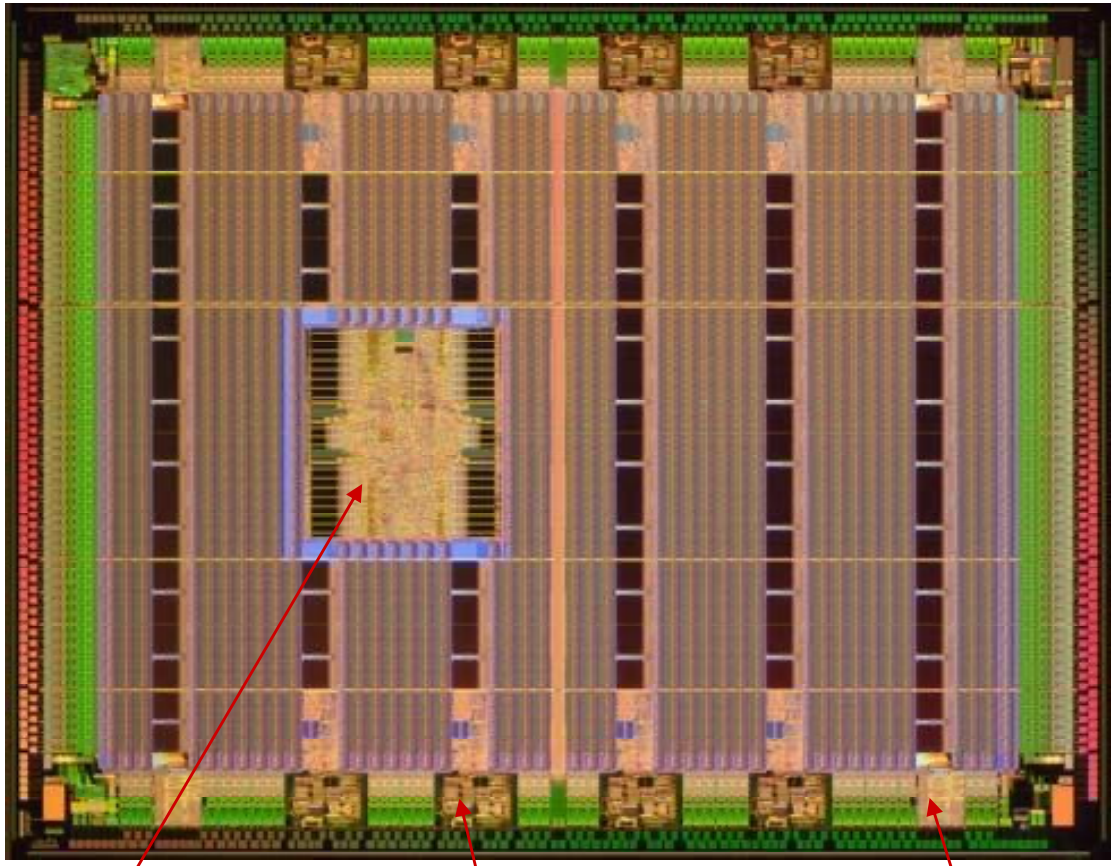
- **4 Slices**
- **8 LUTS / 8 Flip-Flops**
- **2 cadeias de vai-um**
- **64 bits para memória**
- **64 bits para shift-register**



fernando.moraes@pucrs.br

XC2VP7 Virtex-II Pro FPGA (antigo)

Layout do XC2VP7



fernando.moraes@pucrs.br

Demais Componentes de FPGA Moderno (1/2)

Gerenciamento de clock

- Reduz escorregamento de relógio
- Permite multiplicar, dividir, mudar a fase da(s) freqüências de entrada
- Implementações digitais (DCM – Xilinx, mais baratos) e analógica (PLL – Altera, mais felxíveis)

Blocos de memória embarcada

- Tipicamente blocos de 18kbits ou 36Kbits na Xilinx (Altera tem mais variedade de blocos de memória)

Blocos DSP

- Multiplicadores 18bitsx18bits para funções de imagem, áudio, telecomunicações

fernando.moraes@pucrs.br

Demais Componentes de FPGA Moderno (2/2)

Processadores embarcados do tipo hard macro

- Xilinx disponibiliza o processador ARM
- Podem executar sistemas operacionais embarcados como Linux

Transceptores Gigabit

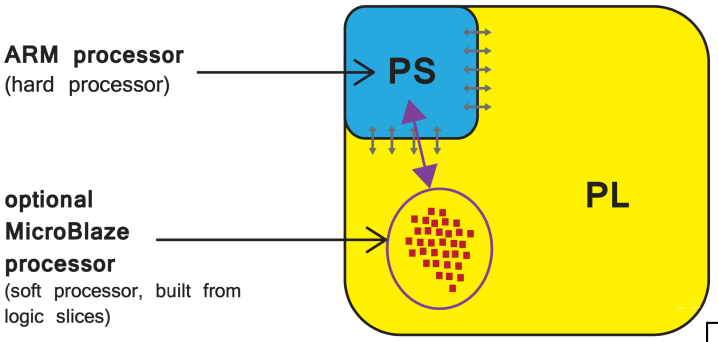
- Blocos serializadores / deserializadores para receber dados em altas taxas de transmissão
- Virtex-4 é capaz de receber e transmitir dados em freqüências de 3.2 Gbps usando dois fio

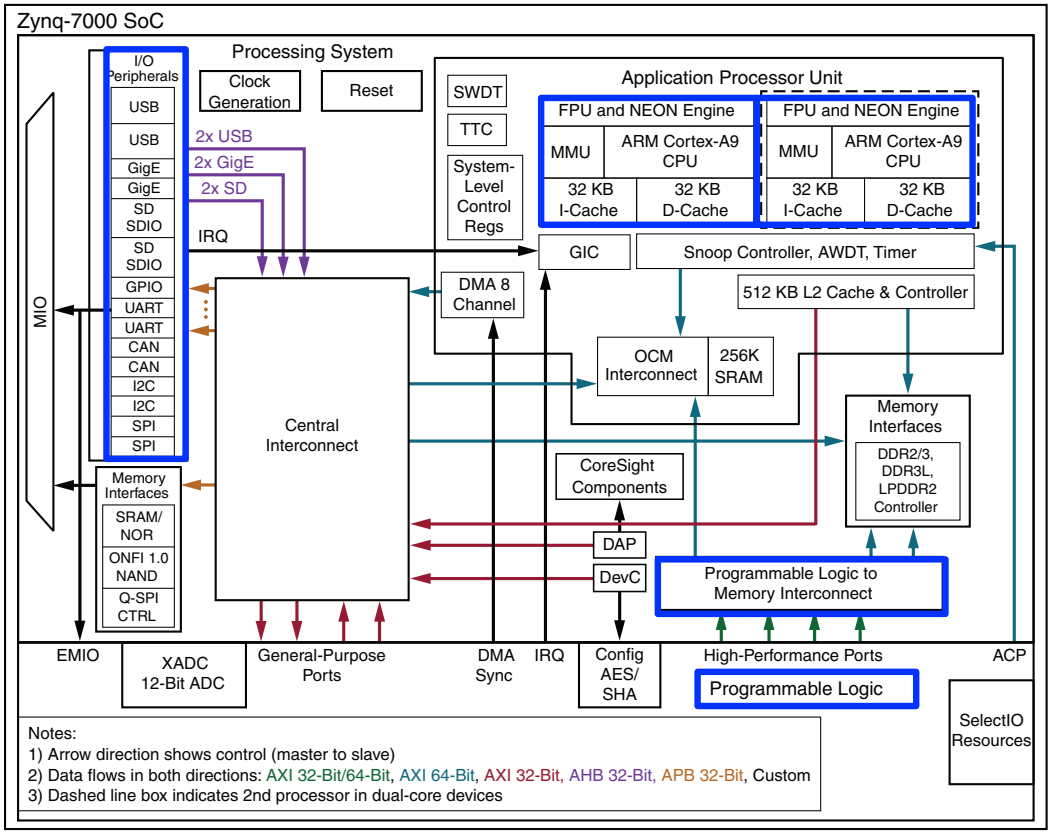
Outros

- Ethernet MAC
- Criptografia do bitstream
- Controle para reconfiguração interna (de dentro do FPGA - ICAP)

fernando.moraes@pucrs.br

FPGAs atuais - SoCs





fernando.moraes@pucrs.br

FPGAs atuais - SoCs

fernando.moraes@pucrs.br

Table 1: **Zynq-7000 and Zynq-7000S SoCs *(Cont'd)***

	Device Name	Z-7007S	Z-7012S	Z-7014S	Z-7010	Z-7015	Z-7020	Z-7030	Z-7035	Z-7045	Z-7100
	Part Number	**XC7Z007S**	**XC7Z012S**	**XC7Z014S**	**XC7Z010**	**XC7Z015**	**XC7Z020**	**XC7Z030**	**XC7Z035**	**XC7Z045**	**XC7Z100**
Programmable Logic	Xilinx 7 Series Programmable Logic Equivalent	Artix®-7 FPGA	Artix-7 FPGA	Artix-7 FPGA	Artix-7 FPGA	Artix-7 FPGA	Artix-7 FPGA	Kintex®-7 FPGA	Kintex-7 FPGA	Kintex-7 FPGA	Kintex-7 FPGA
	Programmable Logic Cells	23K	55K	65K	28K	74K	85K	125K	275K	350K	444K
	Look-Up Tables (LUTs)	14,400	34,400	40,600	17,600	46,200	53,200	78,600	171,900	218,600	277,400
	Flip-Flops	28,800	68,800	81,200	35,200	92,400	106,400	157,200	343,800	437,200	554,800
	Block RAM (# 36 Kb Blocks)	1.8 Mb (50)	2.5 Mb (72)	3.8 Mb (107)	2.1 Mb (60)	3.3 Mb (95)	4.9 Mb (140)	9.3 Mb (265)	17.6 Mb (500)	19.2 Mb (545)	26.5 Mb (755)
	DSP Slices (18x25 MACCs)	66	120	170	80	160	220	400	900	900	2,020
	Peak DSP Performance (Symmetric FIR)	73 GMACs	131 GMACs	187 GMACs	100 GMACs	200 GMACs	276 GMACs	593 GMACs	1,334 GMACs	1,334 GMACs	2,622 GMACs
	PCI Express (Root Complex or Endpoint)[3]		Gen2 x4			Gen2 x4		Gen2 x4	Gen2 x8	Gen2 x8	Gen2 x8
	Analog Mixed Signal (AMS) / XADC	2x 12 bit, MSPS ADCs with up to 17 Differential Inputs									
	Security[2]	AES and SHA 256b for Boot Code and Programmable Logic Configuration, Decryption, and Authentication									

Intel® Agilex™
FPGA ▪ SoC

INTEL® AGILEX™ I-SERIES SOC FPGA PRODUCT TABLE

PRODUCT LINE		AGI 022	
Resources	Logic elements (LEs)	2,200,000	
	Adaptive logic modules (ALMs)	745,763	
	ALM registers	2,983,051	
	eSRAM memory blocks	0	0
	eSRAM memory size (Mb)	0	0
	M20K memory blocks	11,616	13,272
	M20K memory size (Mb)	210	259
	MLAB memory count	32,788	45,640
	MLAB memory size (Mb)	21	29.2
	Variable-precision digital signal processing (DSP) blocks	6,250	8,736
	18 x 19 multipliers	12,500	17,056
	Single-precison or half-precision tera floating point operations per second (TFLOPS)	9.4 / 18.8	11.8 / 23.6
Maximum Available Device Resources	Maximum differential (RX or TX) pairs	552	552
	Memory devices supported	DDR4, QDR IV, RLDRAM 3	
	Secure data manager	AES-256/SHA-256 bitstream encryption or authentication, physically unclonable function (PUF), ECDSA 256/384 boot code authentication, side channel attack protection	
	Hard processor system	Quad-core 64 bit Arm* Cortex*-A53 up to 1.5 GHz with 32 KB I/D cache, NEON* coprocessor, 1 MB L2 cache, direct memory access (DMA), system memory management unit, cache coherency unit, hard memory controllers, USB 2.0x2, 1G EMAC x3, UART x2, serial peripheral interface (SPI) x4, I2C x5, general purpose timers x7, watchdog timer x4	
R-Tile Device Resources	R-Tile PCI Express* (PCIe*) hard IP blocks (Gen5x16) or bifurcateable 2X PCIe Gen5 x8 (EP) or 4X Gen5 x4 (RP)	3	3
	Compute Express Link (CXL) lanes	48	48
F-Tile Device Resources	F-Tile PCIe hard IP blocks (Gen4x16) or bifurcateable 2X PCIe Gen4 x8 (EP) or 4X Gen4 x4 (RP)	3	3
	F-Tile high-speed transceiver channel count PAM4 (up to 112 Gbps) - RS and KP forward error correction (FEC) Non-return-to-zero (NRZ) (up to 56 Gbps)	8x PAM-4 8x NRZ	8x PAM-4 8x NRZ
	F-Tile general-purpose transceiver channels count PAM4 (up to 58 Gbps) - RS and KP FEC NRZ (up to 32 Gbps)	48x PAM-4 64x NRZ	48x PAM-4 64x NRZ
	F-Tile 10/25/50/100/200/400G Ethernet MAC + FEC hard intellectual property (IP) blocks	2	2
GPIO (LVDS) / F-Tile 32 Gbps (58 Gbps) / High-Speed Transceiver 58 Gbps (112 Gbps)			
Package Options	R3343A (F-Tile x4) (59 mm x 53 mm, Hex 1.0 mm pitch)	768(384) /64(48)/ 8(8)	768(384) /64(48)/ 8(8)
GPIO (LVDS) / F-Tile 32 Gbps (58 Gbps) / High-Speed Transceiver 58 Gbps (112 Gbps) / R-Tile 32 Gbps PCIe (CXL) Lanes			
Package Options	R2979A (F-Tile and R-Tile x3) (57.5 mm x 49 mm, Hex 1.0 mm pitch)	768(384)/16(12)/0(0)/48(48)	768(384)/16(12)/0(0)/48(48)
	R3803A (F-Tile x3 and R-Tile) (60 mm x 59 mm, Hex 1.0 mm pitch)	1104(552)/48(36)/8(8)/16(16)	1104(552)/48(36)/8(8)/16(16)

AES-256/SHA-256 bitstream encryption

Quad-core 64 bit Arm* Cortex*-A53 up
management unit, cache coheren

https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/pt/intel-agilex-i-series-product-table.pdf

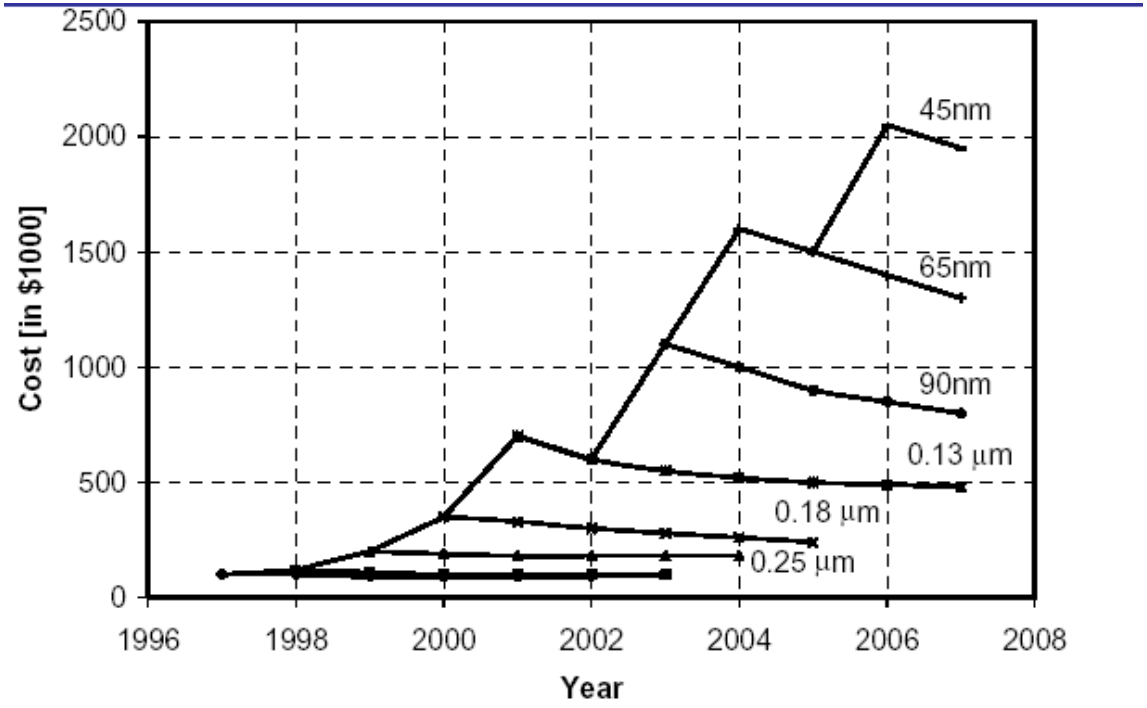
Comparação

Parameter	FPGA*	Structured ASIC	Standard-cell ASIC
Area	40	10	1
Speed	4.5	1.5	1
Power consumed	12	2	1
Unit costs	High	Medium	Low (high V)
NRE cost	Low	Medium	High
Time-to-market	Low	Low-Medium	High
Reconfigurability	Full	No	No
Market Volume	Low-medium	Medium	High

Mask Costs

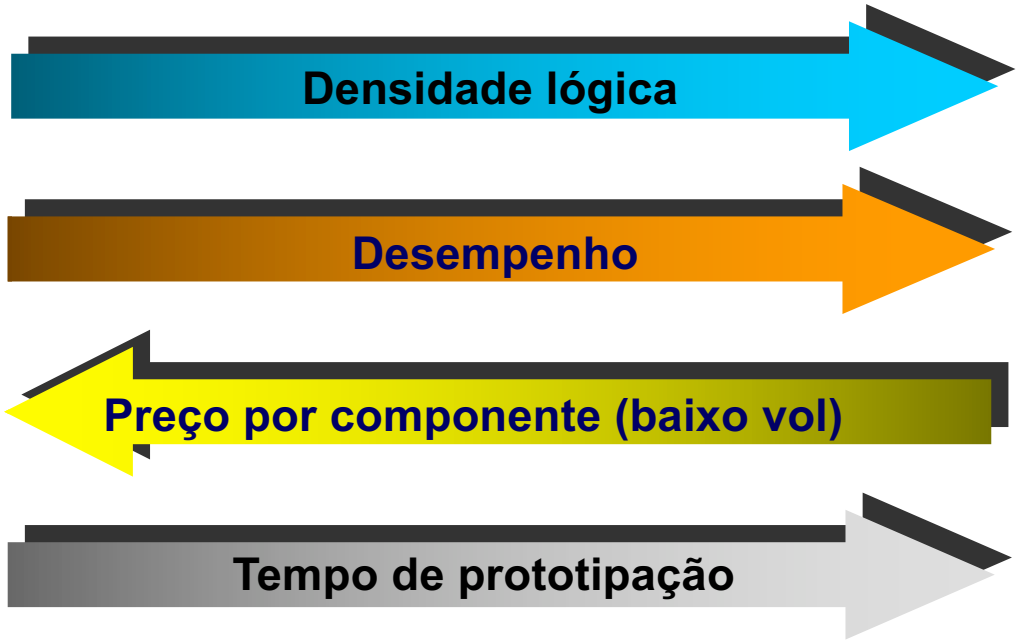
ASIC é vantajoso, mas muito caro!

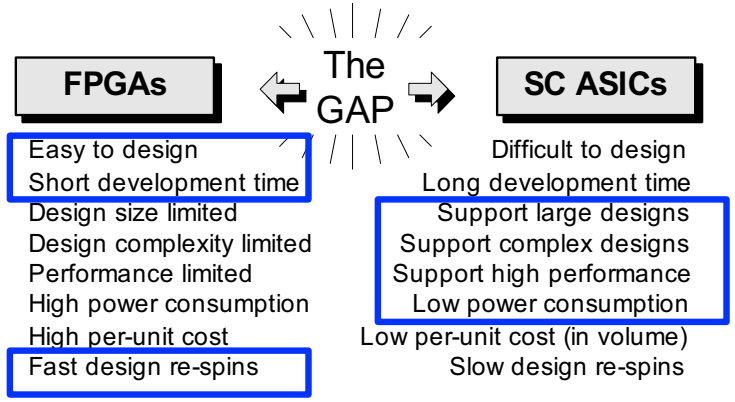
Process (microns)	2.0	0.8	0.6	0.35	0.25	0.18	0.13	0.1
Single Mask Cost ($K)	1.5	1.5	2.5	4.5	7.5	12	40	60
# of Masks	12	12	12	16	20	26	30	34
Mask Set cost ($K)	18	18	30	72	150	312	1000	2000



- A full set of lithography masks can cost between US$1-3M.
- Roughly 25% reduction in ASIC design starts in past 7 years. [Sematech Annual Report 2002], [A. Sangiovanni-Vincentelli “The Tides of EDA”, keynote talk, DAC 2003].
- Need an approach in which different designs share a set of masks

Conclusão





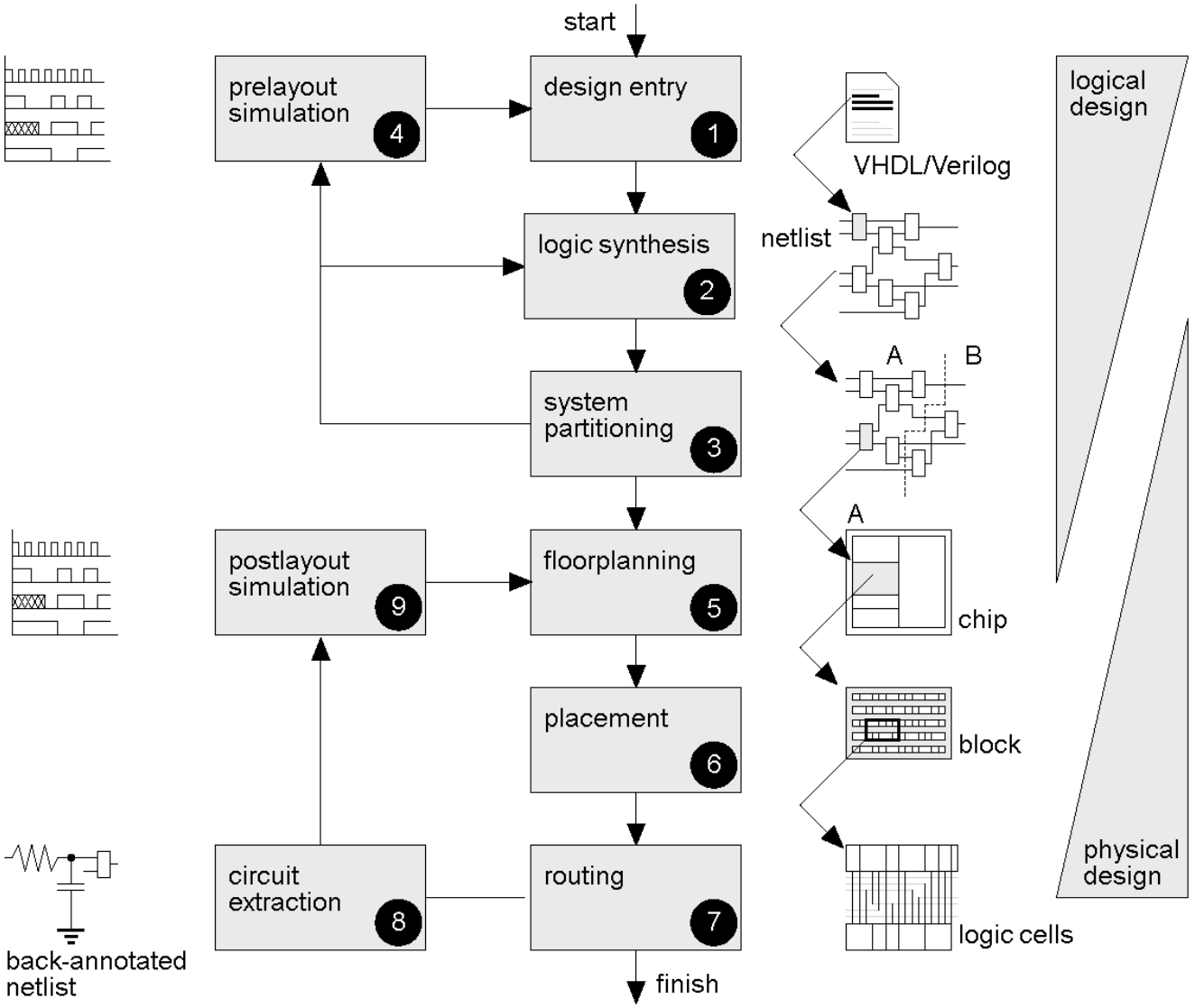
Componentes Programáveis (FPGAs)

Pré-difundidos

Full-custom, Standard-cells e Geração automática

Design Flow

1. **Design entry** - Using a hardware description language (HDL) or schematic entry
2. **Logic synthesis** - Produces a netlist - logic cells and their connections
3. **System partitioning** - Divide a large system into ASIC-sized pieces
4. **Prelayout simulation** - Check to see if the design functions correctly
5. **Floorplanning** - Arrange the blocks of the netlist on the chip
6. **Placement** - Decide the locations of cells in a block
7. **Routing** - Make the connections between cells and blocks
8. **Extraction** - Determine the resistance and capacitance of the interconnect
9. **Postlayout simulation** - Check to see the design still works with the added loads of the interconnect



Figures from Application-Specific Integrated Circuits, Michael John Sebastian Smith, Addison Wesley, 1997

1. Design entry (1/3)

VHDL, Verilog

“think hardware!”:

mesmo que esquemático

seja passado, reconhecer

no VHDL estruturas como

FSMs, decoders, mux,

registradores...

```
--
process( Bus2IP_Clk ) is
  begin

    if Bus2IP_Clk'event and Bus2IP_Clk = '1' then

      if Bus2IP_Reset = '1' then
          slv_reg(0) <= (others => '0');
          slv_reg(1) <= (others => '0');
          slv_reg(2) <= (others => '0');
          slv_reg(3) <= (others => '0');
          slv_reg(4) <= (others => '0');
          slv_reg(5) <= (others => '0');
          slv_reg(6) <= (others => '0');
          slv_reg(7) <= (others => '0');
          slv_reg(8) <= (others => '0');
          slv_reg(9) <= (others => '0');

      elsif reseta_bit_slv_reg0='1' then
          slv_reg(9) <= (others => '0');

      else
             case Bus2IP_WrCE(0 to 14) is
               when "100000000000000" => slv_reg(0) <= Bus2IP_Data;
               when "010000000000000" => slv_reg(1) <= Bus2IP_Data;
               when "001000000000000" => slv_reg(2) <= Bus2IP_Data;
               when "000100000000000" => slv_reg(3) <= Bus2IP_Data;
               when "000010000000000" => slv_reg(4) <= Bus2IP_Data;
               when "000001000000000" => slv_reg(5) <= Bus2IP_Data;
               when "000000100000000" => slv_reg(6) <= Bus2IP_Data;
               when "000000010000000" => slv_reg(7) <= Bus2IP_Data;
               when "000000001000000" => slv_reg(8) <= Bus2IP_Data;
               when "000000000100000" => slv_reg(9) <= Bus2IP_Data;
               when others => null;
             end case;
     end if;

    end if;
  end process;
```

fernando.moraes@pucrs.br

1. Design entry (2/3)

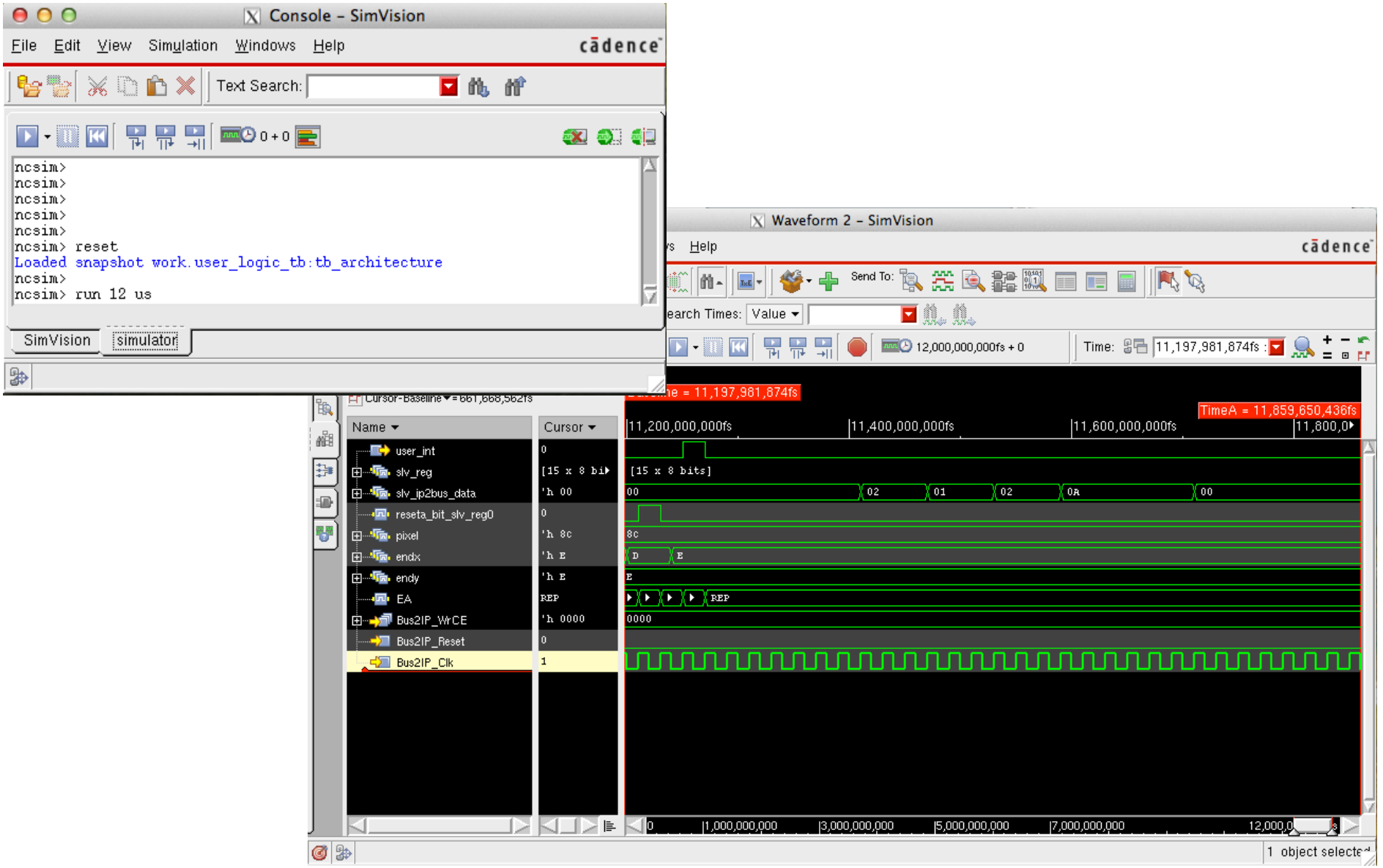
fernando.moraes@pucrs.br

Simulação funcional

- permite verificar a funcionalidade do circuito
- erros de VHDL comuns:
  - duplo driver
  - lista de sensitividade incompleta
  - latch inferida
  - lógica com o sinal de clock
  - lógica combinacional controlada por sinal de clock
- **muito comum** uma simulação funcional correta falhar após a síntese

1. Design entry (3/3)

Exemplo de simulador: incisive (CADENCE)



fernando.moraes@pucrs.br

2. Logic synthesis (1/3)

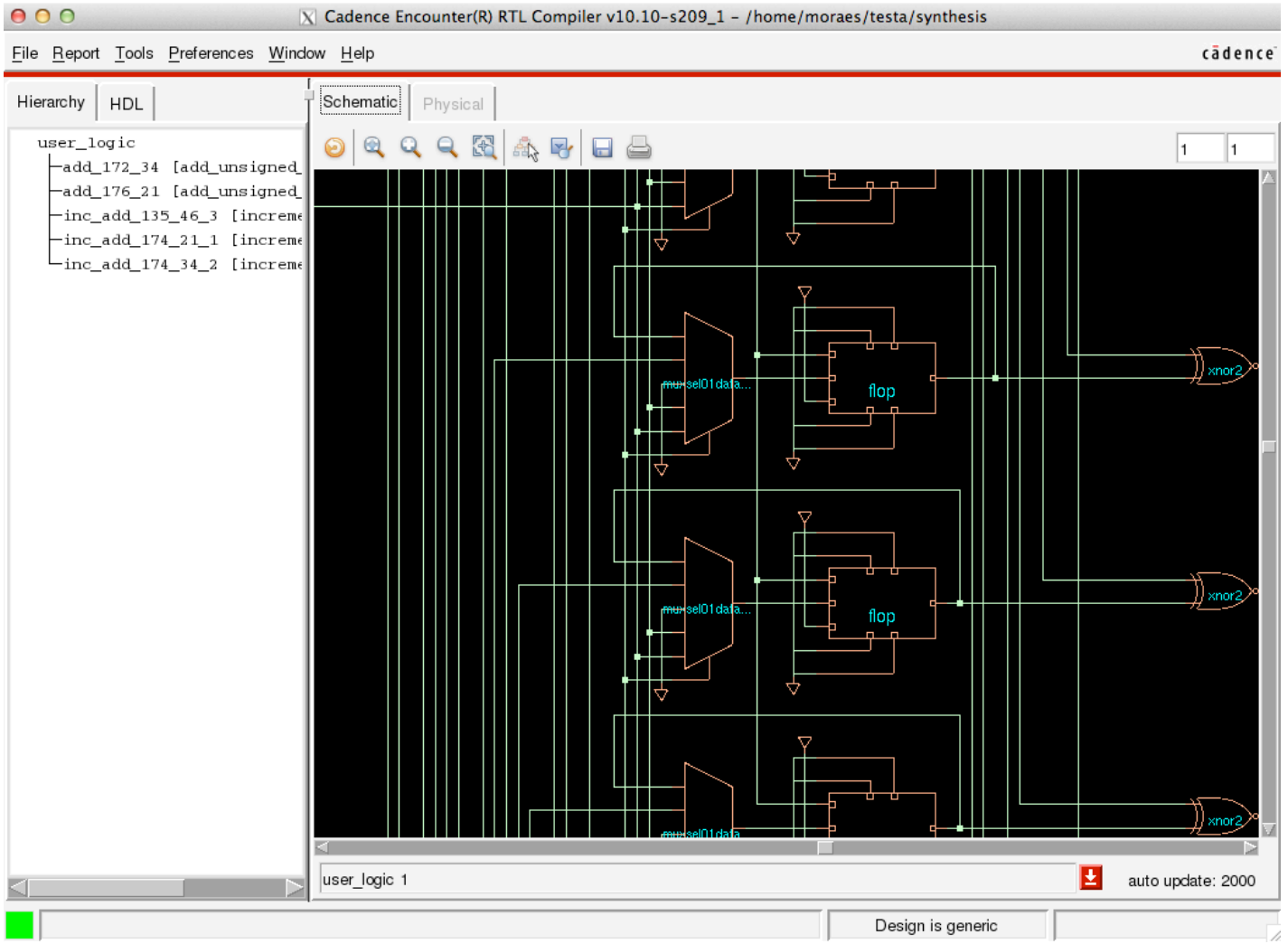
- Transforma o VHDL/Verilog em um netlist mapeado para uma dada tecnologia
- Importante nesta etapa: restrições de projeto

```
create_clock -name {Bus2IP_Clk} -period 2.0 [get_ports {Bus2IP_Clk}]

set_false_path -from [get_ports {Bus2IP_Reset}]

## INPUTS
set_input_delay -clock Bus2IP_Clk -max 0.2 [all_inputs]
set_input_transition -min -rise 0.003 [all_inputs]
set_input_transition -max -rise 0.16 [all_inputs]
set_input_transition -min -fall 0.003 [all_inputs]
set_input_transition -max -fall 0.16 [all_inputs]


## OUTPUTS
set_load -min 0.0014 [all_outputs]
set_load -max 0.32 [all_outputs]
```

fernando.moraes@pucrs.br

2. Logic synthesis (2/3)

- Exemplo de ferramenta: Encounter (Cadence)
- Utilização por script



fernando.moraes@pucrs.br

2. Logic synthesis (3/3)

- Relatório de área

```
 Instance              Cells  Cell Area  Net Area
-------------------------------------------------
-
busca_padrao           1712       9540        23
  inc_add_100_45_3       18         88         0
  add_242_66             16         66         0
  add_241_54             16         66         0
  inc_add_243_47_2        7         34         0
  inc_add_239_54_1        7         34         0
```

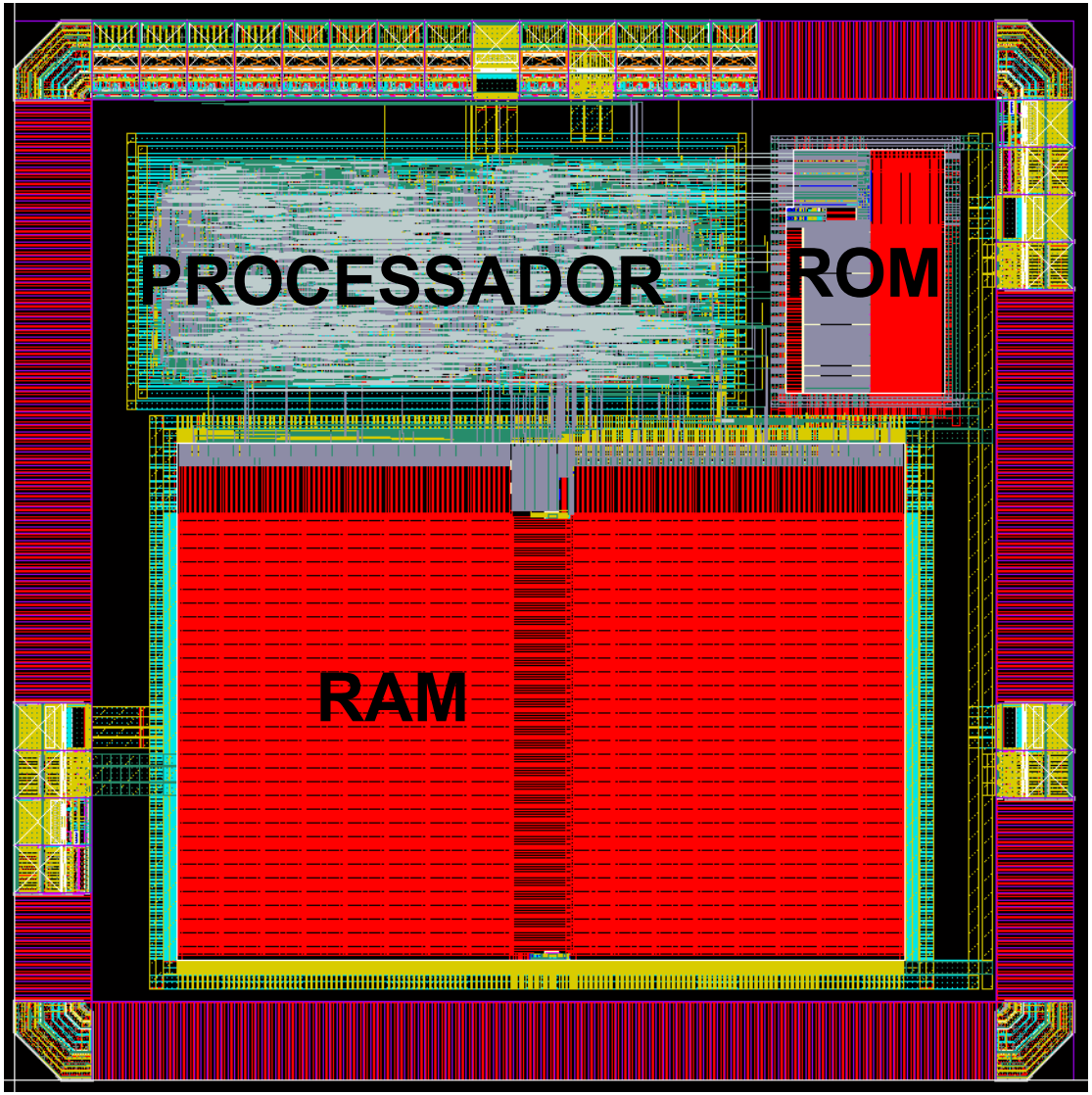
- Relatório de atraso

```
 Pin                    Type                Fanout Load Slew Delay Arrival
                                                   (fF) (ps)  (ps)    (ps)
-------------------------------------------------------------------------
(clock Bus2IP_Clk)      launch                                        1000 F
address_reg[7]/clk                                    0               1000 R
address_reg[7]/q   (u)  unmapped_d_flop        7 41.3    0   +90      1090 R
...
...
g2659/z            (u)  unmapped_not           1  5.9    0   +11      1710 R
EA_reg[1]/d             unmapped_d_flop                    +0      1710
EA_reg[1]/clk           setup                              0   +47  1757 R
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(clock Bus2IP_Clk)      capture                                       2000 R
-------------------------------------------------------------------------
Cost Group   : 'Bus2IP_Clk' (path_group 'Bus2IP_Clk')
Timing slack :     243ps
Start-point  : address_reg[7]/clk
End-point    : EA_reg[1]/d
```

fernando.moraes@pucrs.br

3. System partitioning

Divide o circuito em módulos



fernando.moraes@pucrs.br

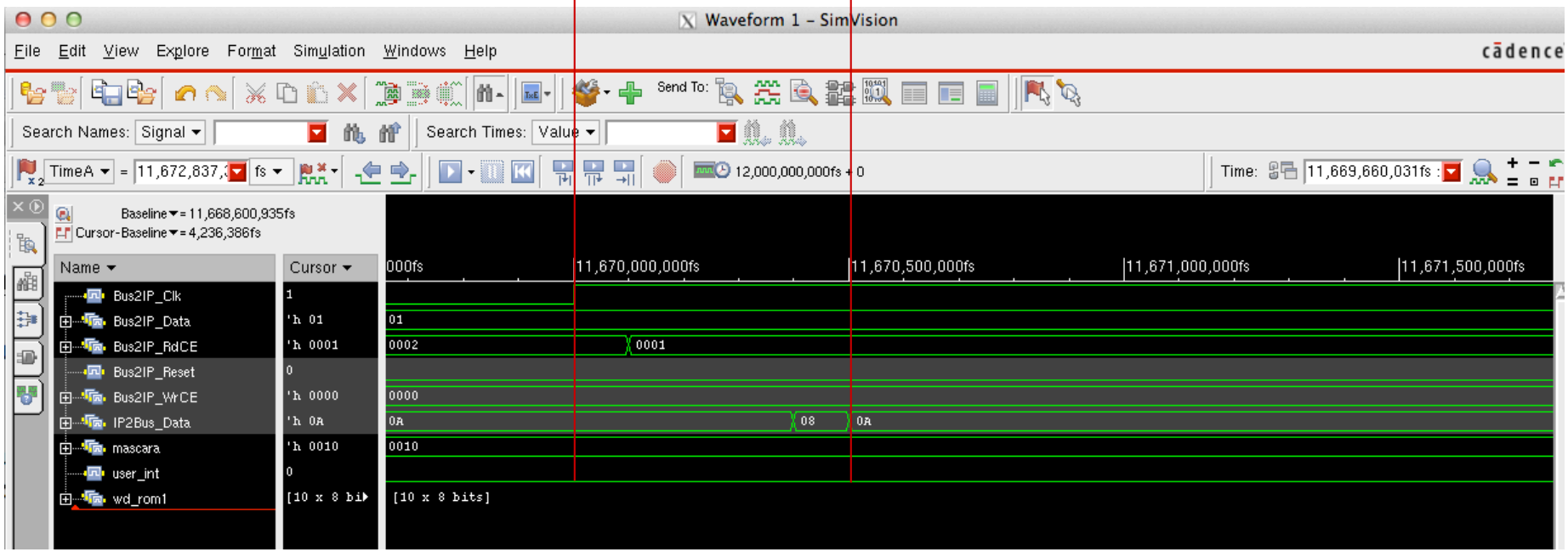
4. Prelayout simulation

Simula o netlist com a descrição das portas lógicas

Exemplo de script:

```
-smartorder -work work -V93 -top user_logic_tb -gui -access +rw
/soft64/design-kits/stm/65nm-cmos065_536/CORE65GPSVT_5.1/behaviour/verilog/CORE65GPSVT.v
/soft64/design-kits/stm/65nm-cmos065_536/CLOCK65GPSVT_3.1/behaviour/verilog/CLOCK65GPSVT.v
../../synthesis/layout/busca_padrao.v
../tb/tb_padrao.vhd
```

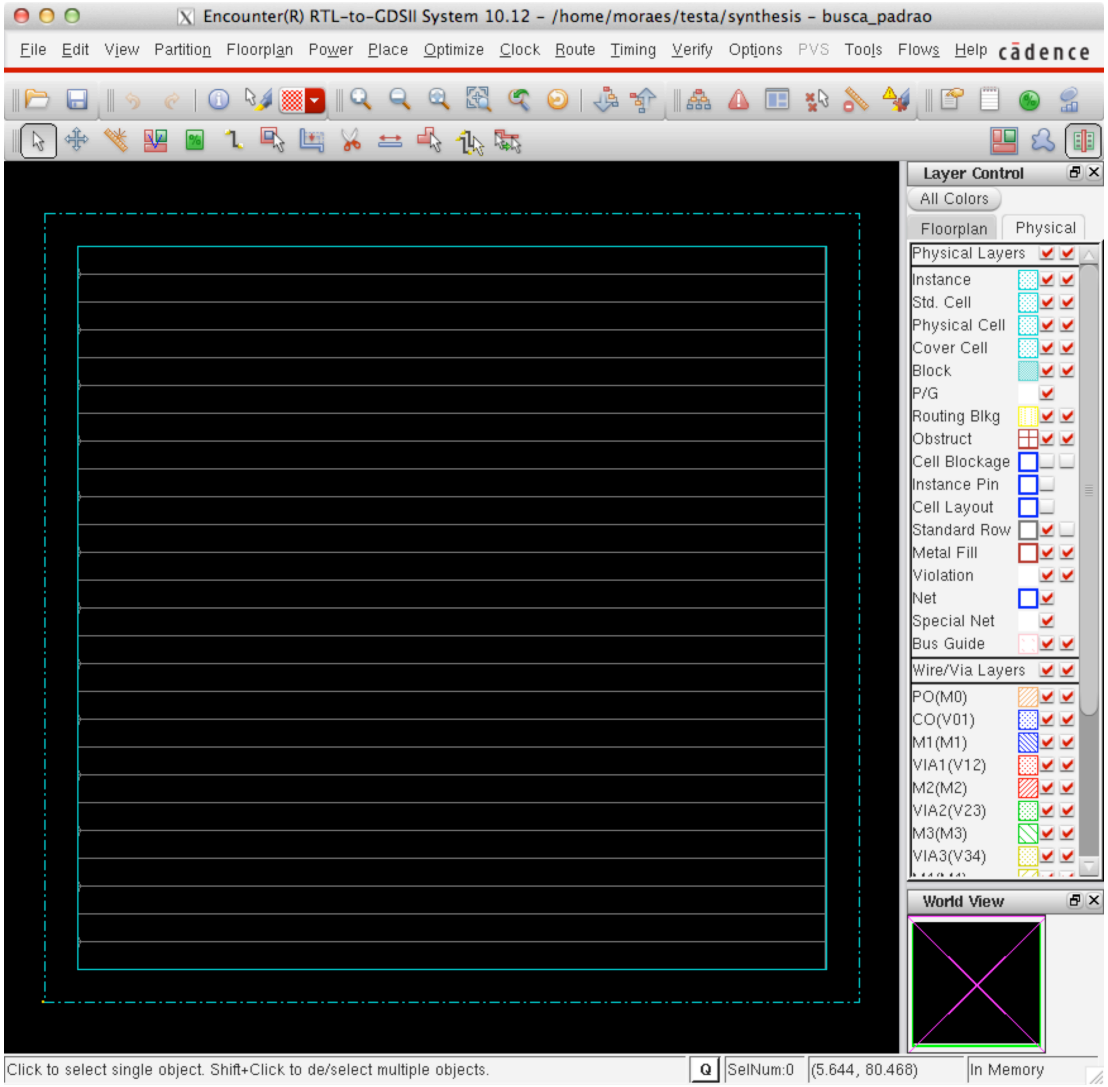
Atraso de porta lógica



fernando.moraes@pucrs.br

5. Floorplanning

Define a forma dos blocos standard cells e o posicionamento dos blocos “fixos”



Prever espaço para roteamento (density +- 0,8)

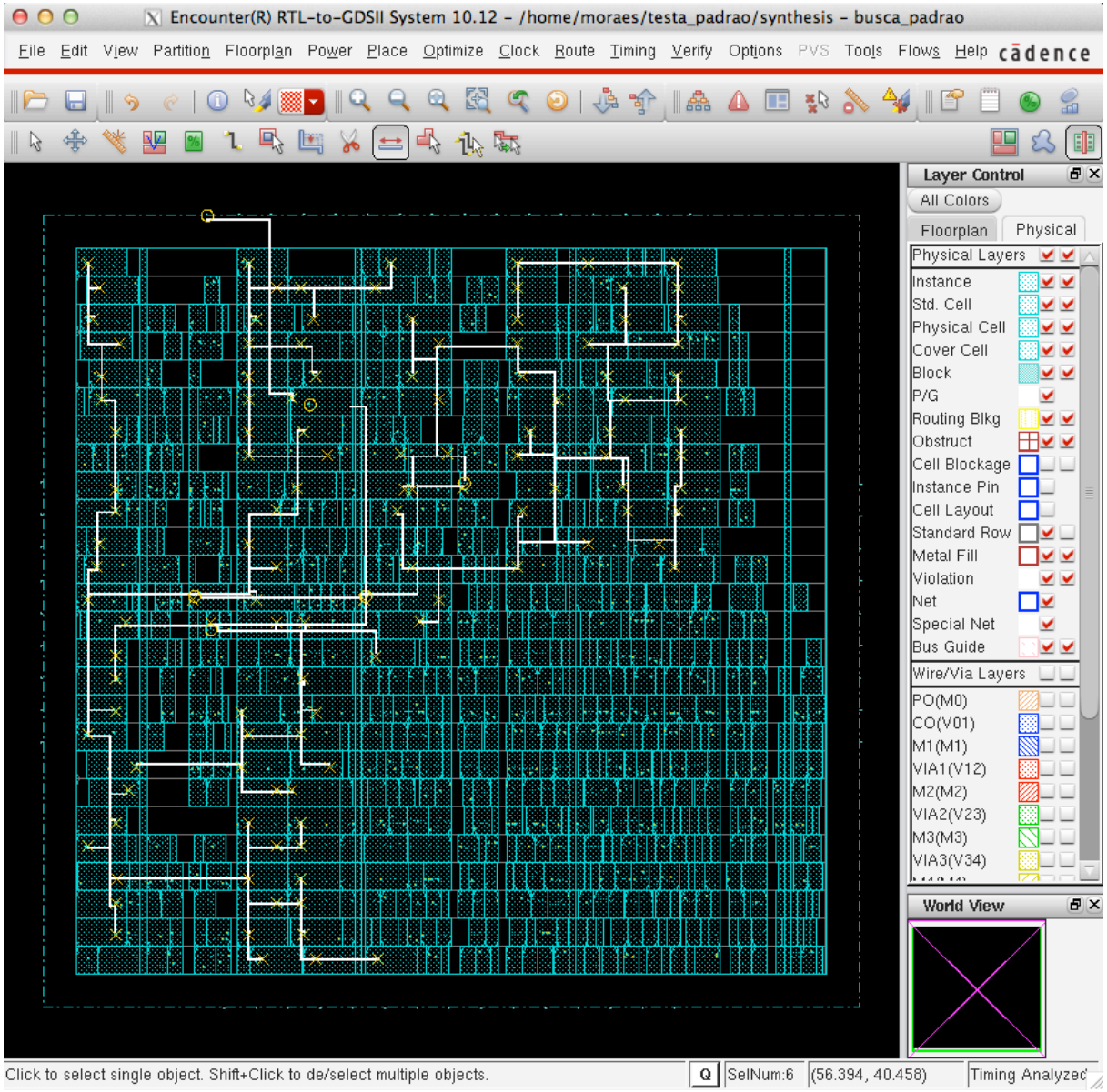
fernando.moraes@pucrs.br

6. Placement (1/2)

Posicionamento das células

Etapas importantes neste processo:

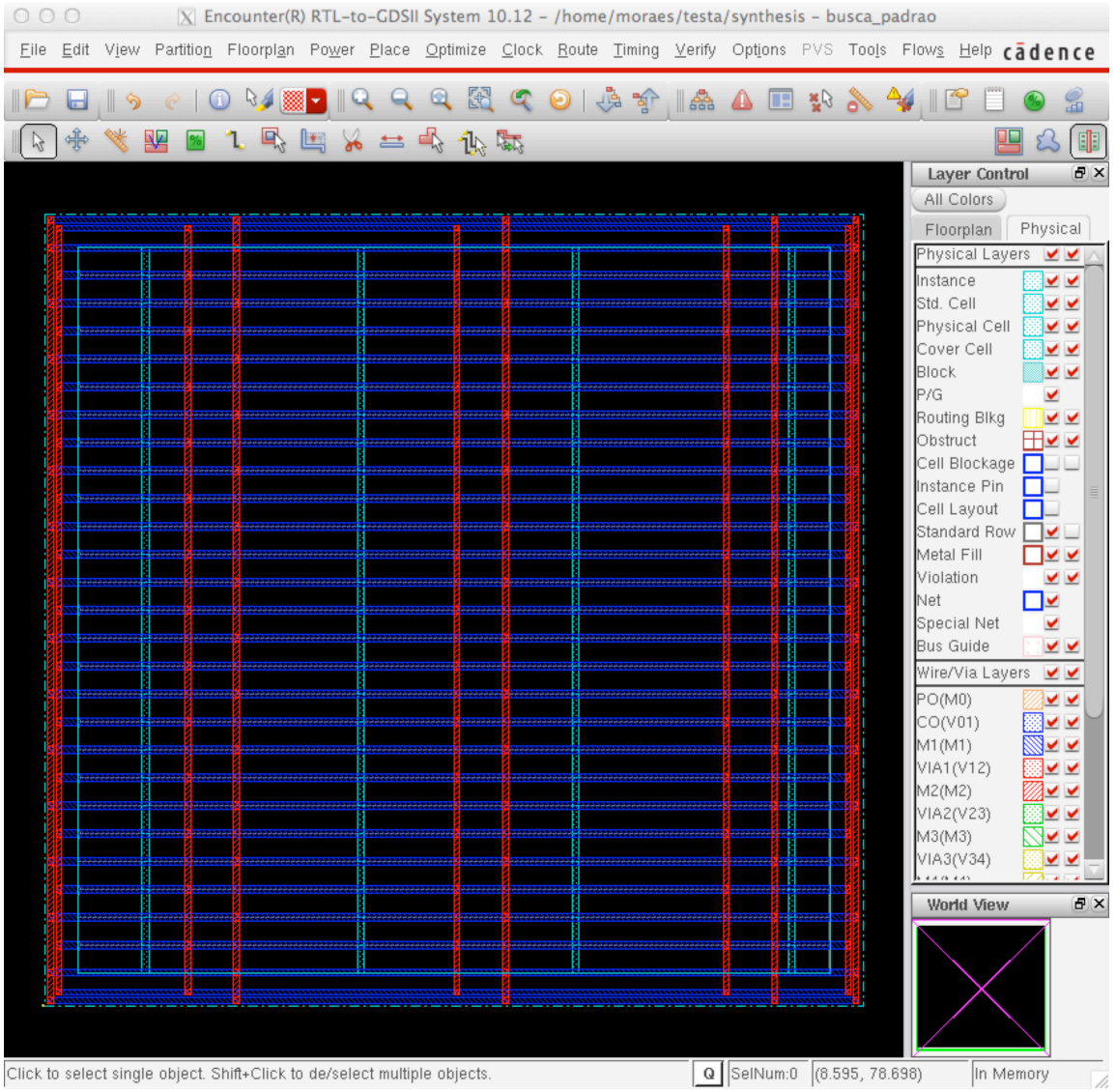
- árvore de clock (em destaque)
- alimentação



6. Placement (2/2)

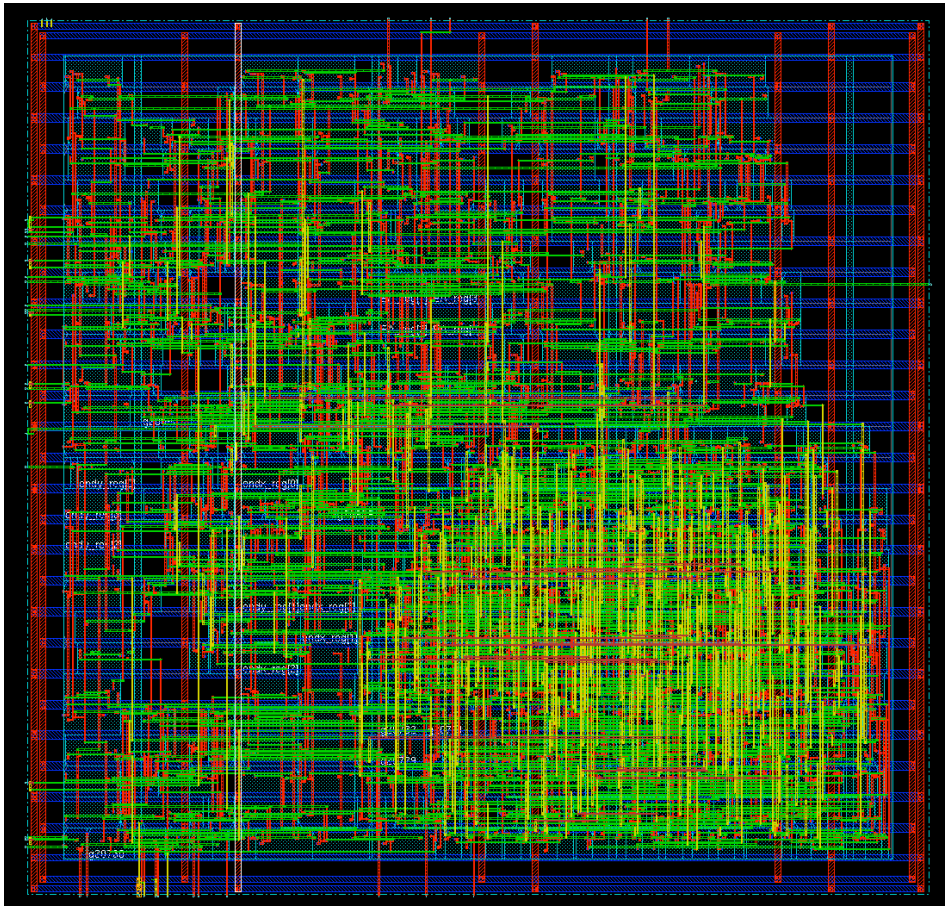
Alimentação do bloco

- Evita ruídos nas linhas de alimentação (IR drop)

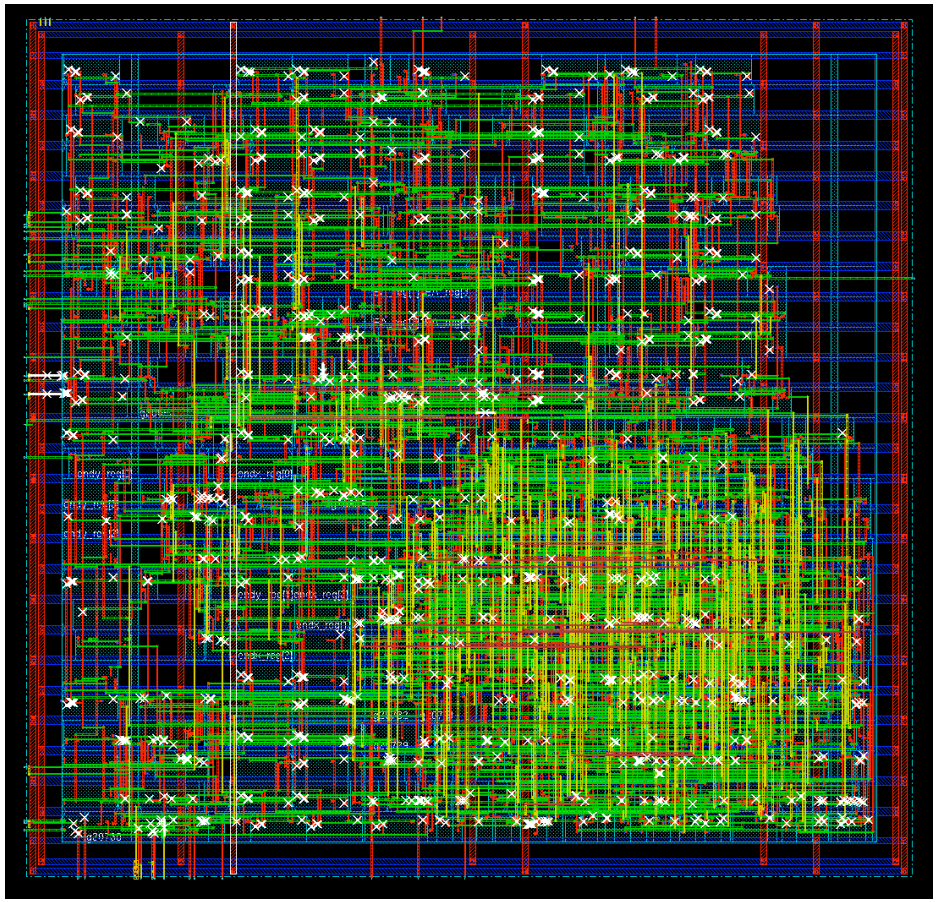


fernando.moraes@pucrs.br

7. Routing (1/2)

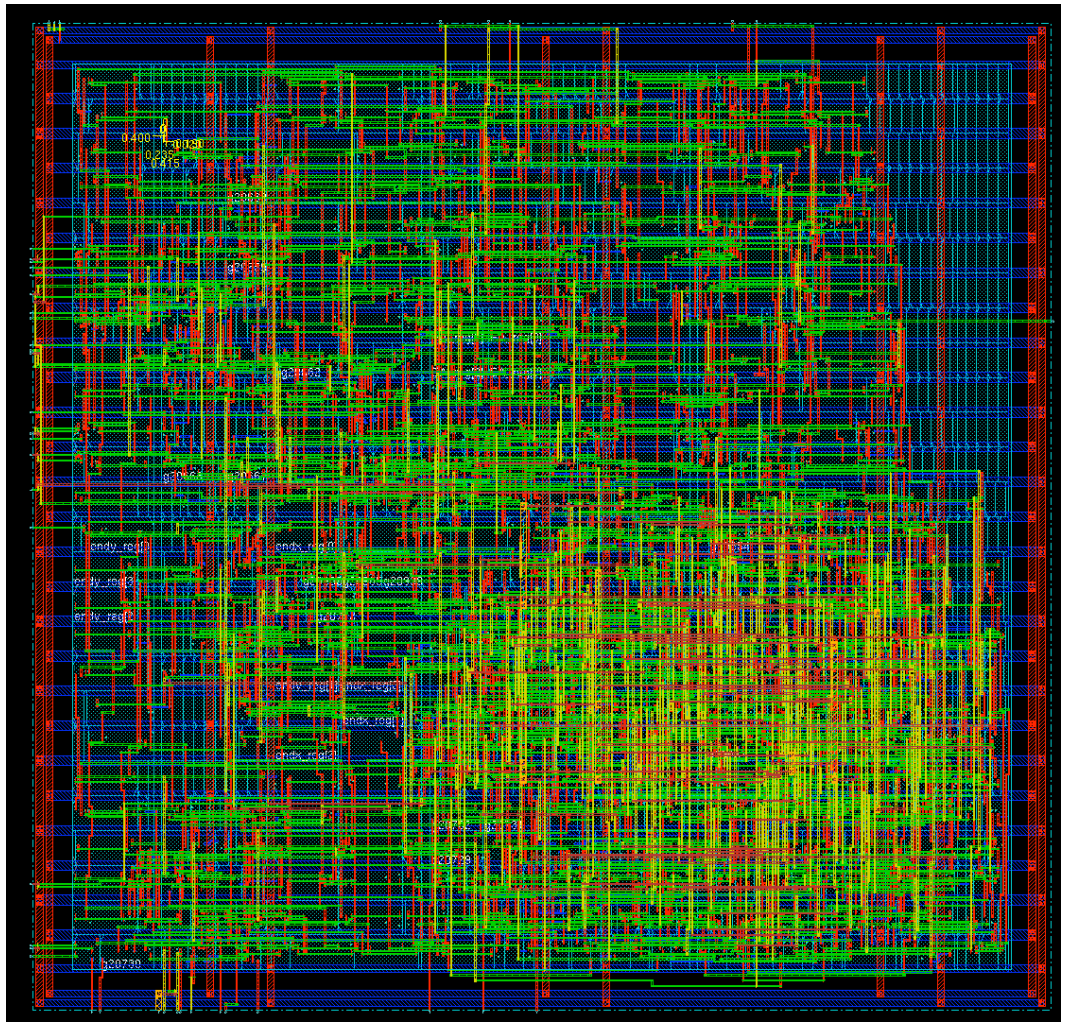


Roteamento inicial



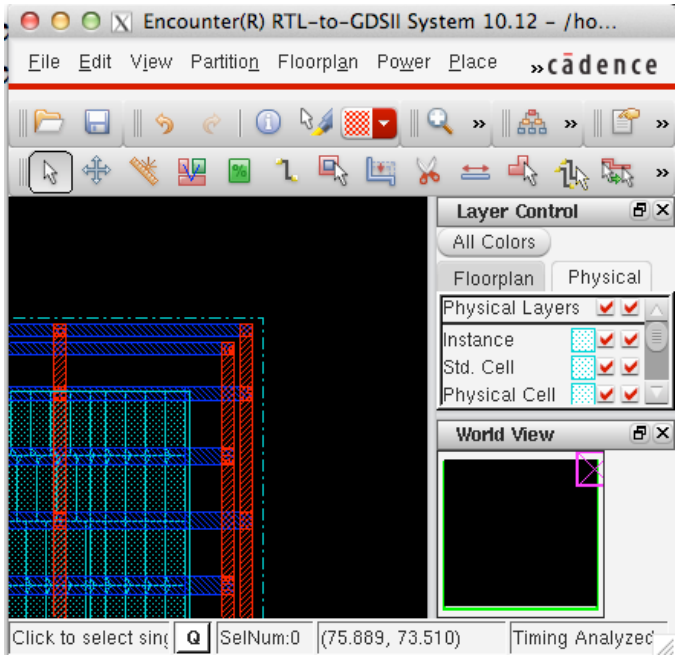
Verificação de DRC e timing

7. Routing (2/2)



Detalhes:

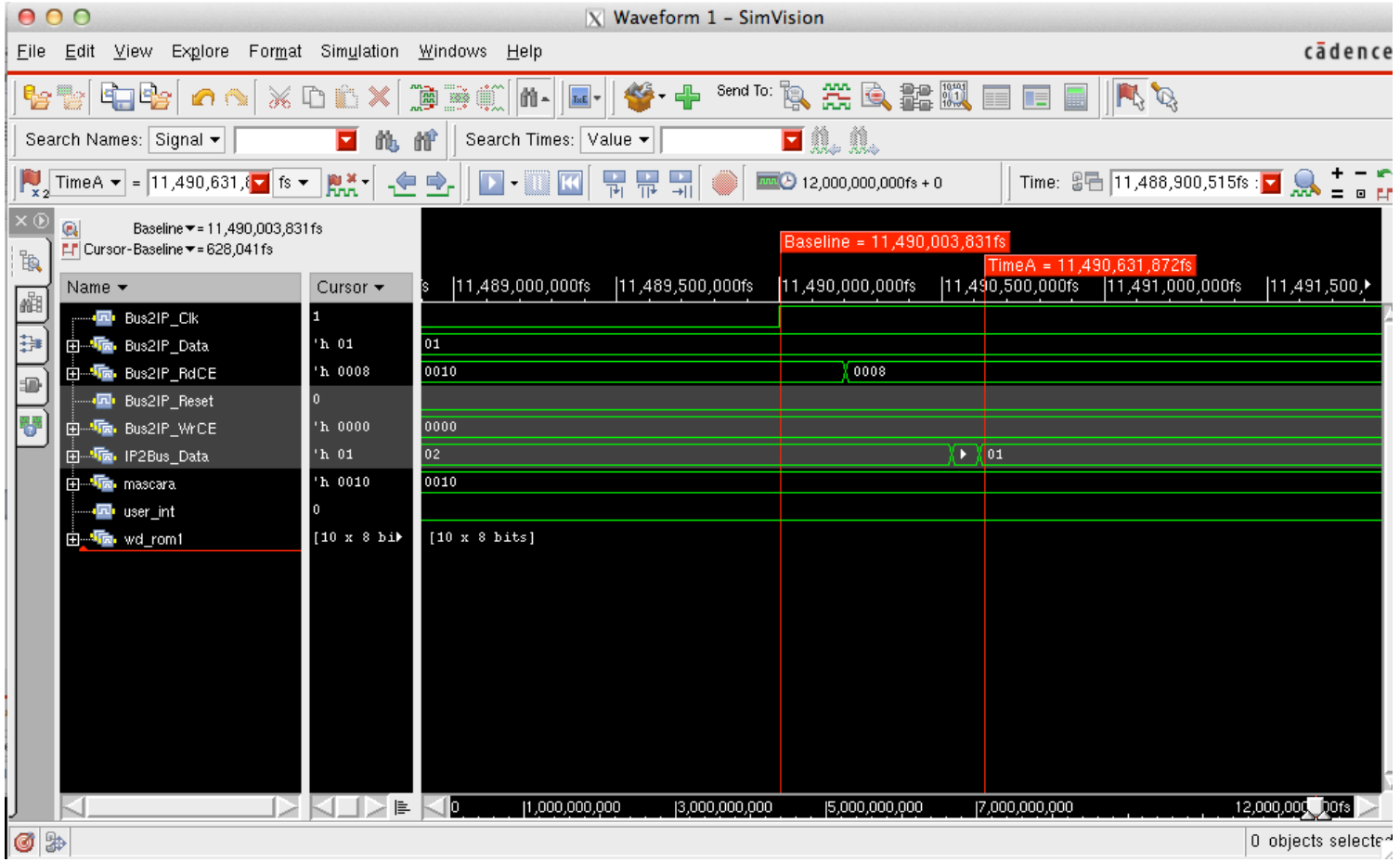
- filler cells
- alimentação



Roteamento final

8/9. Extraction and postlayout simulation

- Arquivo com as capacitâncias de roteamento
  - Formato SDF
- Permite obter uma estimativa precisa do atraso do circuito



fernando.moraes@pucrs.br