

# TF – TRABALHO FINAL – REALIZAR TODAS AS ETAPAS DO LAB4 PARA UMA DAS FUNÇÕES ABAIXO

## 26/novembro/25 (QUA) – apresentação oral ao professor

Cada grupo deve implementar a célula de uma porta complexa, conforme abaixo:

$F_1 = \overline{A} \cdot (\overline{B} \cdot C + D)$	micro 1: Matheus Susin Timmers - Pedro Henrique De Ros
$F_2 = \overline{((A + B) \cdot C)} + D$	micro 2: Lukas Da Silva Ely - Bruno Becker Silva
$F_3 = \overline{A} + \overline{B} + \overline{(C \cdot D)}$	micro 3: Artur Costa Ribeiro - Carlos Cardozo Olmes
$F_4 = \overline{A} \cdot \overline{B} \cdot (C + D)$	micro 4: Henrique Luis Neves Duarte - Mikael Tolotti da Silva
$F_5 = \overline{A} \cdot (\overline{B} + \overline{C} + \overline{D})$	micro 5: Bruno Fetter Zatar - Douglas Cavalheiro Carvalho

### CONTEÚDO DO RELATÓRIO A SER APRESENTADO AO PROFESSOR

- (10%) Esquemático – *print screen* da tela e do console com mensagem se houve ou não erro no esquemático.
- (35%) Layout da célula – *print screen* da tela, indicando as entradas A / B / C / D e a saída F.
- (10%) Relatório do DRC e LVS – telas que mostrem a execução. Não haverá os erros de DCO para estas funções.
- (20%) Extração e Simulação elétrica. Apresentar os arquivos referentes ao *netlist* e às **capacitâncias parasitas**. Dado que é uma porta com 4 entradas, deve-se gerar 16 estímulos para a verificação da funcionalidade completa da porta projetada, como abaixo:

```
v1 a 0 pulse (1 0 0 0.02N 0.02N 1N 2N)
v2 b 0 pulse (1 0 0 0.02N 0.02N 2N 4N)
v3 c 0 pulse (1 0 0 0.02N 0.02N 4N 8N)
v4 d 0 pulse (1 0 0 0.02N 0.02N 8N 16N)
```

**Apresentar a tabela verdade da função e conferir com a simulação.**

Exemplo: <https://programmingdojo.net.nz/study/truth-table-generator/index>

- (5%)  
**LEF:** layout da view abstract. *Print screen* da tela da view abstract.  
**LIB:** geração do arquivo de caracterização elétrica. Apresentar a página de caracterização.
- (10%) Síntese lógica. Sintetizar um circuito simples, negando as entradas e a saída da porta complexa. Deve-se ter os arquivos LEF e o LIB do inversor e da função gerada (**tf**), alterando-se os *scripts* necessários. Apresentar o esquemático gerado pela ferramenta de síntese. Utilizar por exemplo o código SystemVerilog abaixo (estou assumindo que o nome da célula implementada é **tf**), nomeando o arquivo como **final.sv**.

```
module final (
    input logic A, B, C, D,
    output logic Z
);
    logic not_a, not_b, not_c, not_d, not_s;

    inv INVA (.A(A), .Z(not_a));
    inv INVB (.A(B), .Z(not_b));
    inv INVC (.A(C), .Z(not_c));
    inv INVD (.A(D), .Z(not_d));
    tf COMPLEX (.A(not_a), .B(not_b), .C(not_c), .D(not_d), .Z(not_s));
    inv INVE (.A(not_s), .Z(Z));

endmodule
```

→ Alterem o *load\_tcl* (**set DESIGN\_TOP "final"**) e o *cmd\_genus* (**read\_hdl -sv final.sv**)

7. (10%) Síntese física – apresentar o layout para o circuito **final**, e o relatório de DRC. No script de síntese física posicionar os pinos de entrada e saída corretamente (estes nomes devem corresponder ao código SystemVerilog). Por exemplo:

```
# Step 4: Place – pode-se também usar apenas o comando: : assign_io_pin
place_design -no_pre_place_opt
edit_pin -side Top -layer 2 -spread_type center -spacing 3 -pin {A B C D}
edit_pin -side Left -layer 2 -spread_type center -spacing 3 -pin {F}
```

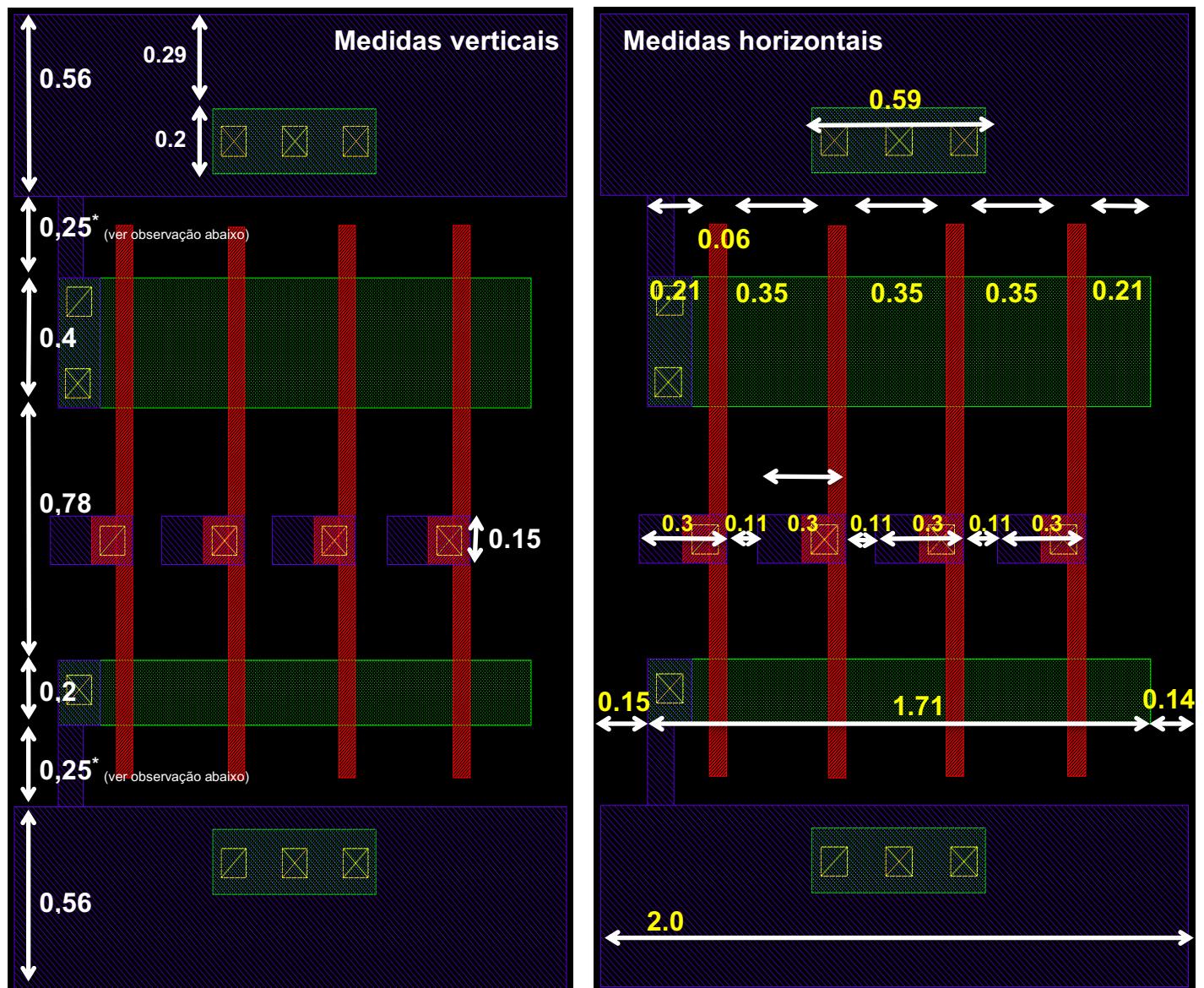
## PASSOS PARA O DESENHO – MODELO PARA CÉLULAS COM 4 ENTRADAS

- Desenhar a célula na mesma biblioteca do inverter, pois esta célula será integrada com o inverter.



- $W_N = 0.2 \mu\text{m} / W_P = 0.4 \mu\text{m}$
- As “cabeças de contato” devem ter um metal1 para a correta inserção dos pinos, como mostrado abaixo ( $0.15 \mu\text{m} \times 0.3 \mu\text{m}$ ), sobre as linhas polisilício. A posição exata das “cabeças de contato” sobre o polisilício vai depender das conexões necessárias – ver o exemplo da NAND4 na página 4 deste documento.

- Desenhar no editor de layout o modelo abaixo, usando os níveis de área ativa (OD), polisilício (PO), metal1 (M1), e contato (CO). Respeitar a altura de  $3 \mu\text{m}$ , com largura múltipla de  $0.2 \mu\text{m}$ . Alguns metais/contatos foram inseridos sobre os drenos/sources a título de exemplo.

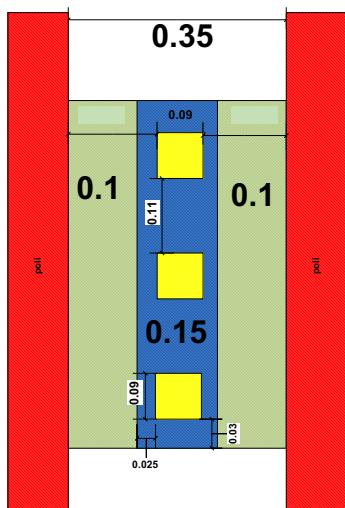


MEDIDA VERTICAL TOTAL:  $0,56 + 0,25 + 0,2 + 0,78 + 0,4 + 0,25 + 0,56 = 3 \mu\text{m}$

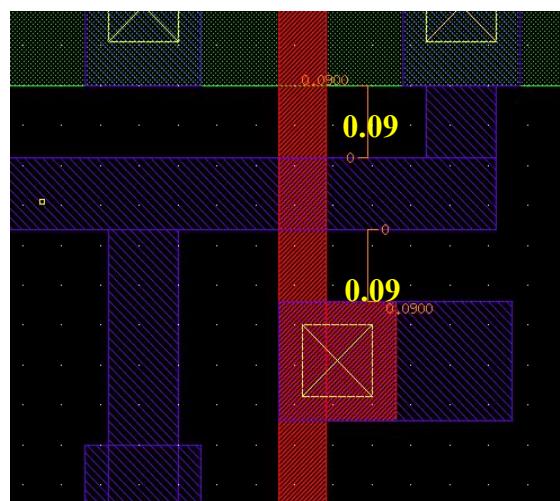
MEDIDAS NA DIFUSÃO (camada OD):  $2 * \text{extremidades} + 4 * \text{largura do poli} + 3 * \text{espaçamento entre polis} = (2 * 0,21) + (4 * 0,06) + (3 * 0,35) = 1,71 \mu\text{m}$

MEDIDA HORIZONTAL TOTAL:  $0,15 + 1,71 + 0,14 = 2 \rightarrow \text{observem o } 0,15 \mu\text{m} (\text{esquerda}) \text{ e } 0,14 \mu\text{m} (\text{direita})$

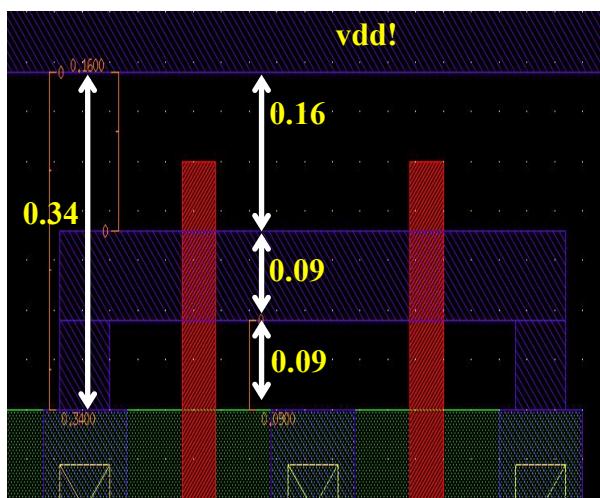
Exemplo de medidas para o contato entre o dreno e source:



Detalhe de uma conexão de metal1 abaixo da linha de difusão P:



O template original é para as conexões entre as linhas de difusão. Em função da célula pode ser necessário conectar entre vdd!/difusão ou gnd!/difusão. Para isto mudar o espaçoamento de 0,25 para 0,34 entre a difusão e a linha de alimentação, como abaixo.



Entre vdd/difusão P



Entre gnd/difusão N

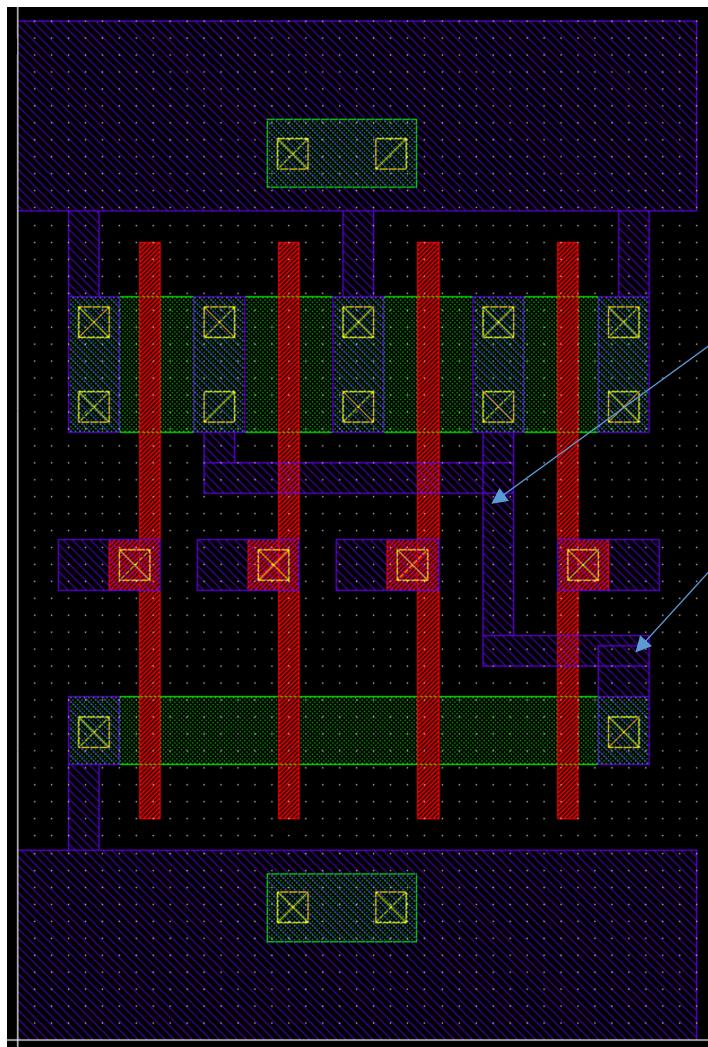
- Colocar os implantes (NP e PP), o poço N (NW) e ao redor de todo o desenho indicar a fronteira da célula (`prBoundary + DCO`). As figuras da NAND4 ilustram esta organização.

#### Atenção:

- As conexões por justaposição (transistores em série) **não** precisam de contato (ver exemplo da NAND4).
- As camadas `prBoundary/DCO` devem ter o tamanho da área definida pelos implantes NP/PP ( $2,6 \mu\text{m} \times 2 \mu\text{m}$ ). Conferir se estas camadas têm  **$2,6 \mu\text{m}$  de altura** também no inversor.
- As camadas `prBoundary/DCO` devem sobrepor-se exatamente sobre o canto inferior esquerdo do PP (limite inferior) até o canto superior direito do NP (limite superior). Conferir no inversor
- Distância mínima entre difusão (transistores) e os implantes (NN/PP):  $0,16 \mu\text{m}$

## EXEMPLO: LAYOUT PARA NAND4

Camadas de METAL1 + DIFUSÃO (OD) + CONTATO (CO) + POLI (PO)

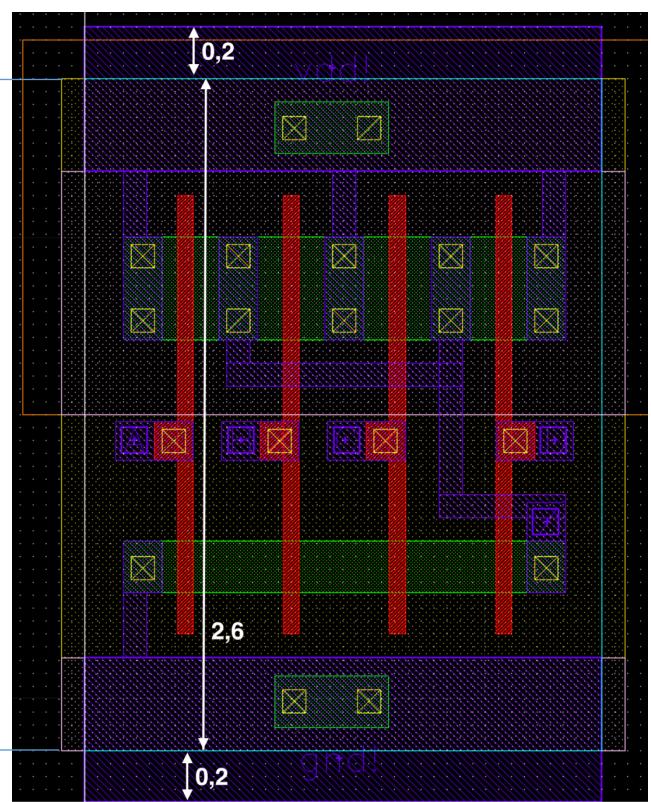
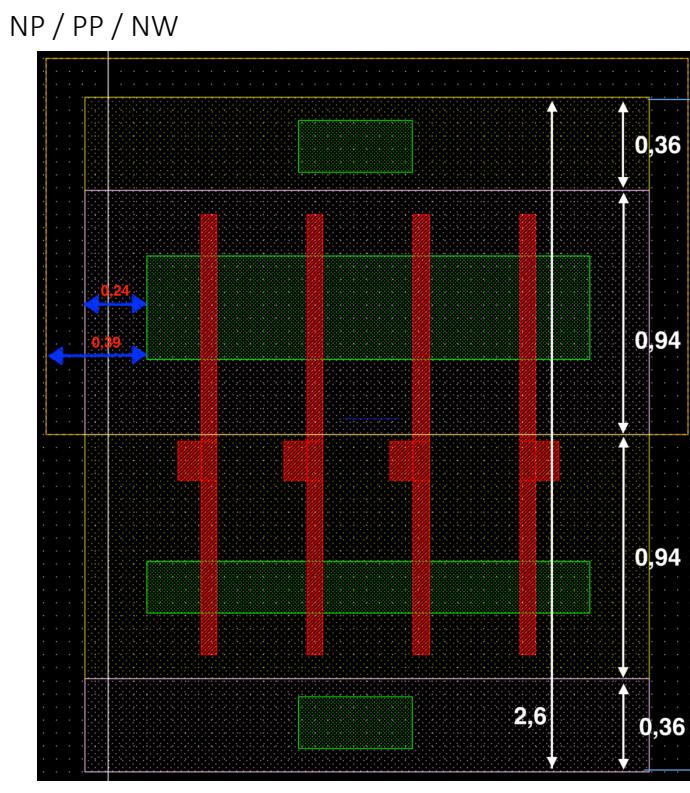


Observar o M1 entre o PDN e o PUN:  
os contatos foram afastados para  
permitir passar o M1

Quadrado de M1 para o pino de saída  
da NAND4

Altura do prBoundary + DCO: 2,6  $\mu\text{m}$

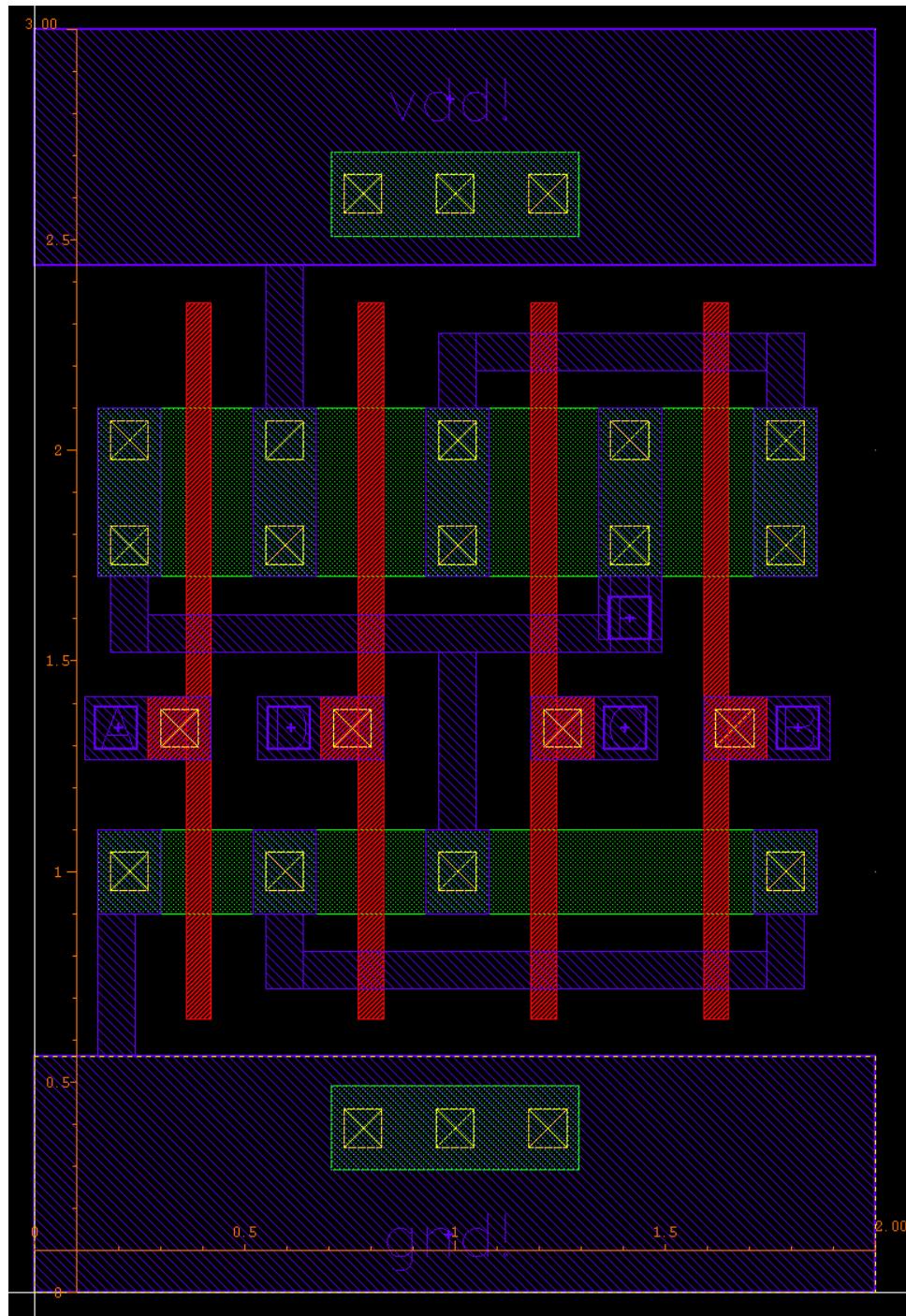
COMPLETO:



## Sem erros de DRC (não ocorrem mais os erros que ocorriam no INV)

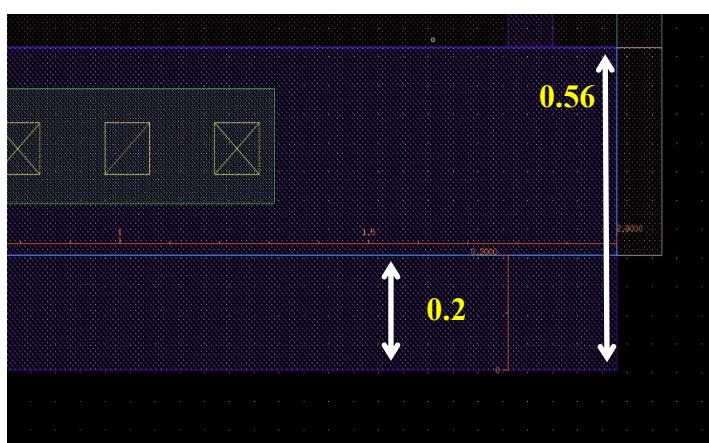


Exemplo de uma porta complexa xcom **roteamento entre difusão e alimentação**

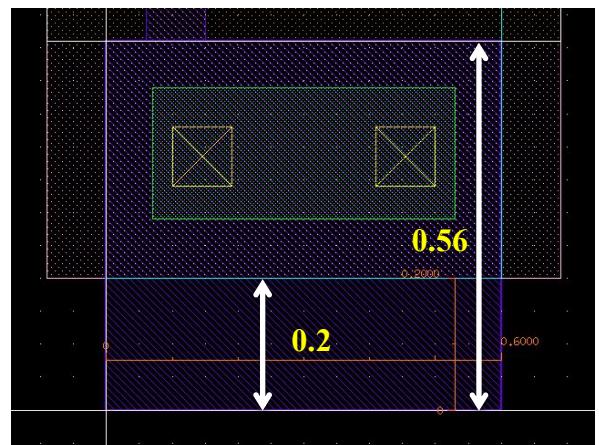


## ATENÇÃO

1. Lembrar que as células do **inversor** e da **porta lógica complexa** serão utilizadas em conjunto, portanto algumas características em comum são necessárias para que isto ocorra, como a altura ser a mesma ( $3 \mu\text{m}$ ), altura de vdd/gnd ser a mesma ( $0,56 \mu\text{m}$ ), e a altura do *DCO/prboundary* deve se  $2,6 \mu\text{m}$ .



Porta complexa (este trabalho)



inversor

2. Geração da view Abstract. Alterar o script para gerar a *view* da porta complexa onde há referência ao inversor.
3. Lembrar de alterar os scripts *cmd\_genus*, *cmd\_innovus* e *load.tcl* para apontar para os caminhos desejados, já que os originais apontam para os arquivos correspondentes ao lab4 (no diretório *synthesis*).
4. No arquivo *load.tcl* adicionar a biblioteca do arquivo LEF/LIB gerado da porta complexa junto com o LEF/LIB do inversor já feito no lab4, nas suas respectivas regiões do script.

```
set_db library "..../characterization/TF.lib ..../characterization/inv.lib /soft64/design-kits/stm/65nm-cmos065_537/CORE65GPSVT/5.2/libs/  
CORE65GPSVT_nom_1.00V_25C.lib"  
  
set_db lef_library "/..... /CORE65GPSVT.lef \  
/...../cmos065_7m4x0y2z_AP_Worst.lef \  
/soft64/design-kits/stm/65nm-cmos065_536/PRHS65_7.0.a/CADENCE/LEF/PRHS65_soc.lef \  
..../TF.lef \  
..../inv.lef"
```

→ Não esquecer de alterar o DESIGN\_TOP (no *load.tcl*) para o chamar a entidade correta a ser executada, o SystemVerilog deverá estar localizado em “<microX>/lab4/synthesis/src”.

5. No arquivo *cmd\_genus* alterar o SystemVerilog a ser lido pelo script.
6. Substituir no arquivo LEF todas as ocorrências de CoreSite por CORE.

Para esta ação execute: sed -i 's/CoreSite/CORE/g' inv.lef

7. Ao final do innovus verifiquem o DRC

```
@innovus 20> check_drc  
#-check_ndr_spacing_auto # enums={true false auto}, default=auto, user setting  
#-check_same_via_cell true # bool, default=false, user setting  
#-report TF.drc.rpt # string, default="", user setting  
*** Starting Verify DRC (MEM: 3076.8) ***  
  
VERIFY DRC ..... Starting Verification
```

```
VERIFY DRC ..... Initializing
VERIFY DRC ..... Deleting Existing Violations
VERIFY DRC ..... Creating Sub-Areas
VERIFY DRC ..... Using new threading
VERIFY DRC ..... Sub-Area: {0.000 0.000 11.200 13.200} 1 of 1
VERIFY DRC ..... Sub-Area : 1 complete 0 Viols.
```

Verification Complete : 0 Viols.

\*\*\* End Verify DRC (CPU TIME: 0:00:00.0 ELAPSED TIME: 0:00:01.0 MEM: 0.0M) \*\*\*

## Exemplo de circuito completo

