darSchematicCreateFETInst(
cvid
libName
cellName
cellType
instName
leftPins
rightPins
topPins
botPins
origin
rotation
width
length
)
- Similar to UvaEceSchematicCreateFETDefault except I added width and length properties to be read in as floats

darSchematicCreateFET(
cvid
libName
cellName
cellType
instName
leftPins
rightPins
topPins
botPins
origin
rotation
width
length
)
- Similar to UvaEceSchematicCreateFETDefault except I added width and length properties to be read in as strings - This is used to instantiate a transistor that will be parameterized.

darChooseTerminalCase()

```
darChooseTerminalCase(
        technologyCase
        )
```

Description
Returns an array with the letters for the FET terminals (drain, gate, source, and bulk) in either upper or lower case, depending on the requirement of the technology.

Arguments

technologyCase	String containing the letter case required for a given technology, only “upper” or “lower”.

Example
darChooseTerminalCase(“upper”)
=> Returns the following array for FET terminals: (“D”, “G”, “S”, “B”).

darChooseTerminalCase(“lower”)
=> Returns the following array for FET terminals: (“d”, “g”, “s”, “b”).

darSchematic.il

* All of the following procedures check the generated schematic, then create the symbol from the schematic, unless otherwise noted.
* VDD and VSS are always an input/output pin in every circuit.
* All parameters are read in as strings, unless otherwise noted.
* All examples assume the use of the 45nm Free PDK, which requires uppercase letters for the FET terminals. Thus, the parameter "technologyCase" will always be "upper" for each example.

darCreateInverterSchematic()

```
darCreateInverterSchematic(
        libName
        cellName
        FETlibName
        PMOScellName
        NMOScellName
        technologyCase
        )
```

Description

Creates a parameterized inverter with an input pin (IN) and an output pin (OUT). The created inverter has the following parameters: wp, lp, wn, ln.

Arguments

libName	String specifying the library to place the cell.
cellName	String specifying the name of the cell.
FETlibName	String specifying the library where the FETs are located.
PMOScellName	String specifying the cell name of the PMOS FET to be placed.
NMOScellName	String specifying the cell name of the NMOS FET to be placed.
technologyCase	String specifying the letter required for the used technology.

Example

darCreateInverterSchematic("ViPro" "INV" "NCSU_Devices_FreePDK45" "PMOS_VTL" "NMOS_VTH" "upper")

=> Creates an inverter called INV in the ViPro libary by placing a PMOS_VTL device and NMOS_VTH device from the NCSU_Devices_FreePDK45 library.

darCreateParInverter()

darCreateParInverter(
libName
cellName
wp
wn
lp
ln
)

Description
Parameterizes an inverter by creating CDF Parameters for the following parameters: wp, wn, lp, ln. This procedure is normally called within darCreateInverterSchematic().

Arguments

libName	String specifying the library where the inverter to be parameterized is located.
cellName	String specifying the cell name of the inverter to be parameterized.
wp	String that gives a default PMOS width. Usually the minimum device size for the given technology.
wn	String that gives a default NMOS width. Usually the minimum device size for the given technology.
lp	String that gives a default PMOS length. Usually the minimum device size for the given technology.
ln	String that gives a default NMOS width. Usually the minimum device size for the given technology.

Examples
darCreateParInverter(“ViPro” “INV” “180n” “90n” “50n” “50n”)
=> Returns INV as a parameterized inverter with the following default values: PMOS width = 180n, NMOS width = 90n, PMOS legnth = 50n, and NMOS length = 50n.

darCreateBufferSchematic()

darCreateBufferSchematic(
libName
cellName
FETlibName
PMOScellName
NMOScellName
technologyCase
)

Description
Creates a parameterized buffer with an input pin (IN) and an output pin (OUT). The created buffer has the following parameters: wp1, lp1, wn1, ln1, wp1, lp1, wn1, and ln1. The 1 denotes the parameters for the first inverter while the 2 denotes the parameters for the second inverter.

Arguments

libName	String specifying the library to place the cell.
cellName	String specifying the name of the cell.
FETlibName	String specifying the library where the FETs are located.
PMOScellName	String specifying the cell name of the PMOS FET to be placed.
NMOScellName	String specifying the cell name of the NMOS FET to be placed.
technologyCase	String specifying the letter required for the used technology.

Example
darCreateBufferSchematic(“ViPro” “BUFF” “NCSU_Devices_FreePDK45” “PMOS_VTL” “NMOS_VTL “upper”)
=> Creates an buffer called BUFF in the ViPro libary by placing PMOS_VTL devices and NMOS_VTL devices from the NCSU_Devices_FreePDK45 library.

darCreateParBuffer()

```
darCreateParBuffer(
        libName
        cellName
        wp1
        wn1
        lp1
        ln1
        wp2
        wn2
        lp2
        ln2
        )
```

Description
Parameterizes a buffer by creating CDF Parameters for the following parameters: wp1, lp1, wn1, ln1, wp1, lp1, wn1, and ln1. This procedure is normally called within darCreateBufferSchematic().

Arguments

libName	String specifying the library where the buffer to be parameterized is located.
cellName	String specifying the cell name of the buffer to be parameterized.
wp1	String that gives a default PMOS width. Usually the minimum device size for the given technology.
wn1	String that gives a default NMOS width. Usually the minimum device size for the given technology.
lp1	String that gives a default PMOS length. Usually the minimum device size for the given technology.
ln1	String that gives a default NMOS width. Usually the minimum device size for the given technology.
wp2	String that gives a default PMOS width. Usually the minimum device size for the given technology.
wn2	String that gives a default NMOS width. Usually the minimum device size for the given technology.
lp2	String that gives a default PMOS length. Usually the minimum device size for the given technology.

ln2	String that gives a default NMOS width. Usually the minimum device size for the given technology.

Example

darCreateParBuffer(“ViPro” “BUFF” “180n” “90n” “50n” “50n” “275n” “125n” “50n” “50n”) => Returns BUFF as a parameterized buffer with the following default values: PMOS1 width = 180n, NMOS1 width = 90n, PMOS1 legnth = 50n, NMOS1 length = 50n, PMOS2 width = 275n, NMOS2 width = 125n, PMOS2 legnth = 50n, and NMOS2 length = 50n.

darCreateNAND2Schematic()

darCreateNAND2Schematic(
libName
cellName
FETlibName
PMOScellName
NMOScellName
technologyCase
)

Description
Creates a parameterized 2-input NAND gate with an input pin (IN) and an output pin (OUT). The created NAND gate has the following parameters: wp, lp, wn, and ln.

Arguments

libName	String specifying the library to place the cell.
cellName	String specifying the name of the cell.
FETlibName	String specifying the library where the FETs are located.
PMOScellName	String specifying the cell name of the PMOS FET to be placed.
NMOScellName	String specifying the cell name of the NMOS FET to be placed.
technologyCase	String specifying the letter required for the used technology.

Examples
darCreateNAND2Schematic(“ViPro” “NAND2” “NCSU_Devices_FreePDK45” “PMOS_VTH ” “NMOS_VTL “upper”)
=> Creates a 2-input NAND gate called NAND2 in the ViPro libary by placing PMOS_VTH devices and NMOS_VTL devices from the NCSU_Devices_FreePDK45 library.

darCreateParNand2()

darCreateParNand2(
libName
cellName
wp
wn
lp
ln
)

Description
Parameterizes a 2-input NAND gate by creating CDF Parameters for the following parameters: wp, lp, wn, ln. This procedure is normally called within darCreateNAND2Schematic().

Arguments

libName	String specifying the library where the inverter to be parameterized is located.
cellName	String specifying the cell name of the inverter to be parameterized.
wp	String that gives a default PMOS width. Usually the minimum device size for the given technology.
wn	String that gives a default NMOS width. Usually the minimum device size for the given technology.
lp	String that gives a default PMOS length. Usually the minimum device size for the given technology.
ln	String that gives a default NMOS width. Usually the minimum device size for the given technology.

Example
darCreateParNand2(“ViPro” “NAND2” “180n” “90n” “50n” “50n”)
=> Returns NAND2 as a parameterized 2-input NAND gate with the following default values: PMOS width = 180n, NMOS width = 90n, PMOS legnth = 50n, and NMOS length = 50n.

darCreateAND2Schematic()

darCreateAND2Schematic(
libName
cellName
FETlibName
PMOScellName
NMOScellName
technologyCase
)
- Creates a parameterized 2-input AND gate with the name specified by cellName in the library specficied by libName. The procedure places FETs from a library specified by FETlibName. The type of PMOS FET is specfied by PMOScellName and the type of NMOS FET is specified by NMOScellName. The procedure also requires the letter case for FET terminals required by the technology being used.
- Inputs: IN
- Outputs: OUT
- The created 2-input AND gate has the following parameters: wpNAND, lpNAND, wnNAND, lnNAND, wpINV, lpINV, wnINV, lnINV.

darCreateParAnd2(libName cellName)
- Parameterizes the 2-input AND gate specified by cellName in library specified by libName. This procedure is normally called within darCreateNAND2Schematic().

darCreateNOR2Schematic(libName cellName FETlibName PMOScellName NMOScellName technologyCase)
- Creates a parameterized 2-input AND gate with the name specified by cellName in the library specficied by libName. The procedure places FETs from a library specified by FETlibName. The type of PMOS FET is specfied by PMOScellName and the type of NMOS FET is specified by NMOScellName. The procedure also requires the letter case for FET terminals required by the technology being used.
- Inputs: IN
- Outputs: OUT
- The created 2-input NOR gate has the following parameters: wp, lp, wn, ln.

darCreateParNor2(libName cellName)
- Parameterizes the 2-input AND gate specified by cellName in library specified by libName. This procedure is normally called within darCreateNOR2Schematic().

darCreateTXGateSchematic(libName cellName FETlibName PMOScellName NMOScellName technologyCase @key (wp 180.0n) (lp 50.0n) (wn 90.0n) (ln 50.0n))
- Creates a transmission gate with the name specified by cellName in the library specficied by libName. The procedure places FETs from a library specified by FETlibName. The type of PMOS FET is specfied by PMOScellName and the type of NMOS FET is specified by NMOScellName. The procedure also requires the letter case for FET terminals required by the technology being used. The size of the FETs is specified by the optional values.

darCreateTriStateInverterSchematic()

darCreateTriStateInverterSchematic(
libName
cellName
FETlibName
PMOScellName
NMOScellName
technologyCase
)

Description
Creates a parameterized tri-state inverter with an input pin (IN), enable pins (TRIP for PMOS enable FET and TRIN for NMOS enable FET) and an output pin (OUT). The created tri-state inverter has the following parameters: wpINV, wpEN, wnEN, wnINV, lpINV, lpEN, lnEN, and lnINV.

Arguments

libName	String specifying the library to place the cell.
cellName	String specifying the name of the cell
FETlibName	String specifying the library where the FETs are located.
PMOScellName	String specifying the cell name of the PMOS FET to be placed.
NMOScellName	String specifying the cell name of the NMOS FET to be placed.
technologyCase	String specifying the letter required for the used technology.

Example
darCreateTriStateInverterSchematic(“ViPro” “TSI” “NCSU_Devices_FreePDK45” “PMOS_VTH” “NMOS_VTL “upper”)
=> Creates a tri-state inverter called TSI in the ViPro libary by placing PMOS_VTH devices and NMOS_VTL devices from the NCSU_Devices_FreePDK45 library.

darCreateParTriStateInverter()

darCreateParTriStateInverter(
libName
cellName
wpINV
wpEN
wnEN
wnINV
lpINV
lpEN
lnEN
lnINV
)

Description
Parameterizes a tri-state inverter by creating CDF Parameters for the following parameters: wpINV, wpEN, wnEN, wnINV, lpINV, lpEN, lnEN, and lnINV. This procedure is normally called within darCreateTriStateInverterSchematic().

Example
darCreateParTriStateInverter(“ViPro” “TSI” “360n” “360n” “180n” 180n” “50n” “50n” “50n” “50n”)
=> Returns TSI as a parameterized tri-state inverter with the following default values: Inverter PMOS width = 360n, Enable PMOS width = 360n, Enable NMOS width = 180n, Inverter NMOS width = 180n, Inverter PMOS length = 50n, Enable PMOS length = 50n, Enable NMOS length = 50n, Inverter NMOS length = 50n

darCreateBitcellSchematic()

darCreateBitcellSchematic(
    libName
    cellName
    FETlibName
    PMOScellName
    NMOScellName
    technologyCase
    @key (wpg wnMin) (lpg lnMin) (wpu wpMin) (lpu lpMin) (wpd wnMin) (lpd lnMin)
)

Description
Creates a typical 6 transistor bitcell with an input word line pin (WL) and input/ouput bit line pins (BL/BLB). FET sizes are set to default which is the minimum device sizes for the given technology. Users can alter any of the 6 FET widths and/or lengths by changing the desired parameter. The bitcell has the following parameters: wpg, lpg, wpu, lpu, wpd, and lpd. Requires these parameters to be passed as strings.

Arguments

libName	String specifying the library to place the cell.
cellName	String specifying the name of the cell.
FETlibName	String specifying the library where the FETs are located.
PMOScellName	String specifying the cell name of the PMOS FET to be placed.
NMOScellName	String specifying the cell name of the NMOS FET to be placed.
technologyCase	String specifying the letter required for the used technology.
wpg	String specifying the width of the pass gate (NMOS) FETs.
lpg	String specifying the legnth of the pass gate (NMOS) FETs.
wpu	String specifying the width of the pull up (PMOS) FETs.
lpu	String specifying the width of the pull up (PMOS) FETs.
wpd	String specifying the width of the pull down (NMOS) FETs.
lpd	String specifying the width of the pull down (NMOS) FETs.

Example
darCreateBitcellSchematic(“ViPro” “BCmin” “NCSU_Devices_FreePDK45” “PMOS_VTL” “

NMOS_VTL “upper”)
=> Creates a 6T bitcell called BCmin in the ViPro libary by placing PMOS_VTL devices and NMOS_VTL devices from the NCSU_Devices_FreePDK45 library. The bitcell will have minimum device sizing because no parameters were specified.

darCreateBitcellSchematic(“ViPro” “BC” “NCSU_Devices_FreePDK45” “PMOS_VTL” “NMOS_VTL “upper” ?wpg “570n” ?wpu “220n”)
=> Creates a 6T bitcell called BCmin in the ViPro libary by placing PMOS_VTL devices and NMOS_VTL devices from the NCSU_Devices_FreePDK45 library. The bitcell will have minimum device sizing except for the pass gate width = 570n and pull up width = 220n..

darCreateSenseAmpSchematic()

darCreateSenseAmpSchematic(
libName
cellName
FETlibName
PMOScellName
NMOScellName
technologyCase
cellName_nand
@key (wen wnMin) (len lnMin) (weql wpMin) (leql wpMin) (wpsa wpMin) (lpsa lpMin) (wnsa wnMin) (lnsa lnMin) (wbl wnMin) (lbl lnMin) (wsapc wpMin) (lsapc lpMin) (wpNAND wpMin) (lpNAND lpMin) (wnNAND wnMin) (lnNAND lnMin)
)

Description
Creates a sense amplifier with input pins enable and precharge (SAE and SAEPREC, respectively), input/output pins for the column-MUXed bit lines (RDWR and NRDWR), and output pin from the SR latch (SD). The procedure also requires the cell name of the 2-input NAND gate to be used in the SR latch. It is assumed the NAND gate is located in the same library as where the sense amp cell will be placed. The sense amp has the following parameters: wen, len, weql, leql, wpsa, lpsa, wnsa, lnsa, wbl, lbl, wsapc, lsapc, wpNAND, lpNAND, wnNAND, and lnNAND. Requires these parameters to be pass as strings.

The abbreviations correspond to the following FETs in the schematic:
en - enable transistor (footer) (NMOS)
eql - equalize transistors for precharging the out/outb nodes of the sense amp (PMOS)
sa - transistors of the actual sense amp (PMOS and NMOS)
bl - transistors for BL/BLB inputs (NMOS)
sapc - precharge transistors for BL/BLB nodes (PMOS)
NAND - the NAND gate for the SR latch

Arguments

libName	String specifying the library to place the cell.
cellName	String specifying the name of the cell.
FETlibName	String specifying the library where the FETs are located.
PMOScellName	String specifying the cell name of the PMOS FET to be placed.
NMOScellName	String specifying the cell name of the NMOS FET to be placed.
technologyCase	String specifying the letter required for the used technology.
cellName_nand	String specifying the cell name of the 2-input NAND gate to be used

	in the SR latch.
wen	String specifying the width of the enable (NMOS) FET (footer device).
len	String specifying the length of the enable (NMOS) FET (footer device).
weql	String specifying the width of the equalize (PMOS) FETs.
leql	String specifying the length of the equalize (PMOS) FETs.
wpsa	String specifying the width of the sense amp (PMOS) FETs.
lpsa	String specifying the length of the sense amp (PMOS) FETs.
wnsa	String specifying the width of the sense amp (NMOS) FETs.
lnsa	String specifying the length of the sense amp (NMOS) FETs.
wbl	String specifying the width of the input BL/BLB (NMOS) FETs.
lbl	String specifying the length of the input BL/BLB (NMOS) FETs.
wsapc	String specifying the width of the BL/BLB precharge (PMOS) FETs.
lsapc	String specifying the length of the BL/BLB precharge (PMOS) FETs.
wpNAND	String specifying the PMOS width of the SR NAND gate.
lpNAND	String specifying the PMOS length of the SR NAND gate.
wnNAND	String specifying the NMOS width of the SR NAND gate.
lnNAND	String specifying the NMOS length of the SR NAND gate.

Examples

darCreateSenseAmpSchematic(“ViPro” “SAmin” “NCSU_Devices_FreePDK45” “PMOS_VTL ” “NMOS_VTL “upper”)
=> Creates a sense amp called SAmin in the ViPro libary by placing PMOS_VTL devices and NMOS_VTL devices from the NCSU_Devices_FreePDK45 library. The sense amp will have minimum device sizing because no parameters were specified.

darCreateSenseAmpSchematic(“ViPro” “SA” “NCSU_Devices_FreePDK45” “PMOS_VTL” “ NMOS_VTL “upper” ?wen “375n ?wnNAND “450n”)
=> Creates a sense amp called SA in the ViPro libary by placing PMOS_VTL devices and NMOS_VTL devices from the NCSU_Devices_FreePDK45 library. The sense amp will have minimum device sizing except for the enable FETs with width = 375n and the NMOS widths in

the NAND-based SR latch = 450n.

darCreateSingleCDSchematic()

```
darCreateSingleCDSchematic(
        libName
        cellName
        FETlibName
        PMOScellName
        NMOScellName
        technologyCase
        @key (wpc wpMin) (lpc lpMin) (wpTX wpMin) (lpTX lpMin) (wnTX wnMin)
        (lnTX lnMin)
        )
```

Description

Creates a single column MUX, employing a transmission gate, with an input pin to select the column (CSEL and CSELB), an input pin to precharge the BL/BLB node (PCH), and input/ output pins of bit lines from the SRAM (BL and BLB) and the MUXed bit lines (RDWR and NRDWR). There is 2 precharge transistors and one equalize transistor. The column MUX has the following parameters: wpc, lpc, wpTX, lpTX, wnTX, and lnTX. Requires these parameters to be passed as strings.

Arguments

libName	String specifying the library to place the cell.
cellName	String specifying the name of the cell.
FETlibName	String specifying the library where the FETs are located.
PMOScellName	String specifying the cell name of the PMOS FET to be placed.
NMOScellName	String specifying the cell name of the NMOS FET to be placed.
technologyCase	String specifying the letter required for the used technology.
wpc	String specifying the width of the precharge/equalize (PMOS) FETs.
lpc	String specifying the length of the precharge/equalize (PMOS) FETs.
wpTX	String specifying the width of the transmission gate PMOS FETs.
lpTX	String specifying the length of the transmission gate PMOS FETs.
wnTX	String specifying the width of the transmission gate NMOS FETs.
lnTX	String specifying the length of the transmission gate NMOS FETs.

Examples

darCreateSingleCDSchematic(“ViPro” “SingleCDmin” “NCSU_Devices_FreePDK45” “PMOS_VTL” “NMOS_VTL “upper”)

=> Creates a single column MUX called SingleCDmin in the ViPro library by placing PMOS_VTL devices and NMOS_VTL devices from the NCSU_Devices_FreePDK45 library. The CD will have minimum device sizing because no parameter was specified

darCreateSingleCDSchematic(“ViPro” “SingleCD” “NCSU_Devices_FreePDK45” “PMOS_VTL” “NMOS_VTL “upper” ?wpTX “265n” ?wnTX “175n”)

=> Creates a single column MUX called SingleCD in the ViPro library by placing PMOS_VTL devices and NMOS_VTL devices from the NCSU_Devices_FreePDK45 library. The MUX has minimum device sizing except for the widths of the PMOS and NMOS of the transmission gate, with PMOS width = 265n and NMOS width = 175n.

darCreateMultiCDSchematic()

darCreateMultiCDSchematic(
libName
cellName
FETlibName
PMOScellName
NMOScellName
technologyCase
MUX
@key (wpc wpMin) (lpc lpMin) (wpTX wpMin) (lpTX lpMin) (wnTX wnMin)
(lnTX lnMin)
)

Description
Creates a multi column MUX, employing a transmission gate, with input pins to select the column (CSEL<0:MUX-1> and CSELB<0:MUX-1> - MUX is the amount of columns divided the word size), an input pin to precharge the BL/BLB node (PCH), and input/output pins of bit lines from the SRAM (BL<0:MUX-1> and BLB<0:MUX-1>) and the MUXed bit lines (RDWR and NRDWR). There is 2 precharge transistors and one equalize transistor. The column MUX has the following parameters: wpc, lpc, wpTX, lpTX, wnTX, and lnTX. Requires these parameters to be passed as strings.

Arguments

libName	String specifying the library to place the cell.
cellName	String specifying the name of the cell.
FETlibName	String specifying the library where the FETs are located.
PMOScellName	String specifying the cell name of the PMOS FET to be placed.
NMOScellName	String specifying the cell name of the NMOS FET to be placed.
technologyCase	String specifying the letter required for the used technology.
wpc	String specifying the width of the precharge/equalize (PMOS) FETs.
lpc	String specifying the length of the precharge/equalize (PMOS) FETs.
wpTX	String specifying the width of the transmission gate PMOS FETs.
lpTX	String specifying the length of the transmission gate PMOS FETs.
wnTX	String specifying the width of the transmission gate NMOS FETs.

lnTX String specifying the length of the transmission gate NMOS FETs.

Examples

darCreateMultiCDSchematic(“ViPro” “MultiCDmin” “NCSU_Devices_FreePDK45” “PMOS_VTL” “NMOS_VTL “upper”)

=> Creates a multi column MUX called MultiCDmin in the ViPro library by placing PMOS_VTL devices and NMOS_VTL devices from the NCSU_Devices_FreePDK45 library. The circuit is similar to the single CD except that there are more CSEL/CSELB signals to choose which BL/BLB to pass through. The CD will have minimum device sizing because no parameter was specified.

darCreateMultiCDSchematic(“ViPro” “MultiCD” “NCSU_Devices_FreePDK45” “PMOS_VTL” “NMOS_VTL “upper” ?wpTX “265n” ?wnTX “175n”)

=> Creates a multi column MUX called MultiCD in the ViPro library by placing PMOS_VTL devices and NMOS_VTL devices from the NCSU_Devices_FreePDK45 library. The circuit is similar to the single CD except that there are more CSEL/CSELB signals to choose which BL/BLB to pass through. The MUX has minimum device sizing except for the widths of the PMOS and NMOS of the transmission gate, with PMOS width = 265n and NMOS width = 175n.

darChooseAndCreateCDSchematic()

```
darChooseAndCreateCDSchematic(
        libName
        cellName
        FETlibName
        PMOScellName
        NMOScellName
        technologyCase
        C
        DW
        @key (wpc wpMin) (lpc lpMin) (wpTX wpMin) (lpTX lpMin) (wnTX wnMin)
        (lnTX lnMin)
        )
```

Description

Invokes the correct procedure to create a column MUX depending on the number of columns (C) and the word size (DW), either a single column MUX or multi column MUX. This procedure passes the following parameters to the actual procedure that creates the column MUX: wpc, lpc, wpTX, lpTX, wnTX, and lnTX.

Arguments

libName	String specifying the library to place the cell.
cellName	String specifying the name of the cell.
FETlibName	String specifying the library where the FETs are located.
PMOScellName	String specifying the cell name of the PMOS FET to be placed.
NMOScellName	String specifying the cell name of the NMOS FET to be placed.
technologyCase	String specifying the letter required for the used technology.
C	Integer specifying the number of columns in the SRAM.
DW	Integer specifying the length of a word in the SRAM.
wpc	String specifying the width of the precharge/equalize (PMOS) FETs.
lpc	String specifying the length of the precharge/equalize (PMOS) FETs.
wpTX	String specifying the width of the transmission gate PMOS FETs.
lpTX	String specifying the length of the transmission gate PMOS FETs.

wnTX	String specifying the width of the transmission gate NMOS FETs.
lnTX	String specifying the length of the transmission gate NMOS FETs.

Examples

darChooseandCreateCDSchematic(“ViPro” “CD” “NCSU_Devices_FreePDK45” “PMOS_VTL ” “NMOS_VTL “upper” 32 32)

=> Creates a single column MUX called CD in the ViPro library by placing PMOS_VTL devices and NMOS_VTL devices from the NCSU_Devices_FreePDK45 library. It is only a one column MUX because the word size is equal to the number of columns. The control signals for the columns are CSEL and CSELB. The circuit then follows the single column MUX procedure. See darCreateSingleCDSchematic() examples for further reference on device sizing.

darChooseandCreateCDSchematic(“ViPro” “CD” “NCSU_Devices_FreePDK45” “PMOS_VTL ” “NMOS_VTL “upper” 64 16)

=> Creates a multi column MUX called CD in the ViPro library by placing PMOS_VTL devices and NMOS_VTL devices from the NCSU_Devices_FreePDK45 library. It is a multi column MUX because the number of columns is greater than word size. The control signals for the columns are CSEL<0:3> and CSELB<0:3>. The circuit then follows the multi column MUX procedure. See darCreateMultiCDSchematic() examples for further reference on device sizing.

darCreateDFFSchematic()

darCreateDFFSchematic(
libName
cellName
FETlibName
PMOScellName
NMOScellName
technologyCase
@key (wpBuffer wpMin) (lpBuffer lpMin) (wnBuffer wnMin) (lnBuffer lnMin) (wpINV wpMin) (lpINV lpMin) (wnINV wnMin) (lnINV lnMin) (wpTXGate wpMin) (lpTXGate lpMin) (wnTXGate wnMin) (lnTXGate lnMin) (wpDriver wpMin) (lpDriver lpMin)
(wnDriver wnMin) (lnDriver lnMin) (minWp wpMin) (minLp lpMin) (minWn wnMin) (minLn lnMin)
)

Description
Creates a D Flip Flop that has input pins for the clock and data input (CLK and D, respectively) and an output for the data to be stored (Q). The buffers for the clock signals are sized at 1.5x minimum device size for the given technology*. The tri-state inverters are minimum sized. The inverters on the output path are sized in an FO4 fashion. "Backward" sizing is implemented until minimum size is reached and all subsequent inverters are minimum sized. The DFF has the following paramters: wpBuffer, lpBuffer, wnBuffer, lnBuffer, wpINV, lpINV, wnINV, lnINV, wpTXGate, lpTXGate, wnTXGate, lnTXGate, wpDriver, lpDriver, wnDriver, lnDriver, minWp, minLp, minWn, and minLn.

* - This procedure requires that the sizes passed for the clock buffer must be 1.5x less than the desired size, i.e. must pass desired size/1.5, because the procedure automatically multiplies by 1.5.

Arguments

libName	String specifying the library to place the cell.
cellName	String specifying the name of the cell.
FETlibName	String specifying the library where the FETs are located.
PMOScellName	String specifying the cell name of the PMOS FET to be placed.
NMOScellName	String specifying the cell name of the NMOS FET to be placed.
technologyCase	String specifying the letter required for the used technology.
wpBuffer	String specifying the PMOS width for the two inverters that create the clock signal. Must be 1.5x less than desired value,

	i.e. desired size/1.5.
lpBuffer	String specifying the PMOS length for the two inverters that create the clock signal.
wnBuffer	String specifying the NMOS width for the two inverters that create the clock signal. Must be 1.5x less than desired value, i.e. desired size/1.5.
lnBuffer	String specifying the NMOS length for the two inverters that create the clock signal.
wpINV	String specifying the PMOS width for the two tri-state inverters.
lpINV	String specifying the PMOS length for the two tri-state inverters.
wnINV	String specifying the NMOS width for the two tri-state inverters.
lnINV	String specifying the NMOS length for the two tri-state inverters.
wpTXGate	String specifying the PMOS width for the transmission gate.
lpTXGate	String specifying the PMOS length for the transmission gate.
wnTXGate	String specifying the NMOS width for the transmission gate.
lnTXGate	String specifying the NMOS length for the transmission gate.
wpDriver	String specifying the PMOS width for the final inverter.
lpDriver	String specifying the PMOS length for the final inverter.
wnDriver	String specifying the NMOS width for the final inverter.
lnDriver	String specifying the NMOS length for the final inverter.
minWp	String specifying the minimum PMOS width for the given technology.
minLp	String specifying the minimum PMOS length for the given technology.
minWn	String specifying the minimum NMOS width for the given technology.
minLn	String specifying the minimum NMOS length for the given technology.

Examples

darCreateDFFSchematic(“ViPro” “DFFmin” “NCSU_Devices_FreePDK45” “PMOS_VTL” “N

MOS_VTL “upper”)
=> Creates a D Flip Flop called DFFmin in the ViPro library by placing PMOS_VTL devices and NMOS_VTL devices from the NCSU_Devices_FreePDK45 library. All devices are minimum sized since no parameters were specified, except for the clock buffers which is hard-coded to be 1.5x mimimum device size for the given technology.

darCreateDFFSchematic(“ViPro” “DFF” “NCSU_Devices_FreePDK45” “PMOS_VTL” “NMOS_VTL “upper” ?wpDriver “22000n” ?wnDriver “14000n”)
=> Creates a D Flip Flop called DFF in the ViPro library by placing PMOS_VTL devices and NMOS_VTL devices from the NCSU_Devices_FreePDK45 library. Since the final inverter width was given (Driver), the last stage has PMOS width = 22000n and NMOS width = 14000n. The previous inverters will have PMOS width = 5500n and NMOS width = 2800n. The PMOS and NMOS width will then be divided by 4 until either minimum device size is reached or the first inverter is sized.

darCreateDelayElementSchematic()

```
darCreateDelayElementSchematic(
        libName
        cellName
        FETlibName
        PMOScellName
        NMOScellName
        technologyCase
        @key (wpDriver wpMin) (lpDriver lpMin) (wnDriver wnMin) (lnDriver lnMin)
        (minWp wpMin) (minLp lpMin) (minWn wnMin) (minLn lnMin)
        )
```

Description
Creates a delay element by placing 8 inverters in series, sized in an F04 fashion. "Backward" sizing is implemented until minimum size is reached and all subsequent inverters are minimum sized. There is an input pin (IN) and an output pin (OUT). The delay element has the following parameters: wpDriver, lpDriver, wnDriver, lnDriver, minWp, minLp, minWn, and minLn.

Arguments

libName	String specifying the library to place the cell.
cellName	String specifying the name of the cell.
FETlibName	String specifying the library where the FETs are located.
PMOScellName	String specifying the cell name of the PMOS FET to be placed.
NMOScellName	String specifying the cell name of the NMOS FET to be placed.
technologyCase	String specifying the letter required for the used technology.
wpDriver	String specifying the PMOS width of the final inverter.
lpDriver	String specifying the PMOS length of the final inverter.
wnDriver	String specifying the NMOS width of the final inverter.
lnDriver	String specifying the NMOS length of the final inverter.
minWp	String specifying the minimum PMOS width for the given technology.
minLp	String specifying the minimum PMOS length for the given technology.
minWn	String specifying the minimum NMOS width for the given technology.

minLn String specifying the minimum NMOS length for the given technology.

Examples

darCreateDelayElementSchematic(“ViPro” “DEmin” “NCSU_Devices_FreePDK45” “PMOS_VTL” “NMOS_VTL “upper”)

=> Creates a delay element called DEmin in the ViPro library by placing PMOS_VTL devices and NMOS_VTL devices from the NCSU_Devices_FreePDK45 library. Since the final stage size was not specified, all inverters in the chain are minimum sized, specific to the technology being used.

darCreateDelayElementSchematic(“ViPro” “DE” “NCSU_Devices_FreePDK45” “PMOS_VTL” “NMOS_VTL “upper” ?wpDriver “900n” ?wnDriver “500n”)

=> Creates a delay element called DE in the ViPro library by placing PMOS_VTL devices and NMOS_VTL devices from the NCSU_Devices_FreePDK45 library. Since final stage size was specified, the final (8th) inverter has PMOS width = 900n and NMOS width = 500n. The previous (7th) inverter then has PMOS width = 225n and NMOS width = 125n (900n and 500n were both divided by 4). The subsequent previous inverters will then be sized 4 times smaller than the current stage until minimum device sizes are reached.

darCreate2to1MUXSchematic()

darCreate2to1MUXSchematic(
libName
cellName
cellName_inverter
cellName_nand
@key (wpNAND wpMin) (wnNAND wnMin) (lpNAND lpMin) (lnNAND lnMin)
(wpINV wpMin) (wnINV wnMin) (lpINV lpMin) (lnINV lnMin)
)

Description
Creates a 2:1 MUX with input pins for data (IN0 and IN1) and the select line (SELECT). The procedure requires the cell names of the inverter and NAND gates to be used. The 2:1 MUX has the following parameters: wpNAND, wnNAND, lpNAND, lnNAND, wpINV, wnINV, lpINV, and lnINV.

Arguments

libName	String specifying the library to place the cell.
cellName	String specifying the name of the cell.
cellName_inverter	String specifying the cell name of the inverter to be used.
cellName_nand	String specifying the cell name of the 2-input NAND gate to be used.

Example
darCreate2to1MUXSchematic(“ViPro” “2to1” “INV” “NAND2”)
=> Creates a 2:1 MUX called 2to1 in the ViPro library. Both the inverter and NAND gates used are minimum sized since no parameters were specified.

darCreate2to1MUXSchematic(“ViPro” “2to1” “INV” “NAND2” ?wpINV “225n” ? wnNAND “300n”)
=> Creates a 2:1 MUX called 2to1 in the ViPro library. The inverter has PMOS width = 225n and the NAND gate has NMOS widths = 300n. All other devices are minimum sized because no other parameters were specified

darCreate4to1MUXSchematic()

darCreate4to1MUXSchematic(
    libName
    cellName
    cellName_2to1MUX
)

Description
- Creates a 4:1 MUX with the name specified by cellName in the library specficied by libName.
- The 4:1 MUX is made by taking 2 2:1 MUX and inputting these intermediate outputs into a final 2:1 MUX.
- The procedure requires the cell name of a 2:1 MUX.
- Inputs: IN0, IN1, IN2, IN3, SELECT0, SELECT1
- Outputs: OUT

darCreateTunableDelaySchematic()

darCreateTunableDelaySchematic(
libName
cellName
cellName_delay
cellName_4to1
)

Description
Creates a tunable delay element which is 4 identical delay elements in series. The output of each delay element is put into a 4:1 MUX which then chooses how much delay to use. There is an input pin for the signal to be delayed, input pins for the select lines of the MUX (SELECT0 and SELECT1), and an output pin of the MUX (OUT). The cell names of the delay element and 4:1 MUX to be used must be passed as arguments. To size the delay element, must first create a delay element (See darCreateDelayElementSchematic() for further reference). To size the 4:1 MUX, must call darCreate2to1MUXSchematic() and pass desired sizing, then pass the 2:1 MUX's cell name to darCreate4to1MUXSchematic()

Arguments

libName	String specifying the library to place the cell.
cellName	String specifying the name of the cell.
cellName_delay	String specifying the cell name of the delay element to be used.
cellName_4to1	String specifying the cell name of the 4:1 MUX to be used.

Example
darCreateTunableDelaySchematic("ViPro" "TuneDelay" "DE" "4to1")
=> Creates a tunable delay element called TuneDelay in the ViPro library. The delay element DE is used for the four delays while the MUX 4to1 is used to MUX the four delayed signals. The sizing of the delay element and MUX was already set when those individual components were created.

darCreateIOSchematic()

darCreateIOSchematic(
libName
cellName
cellName_inverter
cellName_triState
cellName_DFF
)

Description
Creates the Input/Output Block for the SRAM. This block contains two DFFs to store the input and output data value for a bit. There is also an inverter and two tri-state inverters to generate the signal to be placed on the bit lines during a write. In addition, there are two inverters to create the enable signals for the tri-state inverters. There are several input pins: data in signal (DIN<i>), two clock signals (ICLK and OCLK), the data bit to be stored (SD<i>) which comes from the output of the Sense Amp's SR latch, and a write enable signal (WEN). There are two input/ouput pins for the bit lines (RDWR<i> and NRWDR<i>). There is a single output pin for the data out signal (DOUT<i>). The procedure requires the cell names of the inverter, tri-state inverter, and DFF to be used within this block. The sizing of the inverter and tri-state inverters can be altered through the procedure parameters while the sizing of the DFF must have been set when creating the DFF.

Arguments

libName	String specifying the library to place the cell.
cellName	String specifying the name of the cell.
cellName_inverter	String specifying the cell name of the inverter to be used.
cellName_triState	String specifying the cell name of the tri-state inverter to be used.
cellName_DFF	String specifying the cell name of the DFF to be used.

Examples
darCreateIOSchematic("ViPro" "IOBlock" "INV" "TSI" "DFF")
=> Creates the input/output block called IOBlock in the ViPro library. It uses the inverter INV, the tri-state inverter TSI, and the D Flip Flop DFF within the block.

darCreateIOSchematic("ViPro" "IOBlock" "INV" "TSI" "DFF")
=> **NEED TO MAKE EXAMPLE WHERE PARAMETERS ARE USED**

darCreateBufferChainSchematic()

```
darCreateBufferChainSchematic(
        libName
        cellName
        FETlibName
        PMOScellName
        NMOScellName
        technologyCase
        stages
        @key (wpDriver wpMin) (lpDriver lpMin) (wnDriver wnMin) (lnDriver lnMin)
        (minWp wpMin) (minLp lpMin) (minWn wnMin) (minLn lnMin)
        )
```

Description

Creates a buffer chain by placing X inverters in series. X is a user specified input denoting the number of stages - how many inverters to place in series. The entire circuit is sized in an F04 fashion. "Backward" sizing is implemented until minimum size is reached and all subsequent inverters are minimum sized. There is an input pin (IN) and an output pin (OUT). The delay element has the following parameters: wpDriver, lpDriver, wnDriver, lnDriver, minWp, minLp, minWn, and minLn.

Arguments

libName	String specifying the library to place the cell.
cellName	String specifying the name of the cell.
FETlibName	String specifying the library where the FETs are located.
PMOScellName	String specifying the cell name of the PMOS FET to be placed.
NMOScellName	String specifying the cell name of the NMOS FET to be placed.
technologyCase	String specifying the letter required for the used technology.
stages	Integer specifying the number of stages (inverters) in the buffer chain.
wpDriver	String specifying the PMOS width of the final inverter.
lpDriver	String specifying the PMOS length of the final inverter.
wnDriver	String specifying the NMOS width of the final inverter.
lnDriver	String specifying the NMOS length of the final inverter.
minWp	String specifying the minimum PMOS width for the given technology.

minLp	String specifying the minimum PMOS length for the given technology.
minWn	String specifying the minimum NMOS width for the given technology.
minLn	String specifying the minimum NMOS length for the given technology.

Examples

darCreateBufferChainSchematic(“ViPro” “BuffChain” “NCSU_Devices_FreePDK45” “PMOS_VTL” “NMOS_VTL “upper” 10)

=> Creates a buffer chain of 10 series inverters called BuffChain in the ViPro library by placing PMOS_VTL devices and NMOS_VTL devices from the NCSU_Devices_FreePDK45 library. The procedure is exactly like darCreateDelayElementSchematic() with respect to sizing and circuit creation, except that the number of stages can be specified (whereas in darCreateDelayElementSchematic(), the number of stages was fixed to 8). See darCreateDelayElementSchematic() for further reference.

darCreateWLDriver8Schematic()

darCreateWLDriver8Schematic(
libName
cellName
cellName_inverter
cellName_nand
cellName_and
@key (wpInv wpMin) (wnInv wnMin) (lpInv lpMin) (lnInv lnMin) (wpNAND wpMin) (lpNAND lpMin) (wnNAND wnMin) (lnNAND lnMin) (wpANDnand wpMin) (lpANDnand lpMin) (wnANDnand wnMin) (lnANDnand lnMin) (wpANDinv wpMin) (lpANDinv lpMin) (wnANDinv wnMin) (lnANDinv lnMin)
)

Description
Creates the word line driver for 8 word lines by combining the outputs from the three predecoders.
The input pins are the 8 outputs from each of the 3 predecoders (PRE8_6<0:7>, PRE5_3<0:7>, WLclk<0:7>). There are also 2 input pins (A<0:7>, B<0:7>) which pluck off the correct output from the 2 most significant decoders to access the desired group of 8 word lines. There are 8 output pins for the word lines (WL<0:7>). The procedure requires the cell names of the NAND, AND, and inverter gates to be used. Sizing of each gate is accessible through the following parameters: wpInv, wnInv, lpInv, lnInv, wpNAND, lpNAND, wnNAND, lnNAND, wpANDnand, lpANDnand, wnANDnand, lnANDnand, wpANDinv, lpANDinv, wnANDinv, lnANDinv

Arguments

libName	String specifying the library to place the cell.
cellName	String specifying the name of the cell.
cellName_inverter	String specifying the cell name of the inverter to be used.
cellName_nand	String specifying the cell name of the NAND gate to be used.
cellName_and	String specifying the cell name of the AND gate to be used.

Examples
darCreateWLDriver8Schematic(“ViPro” “WLDriver8” “INV” “NAND2” “AND2”)
=> Creates a word line driver for 8 word lines called WLDriver8 in the ViPro library. Since no parameters were specified, minimum sizing is used for the inverter, NAND, and AND gates. Th

darCreateWLDriver8Schematic(“ViPro” “WLD8” “INV” “NAND2” “AND2” ?wpInv “225n” ?wnANDnand “200n” ?wnANDinv “100n”)
=> Creates a word line driver for 8 word lines called WLD8 in the ViPro library. The inverter has PMOS width = 225n, the NMOSs in the NAND part of the AND gate has width = 200n,

and the NMOS in the inverter part of the AND gate has width = 100n. All other devices are minimum size since no other parameters were specified.

darCreateWLBufferSchematic()

darCreateWLBufferSchematic(
libName
cellName
R
cellName_WLBufferChain
)

Description
Creates the buffer chain ("k" chain) for all of the word lines, which allows for buffering up the signal coming from the word line driver. This buffered signal then goes into the bitcell array. The procedure requires the number of rows in the SRAM and the cell name of the already created "k" buffer chain. There are input pins for all the word lines (IN<0:R-1>) and output pins for all the word lines (OUT<0:R-1>).

Arguments

libName	String specifying the library to place the cell.
cellName	String specifying the name of the cell.
R	Integer specifying the number of rows in the SRAM.
cellName_WLBufferChain	String specifying the cell name of the "k" buffer chain to be used.

Example
darCreateWLBufferSchematic("ViPro" "WLBuffer" 64 "KChain")
=> Creates a buffer chain called WLBuffer in the ViPro library for all 64 word lines. The buffer chain to be used (the "k" chain) should have been previously created and is called KChain. Thus, the sizing for the "k buffers" was set when the single "k chain" was created. This procedure only places the previously created "k" chain for the number of rows in the SRAM.

darCreatePREWLBufferSchematic()

darCreatePREWLBufferSchematic(
    libName
    cellName
    cellName_PRE_WLBufferChain
)

Description
Creates the buffer chain ("n" chain) for all of the 8 outputs of the three 3:8 predecoders. These buffered signals then go into the word line driver. The procedure requires the cell name of the already created "n" buffer chain. There are input pins for all outputs of all the predecoders (PRE_PRE8_6<0:7>, PRE_PRE5_3<0:7>, and PRE_WLclk<0:7>) and output pins for buffered signals (PRE8_6<0:7>, PRE5_3<0:7>, and WLclk<0:7>).

Arguments

libName	String specifying the library to place the cell.
cellName	String specifying the name of the cell.
cellName_PRE_WLBufferChain	String specifying the cell name of the "n" buffer chain to be used.

Example
darCreatePREWLBufferSchematic("ViPro" "PRE_WLBuffer" "Chain")
=> Creates a buffer chain called PRE_WLBuffer in the ViPro library for all 24 outputs from the three 3:8 predecoders. The buffer chain to be used (the "n" chain) should have been previously created and is called NChain. Thus, the sizing for the "n buffers" was set when the single "n chain" was created. This procedure only places the previously created "n chain" for the each output of all the predecoders.

darChoosePREWLDriverNode()

darChoosePREWLDriverNode(
n
)

Description
Returns an array with the names of the nodes after the three 3:8 decoders. The input is the "n" from the decoder output, the number of buffer stages after the three 3:8 decoders.

Arugments
n Integer specifying the number of "n buffers" after the predecoders.

Examples
darChoosePREWLDriverNode(0)
=> Returns the list ("PRE8_6<0:7>" "PRE5_3<0:7>" "WLclk<0:7>") to be used as the node names after the predecoders - no "n buffers" so it connects the predecoders' outputs straight into the word line driver.

darChoosePREWLDriverNode(3)
=> Returns the list ("PRE_PRE8_6<0:7>" "PRE_PRE5_3<0:7>" "PRE_WLclk<0:7>") to be used as the node names after the predecoders - there are three "n buffers" so it connects the predecoders' outputs into the buffers.

darChoosePREWLNode()

darChoosePREWLNode(
k
R
)

Description
Returns a string with the name of the nodes after the word line driver. The input is the number of rows in the SRAM and the "k" from the decoder output, the number of buffer stages after the word line driver.

Arguments

k	Integer specifying the number of "k buffers" after the word line driver.
R	Integer specifying the number of rows in the SRAM.

Examples
darChoosePREWLNode(2 128)
=> Returns the string "PRE_WL<0:127>" to be used as the node name after the word line driver - there are two "k buffers" so it connects the word line driver to the buffers.

darChoosePREWLNode(0 128)
=> Returns the string "WL<0:127>" to be used as the node name after the word line driver - no "k buffers" so it connects the word line driver straight into the bitcell Array.

darBasicSchematic.il

darUvaEceSchematicCreateInstParInverter(cvid libName cellName Iname in out VDD VSS location @key (wp 180n) (wn 90n) (lp 50n) (ln 50n) (m 1))

darUvaEceSchematicCreateInstParNand2(cvid libName cellName Iname inA inB out VDD VSS location @key (wp 180n) (wn 90n) (ln 50n) (lp 50n) (m 1))

darUvaEceSchematicCreateInstParAnd2(cvid libName cellName Iname inA inB out VDD VSS location @key (wpNAND 180n) (wnNAND 180n) (wpINV 180n) (wnINV 90n) (lpNAND 50n) (lnNAND 50n) (lpINV 50n) (lnINV 50n) (m 1))

darUvaEceSchematicCreateInstParNor2(cvid libName cellName Iname inA inB out VDD VSS location @key (wp 360n) (wn 90n) (ln 50n) (lp 50n) (m 1))

darUvaEceSchematicCreateInstParBuf(cvid libName cellName Iname in out VDD VSS location @key (wp1 180n) (wn1 90n) (wp2 180n) (wn2 90n) (lp1 50n) (ln1 50n) (lp2 50n) (ln2 50n) (m 1))

darUvaEceSchematicCreateInstParTriState(cvid libName cellName Iname in triP triN out VDD VSS location @key (wpInv 360n) (wnInv 180n) (wpEN 360n) (wnEN 180n) (lpInv 50n) (lnInv 50n) (lpEN 50n) (lnEN 50n) (m 1))

darBitcellArray_Schem.il

darCreateArraySchematic()

darCreateArraySchematic(
    libName
    cellName
    rows
    columns
    cellName_bitcell
)

Description
Creates a bitcell array with the specified number of rows and columns. There are input pins for the word lines (WL<0:rows-1>) and innput/output pins for the bit lines (BL/BLB<0:columns-1>). The procedure requires the cell name of the bitcell to be used. To modify the sizing of the bitcell, must first create bitcell with desired sizing and pass its cell name to this procedure.

Arguments

libName	String specifying the library to place the cell.
cellName	String specifying the name of the cell.
rows	Integer specifying the number of rows in the SRAM.
columns	Integer specifying the number of columns in the SRAM.
cellName_bitcell	String specifying the cell name of the bitcell to be used.

Example
darCreateArraySchematic(“ViPro” “BCArray_128x16” 128 16 “BC”)
=> Creates a 128 row by 16 column array of bitcells called BCArray_128x16 in the ViPro library. This array has the following pins: WL<0:127>, BL<0:15>, and BLB<0:15>.

darUvaEceWLDriver_Schem.il

darUvaEceCreateWLDriverSchematic()

darUvaEceCreateWLDriverSchematic(
    libName
    cellName
    R
    cellName_WLDriver8
)

Description
- Creates the complete word line driver with the name specified by cellName in the library specficied by libName.
- The procedure takes the number of rows and the cell name of the 8 word line driver circuit (which means the 8 word line driver must already be properly sized).
- Inputs: PRE8_6<0:7>, PRE5_3<0:7>, WLclk<0:7> (the outputs of the three 3:8 predecoders, either buffered (by the ”n” chain) or unbuffered).
- Input/Outputs: WL<0:R-1>, where R is the number of rows in the SRAM.

darUvaEceBitSlice_Schem.il

connects SA, I/O, CD together

darUvaEceRowPredec_Schem.il

creates static AND decoder

darUvaEceColDecoder_Schem.il

creates 3:8 static AND decoder. Predecode 2 LSB then combine with MSB.

darTimingBlock_Schem.il

creates timing block, including 7 buffer chains

UvaEcePredecTiming_Schem.il

Connects row decoder, column decoder, timing block together

darUvaEceSchematic.il

THE PARAMETERS FILE, I.E. “SIZES.TXT”, MUST HAVE ATE LEAST ONE BLANK LINE AT THE END OF IT IN ORDER TO READ THE FILE CORRECTLY

darUvaEceSRAM_Schem.il

UvaEceCreateSRAMSchematic()

UvaEceCreateSRAMSchematic(
libName
cellName
R
C
DW
n
k
)

Description
Creates the entire SRAM schematic. Connects the 4 leaf nodes together: Bitcell Array, WLDriver, Predecode & Timing, Bitslice. Also connects the ”n” and ”k” buffer chains, if necessary.
- The inputs are the number of rows, the number of columsn, the word size, the length of the ”n” buffer chain, and the length of the ”k” buffer chain.
- Inputs: AR<0:8>, AC<0:2>, DATA_IN<0:%d>, WL_CTRL<0:1>, SA_CTRL<0:1>, CLK, WR
- Input/Outputs: DATA_OUT<0:%d>

Arguments

libName	String specifying the library to place the cell.
cellName	String specifying the name of the cell.
R	Integer specifying the number of rows in the SRAM.
C	Integer specifying the number of columns in the SRAM.
DW	Integer specifying the length of a word.
n	Integer specifying the number of “n buffers” after the predecoders.
k	Integer specifying the number of “k buffers” after the word line driver.