EE434
ASIC & Digital Systems

Project

Design of a High-Speed Low-Power 32-Bit Adder

Spring 2015

Dae Hyun Kim
daehyun@eecs.wsu.edu

Encounter

- Use *VKSA32_fm* to design a high-speed Kogge-Stone adder.
- Initial core utilization: 0.5~0.6
- Core-to-I/O distance: 5um
- Skip CTS & post-CTS optimization (because this is a combinational logic).
- Skip Fill insertion.
- Save the final design (use “saveDesign”, e.g., “saveDesign my_postroute_opt.enc).

Encounter to GDSII

- Use the following command in Encounter.
  - streamOut <gds_file_name> -mapFile ng45nm.map -libName <lib_name> -structureName VKSA32 -dieAreaAsBoundary -units 2000 -mode ALL

  - gds_file_name: you can decide it. (e.g., my_ksa_final.gds)
  - ng45nm.map is included in the project zip file.
  - lib_name: you can decide it.
    - This name should be the same as your library name in Virtuoso.

Encounter to Verilog

- Get the final verilog file.
  - Suppose you save your final design into my_postroute_opt.enc.
  - Then, it will automatically create a directory (my_postroute_opt.enc.dat).
  - See the contents of the directory. There is a zipped verilog netlist.
    - Ex) VKSA32.v.gz
  - Copy this into your working directory.
    - Ex) cp my_postroute_opt.enc.dat/VKSA32.v.gz .
  - Unzip it.
    - Ex) gunzip VKSA32.v.gz
  - Then, you will get VKSA32.v, which is the final verilog file.

Verilog to SPICE

- Source calibre.sh in lab1.tar.gz.
- Run the following command:
  - v2lvs –v <verilog> -l <library> -o <output>

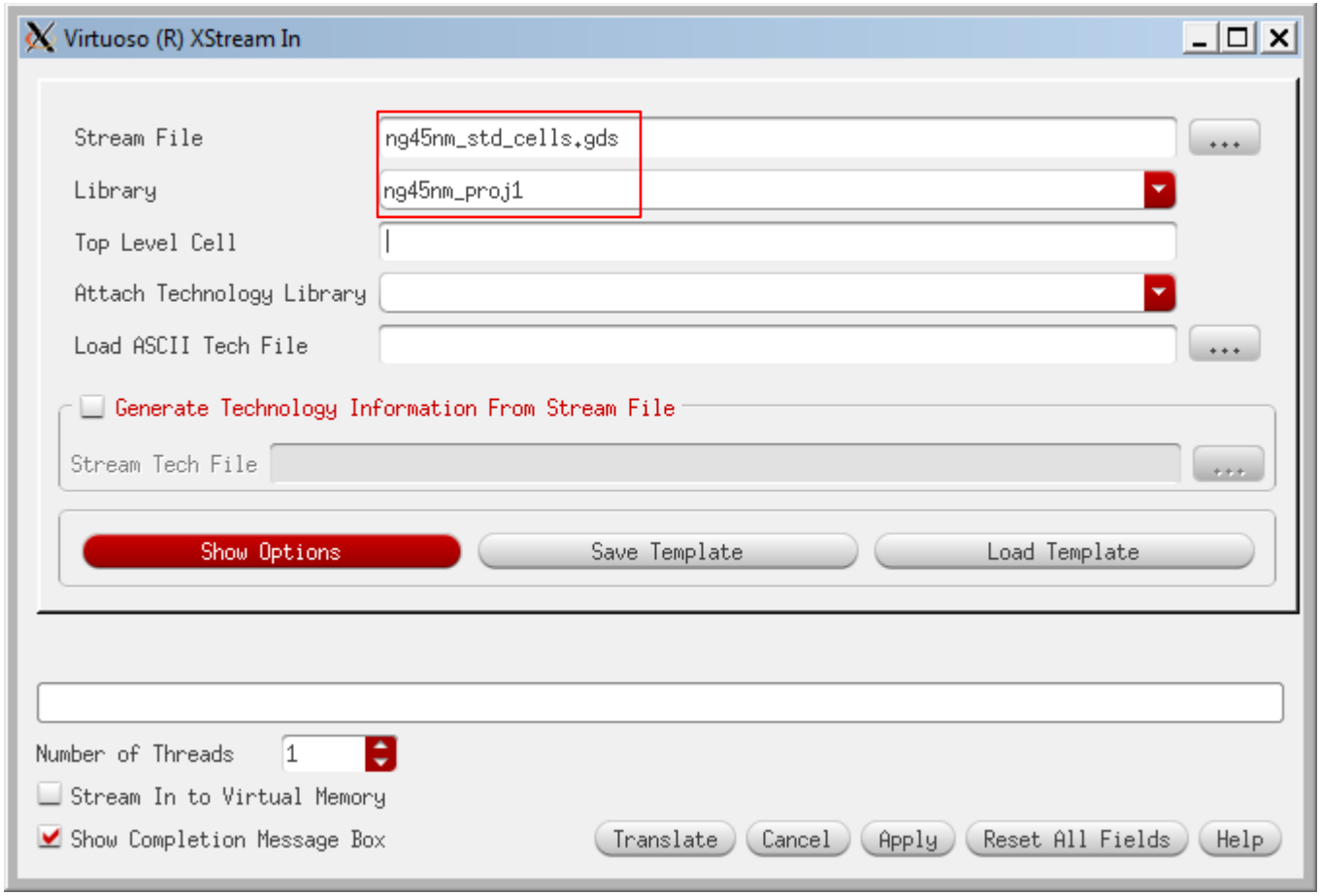
  - verilog: Your verilog netlist (e.g., VKSA32.v)
  - library: netlist library (use NangateOpenCellLibrary.v included in proj.tar.gz).
  - output: output file.

  - Ex) v2lvs –v VKSA32.v –l NangateOpenCellLibrary.v –o VKSA32.sp

Virtuoso

- Use tech_ng45nm.tf to create a library.
  - Note that the name of the library you create should be the same as the library name you specified in the GDSII export command.
- Import a 45nm standard cell library into your library.
  - In CIW, click File → Import → Stream.
  - Choose ng45nm_std_cell.gds in the Stream File text box.
  - Choose your library in the Library text box.
  - Then, click OK. This will import all the cells in the gds file.

Virtuoso

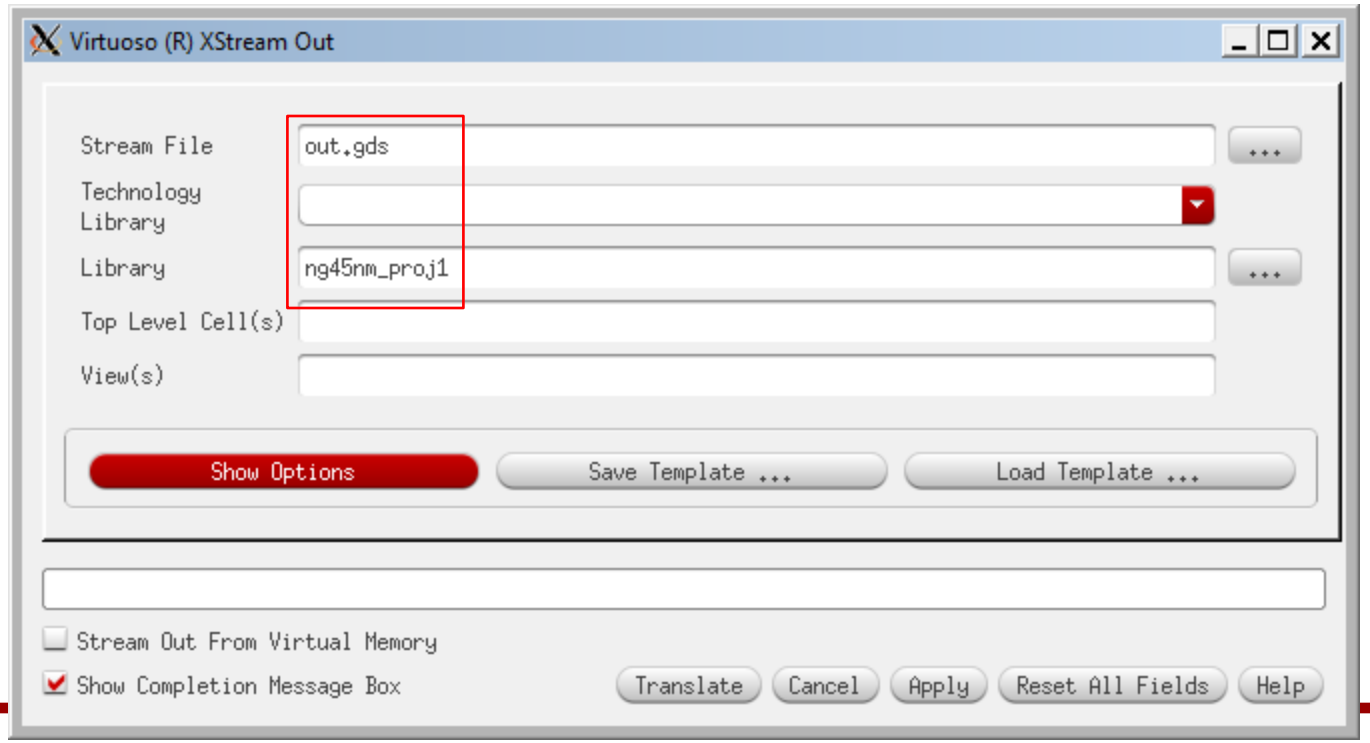

Virtuoso (R) XStream In
Stream File
ng45nm_std_cells.gds
Library
ng45nm_proj1
Top Level Cell
Attach Technology Library
Load ASCII Tech File
Generate Technology Information From Stream File
Stream Tech File
Show Options
Save Template
Load Template
Number of Threads
1
Stream In to Virtual Memory
Show Completion Message Box
Translate
Cancel
Apply
Reset All Fields
Help

Virtuoso

- Then, import your 32-bit adder gdsii into the same library.
- So, you will see all the standard cells and your adder in the library.
- Open VKSA32 to see the layout.

Virtuoso

- Let’s double check whether you can run LVS.
- Export your library into a gds file.
  - Leave “Top Level Cell(s)” blank. Then, it will export all the cells into a single gds file.



Virtuoso

- Run LVS.
- For the layout, choose the gds you just exported.
- The name of the top-level cell should be VKSA32.
- For the source, choose the SPICE netlist you made from the final verilog file.
- However, you have to add the netlists of all the standard cells into your SPICE netlist.
- So, open the SPICE netlist and add the following line into the netlist.
  - .include NangateOpenCellLibrary.spi

Virtuoso

- The following shows my netlist.
  - Note that the first line in a SPICE netlist is always a comment.

```
$ Spice netlist generated by v2lvs
$ v2013.2_18.13    Thu May 16 11:29:55 PDT 2013

.include NangateOpenCellLibrary.spi

.SUBCKT VKSA32 iA[31] iA[30] iA[29] iA[28] iA[27] iA[26] iA[25] iA[24] iA[23]
+ iA[22] iA[21] iA[20] iA[19] iA[18] iA[17] iA[16] iA[15] iA[14] iA[13] iA[12]
+ iA[11] iA[10] iA[9] iA[8] iA[7] iA[6] iA[5] iA[4] iA[3] iA[2] iA[1] iA[0]
+ iB[31] iB[30] iB[29] iB[28] iB[27] iB[26] iB[25] iB[24] iB[23] iB[22] iB[21]
+ iB[20] iB[19] iB[18] iB[17] iB[16] iB[15] iB[14] iB[13] iB[12] iB[11] iB[10]
+ iB[9] iB[8] iB[7] iB[6] iB[5] iB[4] iB[3] iB[2] iB[1] iB[0] iCin oS[32]
+ oS[31] oS[30] oS[29] oS[28] oS[27] oS[26] oS[25] oS[24] oS[23] oS[22] oS[21]
+ oS[20] oS[19] oS[18] oS[17] oS[16] oS[15] oS[14] oS[13] oS[12] oS[11] oS[10]
+ oS[9] oS[8] oS[7] oS[6] oS[5] oS[4] oS[3] oS[2] oS[1] oS[0]
XFE_OCPC1_np_L0_b10 BUF_X1 $PINS Z=FE_OCPN1_np_L0_b10 A=np_L0_b10
XFE_RC_226_0 OAI22_X2 $PINS ZN=np_L0_b7 B2=FE_RN_11_0 B1=iB[7] A2=iA[7]
+ A1=FE_RN_10_0
XFE_RC_225_0 OAI22_X1 $PINS ZN=np_L0_b11 B2=FE_RN_47_0 B1=iB[11] A2=iA[11]
+ A1=FE_RN_46_0
XFE_RC_224_0 OAI22_X2 $PINS ZN=np_L0_b8 B2=FE_RN_60_0 B1=iB[8] A2=iA[8]
+ A1=FE_RN_59_0
XFE_OCPC0_np_L0_b11 BUF_X1 $PINS Z=FE_OCPN0_np_L0_b11 A=np_L0_b11
XFE_RC_222_0 OR2_X2 $PINS ZN=FE_RN_154_0 A2=U_L5_b18/n1 A1=U_L4_b18/n1
XFE_RC_221_0 OR2_X4 $PINS ZN=ng_L5_b18 A2=FE_RN_154_0 A1=ng_L3_b18
XFE_RC_220_0 OAI22_X2 $PINS ZN=np_L0_b10 B2=FE_RN_80_0 B1=iB[10] A2=iA[10]
+ A1=FE_RN_79_0
XFE_RC_219_0 OAI22_X2 $PINS ZN=np_L0_b9 B2=FE_RN_19_0 B1=iB[9] A2=iA[9]
```

Virtuoso

- Run PEX to extract RC.

- As you know, this will generate three SPICE netlist files, but one of them is the top-most netlist file.

Design and Test

- Create a SPICE netlist to test it.
  - Instantiate the adder.
  - Add input signals.
  - Simulate and test it (just to see whether the netlists work correctly).

Design and Test

- The following shows my netlist to test it.

```
* The first line is always a comment line.
* Case-insensitive

* Control
.option post INGOLD=2

* Include the following file to load transistor models.
.include './45nm_PTM_HP_v2.1.pm'

* Parameters
.param Vsup = 1.0V

.include "./adder_pex.sp"

X1 VS29 VB19 VB21 VS21 VA20 VA21 VB20 VA22 VB22
+ VA29 VB18 VB29 VB0 VA30 VA1 VB30 VA17 VB1 VB17 VB28 VCIN
+ VA2 VS19 VB27 VB2 VA26 VB26 VB3 VB25 VA3 VB31 VA31
+ VS11 VS20 VB23 VB24 VB16 VA24 VA16 VB15 VS16 VA15 VS22
+ VS32 VB14 VA14 VB4 VB12 VA12 VA5 VB5 VB6 VA6 VB13 VA13
+ VB9 VA9 VB11 VA7 VB7 VA11 VB10 VB8 VA10 VA8 VA19 VA0
+ VA18 VS13 VS0 VS1 VS17 VS25 VA28 VS27 VS2 VA27 VS3
+ VS31 VS23 VA25 VA23 VS15 VS12 VS7 VS30 VS24 VS28 VS18
+ VS8 VS4 VA4 VS14 VS6 VS9 VS26 VS10 VS5 0 VXVDD VKSA32

* Load capacitance.
Cout0 VS0 0 10f
Cout1 VS1 0 10f
Cout2 VS2 0 10f
Cout3 VS3 0 10f
Cout4 VS4 0 10f
Cout5 VS5 0 10f
Cout6 VS6 0 10f
Cout7 VS7 0 10f
Cout8 VS8 0 10f
```

```
Cout31 VS31 0 10f

* Power supply
VVdd VXVDD 0 Vsup

* Input signals (independent voltage source)
* Format: <signal name> <node 1> <node 2> <signal>
* For the signal, we use a piecewise linear (PWL) source. Format: time1 value1 time2 value2 ...

VsA0 VA0 0 PWL 0p 0 100p 0 110p Vsup 1n Vsup
VsA1 VA1 0 PWL 0p 0 100p 0 110p Vsup 1n Vsup
VsA2 VA2 0 PWL 0p 0 100p 0 110p Vsup 1n Vsup
VsA3 VA3 0 PWL 0p 0 100p 0 110p Vsup 1n Vsup
VsA4 VA4 0 PWL 0p 0 100p 0 110p Vsup 1n Vsup
VsA5 VA5 0 PWL 0p 0 100p 0 110p Vsup 1n Vsup
VsA6 VA6 0 PWL 0p 0 100p 0 110p Vsup 1n Vsup
VsA7 VA7 0 PWL 0p 0 100p 0 110p Vsup 1n Vsup
VsA8 VA8 0 PWL 0p 0 100p 0 110p Vsup 1n Vsup
VsA9 VA9 0 PWL 0p 0 100p 0 110p Vsup 1n Vsup
```

Design and Test

- Now, add a PMOS and an NMOS transistors in the SPICE netlist.

- Size the PMOS and NMOS transistors.
  - Do not violate the timing constraint.
  - Minimize the size of the transistors.

Virtuoso

- Once the simulation is done, open your VKSA32 layout in Virtuoso and draw a PMOS and an NMOS transistors in the layout.
- Run LVS and PEX. (Do not run DRC).
- Re-simulate the PEX SPICE netlist.