Introduction to Simulation using IRSIM

1. Introduction

IRSIM is an event-driven logic level simulator for MOS circuits. It can simulate in two modes: ***switch*** and ***linear.*** The *switch* mode is useful for initializing or determining the functionality of a MOS network; while the *linear* mode is useful for determining transition times and gate delays. This page is a brief introduction to the basics of IRSIM and is not meant to replace the IRSIM manual. ***Please refer to the IRSIM man page for details.***

2. Using IRSIM for Combinational Logic

For the purpose of this simple tutorial we will use the 2-input XOR gate depicted in Fig. 1. This layout was drawn using *Magic* and saved in ***2xor.mag***

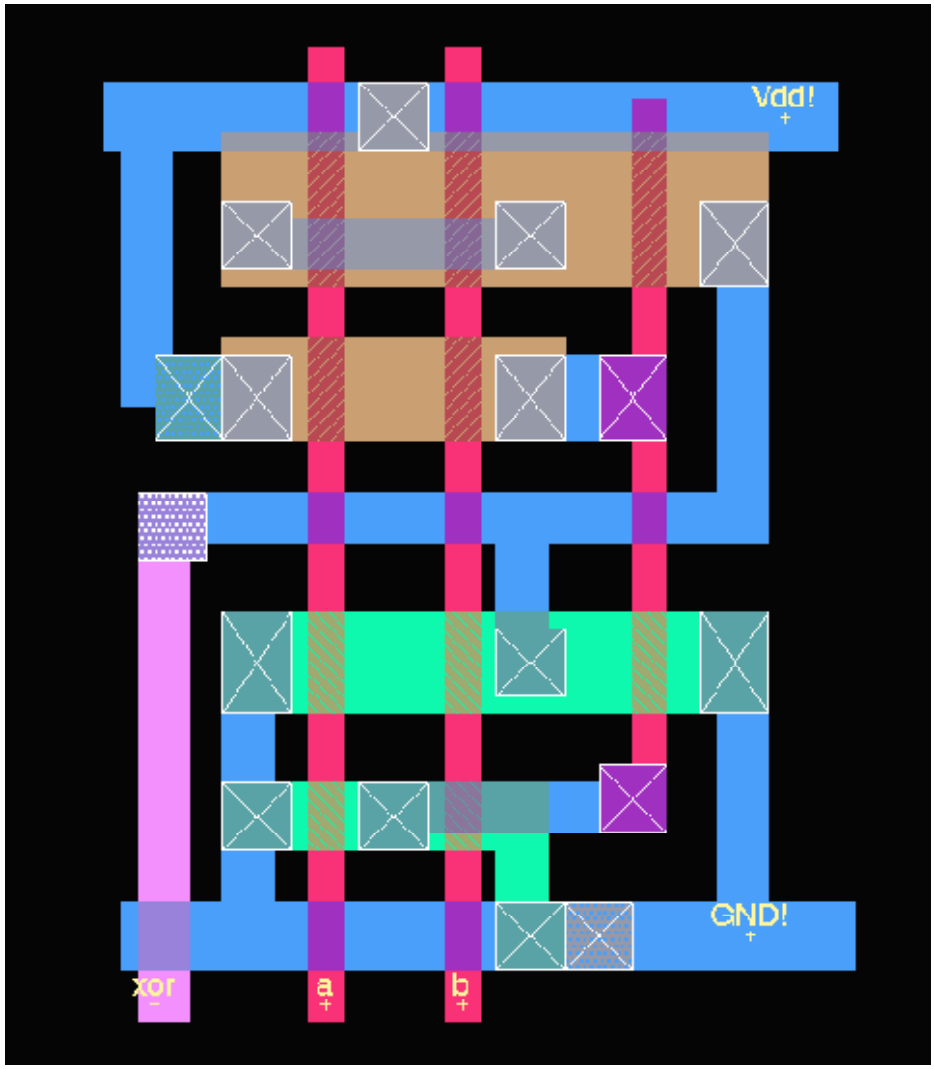

Figure 1: Layout of a 2-input XOR gate

IRSIM files are just text files that contain device connectivity and parasitics information pertaining to a circuit. Follow the procedure described below to create these files and use them to simulate the above 2-input XOR gate in *IRSIM.*

- ***Create an extracted version of your circuit:*** Make sure that you have labeled all power and ground nodes as **Vdd!** and **GND!**, respectively. You should also label all other nodes that you are interested in monitoring during simulation. Remember to save your layout file after you make any changes. Before exiting the layout editor *Magic* execute the long command `:ext all` to perform circuit extraction. This will create a file ***2xor.ext*** in your present working directory.

- ***Create a .sim file:*** After quitting *Magic* type the following command at the Unix prompt. This will convert the ***2xor.ext*** file to ***2xor.sim*** file

```
ext2sim -t! -t# 2xor.ext
```

  This will create the following files in your current working directory:

  - ***2xor.al*** -- list of all nodes that have different names but are connected together (i.e., aliased nodes).
  - ***2xor.nodes*** -- list of all nodes in the circuit.
  - ***2xor.sim*** -- the ***.sim*** file to used for logic simulation.

  Please see the man page for **ext2sim** for details.

- ***Create an IRSIM command script:*** Create a script file **(*2xor.icmd*)** containing the *IRSIM* commands listed below. You can also enter the commands interactively. However, having a script file enables us to simulate different versions of the same circuit without retyping the simulation commands.

```
model  linear
w   a  b  xor
vector   in  a  b
ana   a  b  in  xor

set   in  00
s   10

set   in  01
s   10

path   xor

set   in  10
s   10

set   in  11
s   10

path   xor
```

  The `**w'** command tells *IRSIM* to watch the nodes in following list. The `**vector'** command groups a list of inputs together so that they all can be set simultaneously. In the above script, input nodes **a** and **b** are grouped together in vector "**in".** The `**ana'** (analyzer) command causes *IRSIM* to display the listed nodes and vectors using its builtin graphical logic analyzer. The `**set in 01'** command is used to the set the vector "**in**" to **01** (i.e., **a=0, b=1**). The `**s'** command tells *IRSIM* to simulate for a certain period of time (in the above script it is 10ns). The `**path'** command shows the critical path for the last transition of a node (see

details in the output below).

- ***Run IRSIM:*** Run the *IRSIM* simulator using the command given below:

```
irsim  scmos1.2um.prm  2xor.sim  -2xor.icmd
```

In the above command ***scmos1.2um.prm*** is the parameter file for the 1.2um CMOS technology. Parameter files for other technologies are available in the */home/cad/vlsi/models/irsim*. *IRSIM* will respond with the following simulation output.

```
*** IRSIM version 9.4.2 ***
info: SU format --> using S/D attrs to compute junction parasitics
Warning: Aliasing nodes 'GND' and 'Gnd'
2xor.sim: Ignoring lumped-resistance ('R' construct)

Read 2xor.sim lambda:0.60u format:SU
9 nodes; transistors: n-channel=5 p-channel=5
parallel txtors:none
xor=0 b=0 a=0
time = 10.00ns

xor=1 b=1 a=0
time = 20.00ns

critical path for last transition of xor:
  b -> 1 @ 10.00ns , node was an input
  a_n29_n23 -> 0 @ 10.17ns   (0.17ns)
  xor -> 1 @ 10.51ns   (0.34ns)

xor=1 b=0 a=1
time = 30.00ns

xor=1 b=1 a=0
time = 40.00ns

xor=0 b=1 a=1
time = 50.00ns

critical path for last transition of xor:
  a -> 1 @ 40.00ns , node was an input
  xor -> 0 @ 40.32ns   (0.32ns)
```

Because of the `**ana'** command in the above script, *IRSIM* also opens up the *analyzer* window shown in Fig. 2.

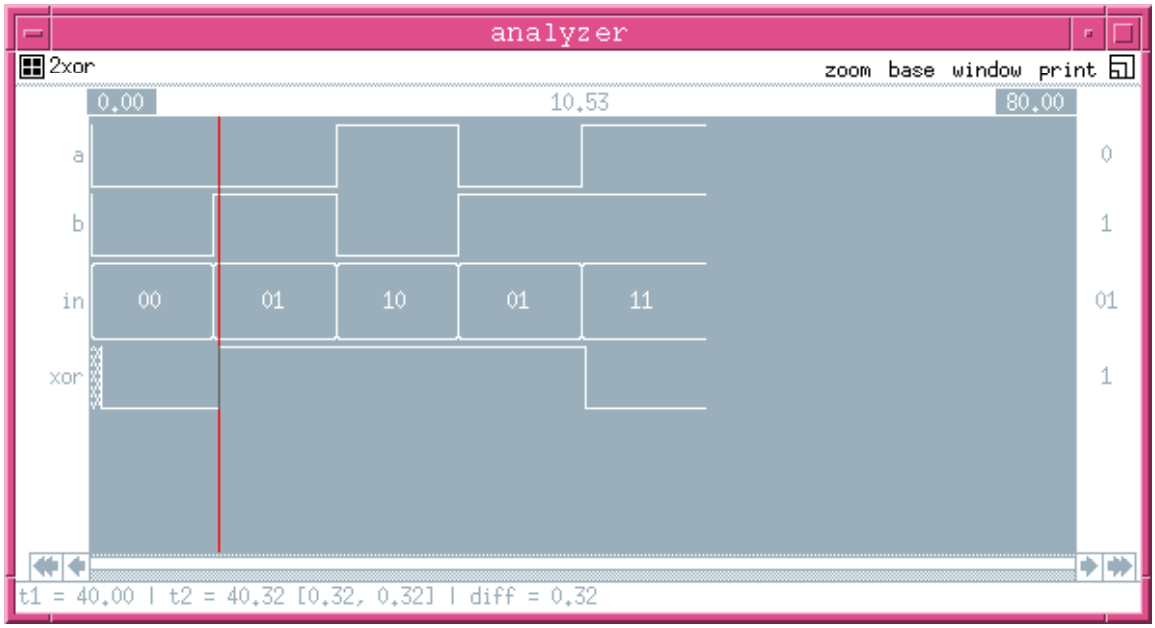

Figure 2: IRSIM Analyzer Window

- ***Using the IRSIM analyzer:*** You can use the analyzer window to do time measurement, print outputs, change view, etc.
  - To find time of an event click in the analyzer's waveform window to get a vertical cursor. The time where this vertical cursor is is displayed above.
  - You can also determine the period between two events: Select "delta T" from "window" menu. Click on the first event and then click on the second event. The period between these events is displayed as "diff" at the bottom of the window as shown in Fig 2 above.
  - For a multi-bit vector, you can change the *base* used for the display using the "base" menu.
  - You can change the width of view by using the "zoom" menu.
  - To print the waveform to a postscript file, go to the "print" menu and select "file". You will be asked for a filename. Hit return to select the default filename shown. Once you exit from *IRSIM* you will have a file ***2xor.ps*** in your current working directory. You can print this file on any postscript printer. The output file for the above simulation is shown in Fig. 3.

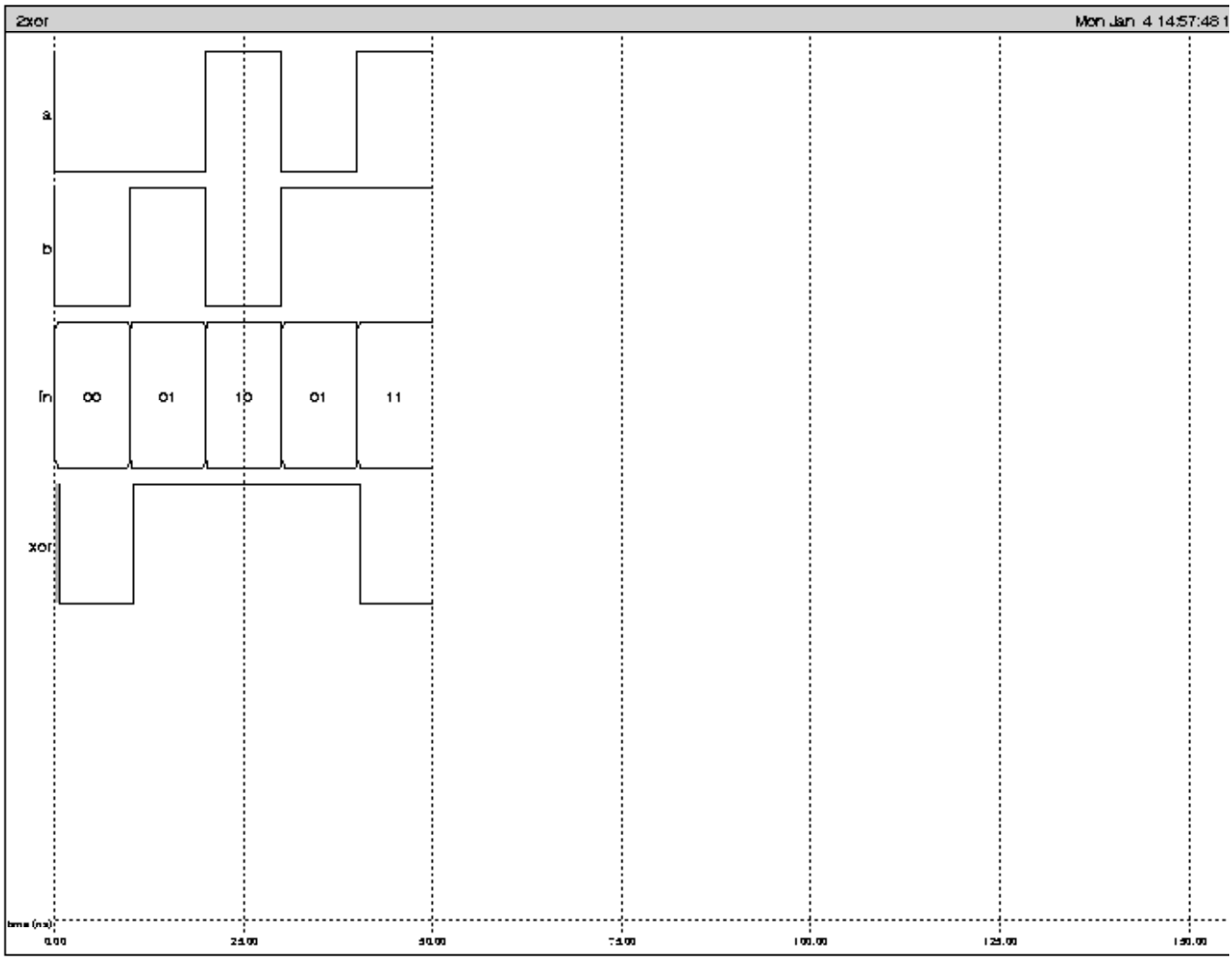

Figure 3: IRSIM simulation output file created using the "print" menu command

- ***End IRSIM Simulation:*** You can end your simulation anytime by typing in the **quit** command at the `IRSIM>` prompt.

3. Using IRSIM for Synchronous Sequential Logic

Example 1: The *2-bit up/down counter* described in the *Meg*tutorials can be simulated in *IRSIM* using the ***counter.icmd*** script given below:

```
model   linear

clock  clka         0  1  0  0
clock  clkabar   1  0  1  1
clock  clkb         0  0  0  1
clock  clkbbar   1  1  1  0
```

```
vector   state   StBit{0:1}*
vector   control   up  down
vector   count   b1  b0

ana   clka  clkb  StBit1*  StBit0*  state  restart  up  down  control  b1 b0
count

@  reset.icmd

|   16 clock cycles
V   restart   0  1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
V  control  10 10 00 00 10 10 10 10 10 00 00 01 01 01 01 01
R

|   2 clock cycles
V   restart  0 0
set  control  10
R
```

The above simulation script includes several new features. The **`clock'** commands are used to describe a 2-phase non-overlapping clock. The nodes in the definition of vector "**state**" are described using a compact notation **StBit{1:0}*** instead of **StBit1* StBit0***. The above script invokes another script ***reset.icmd***. This nesting of other script files enables you to build specialized test sequences and include them as and when needed in the main script file. The script also demonstrates the use of the **`V'** command (as an alternative to the **'set'** command) to provide the control input. The **`V'** command defined to ***restart*** the counter is intentionally asserted on the second clock cycle so that the power-up of the counter in an undefined state can be observed. The ***reset.icmd*** file is described below.

```
V   restart  0   1   0
V   control  00  00  00
R
```

The results of the *IRSIM* simulation of the *2-bit up/down counter* using the above script are depicted in Fig. 4.

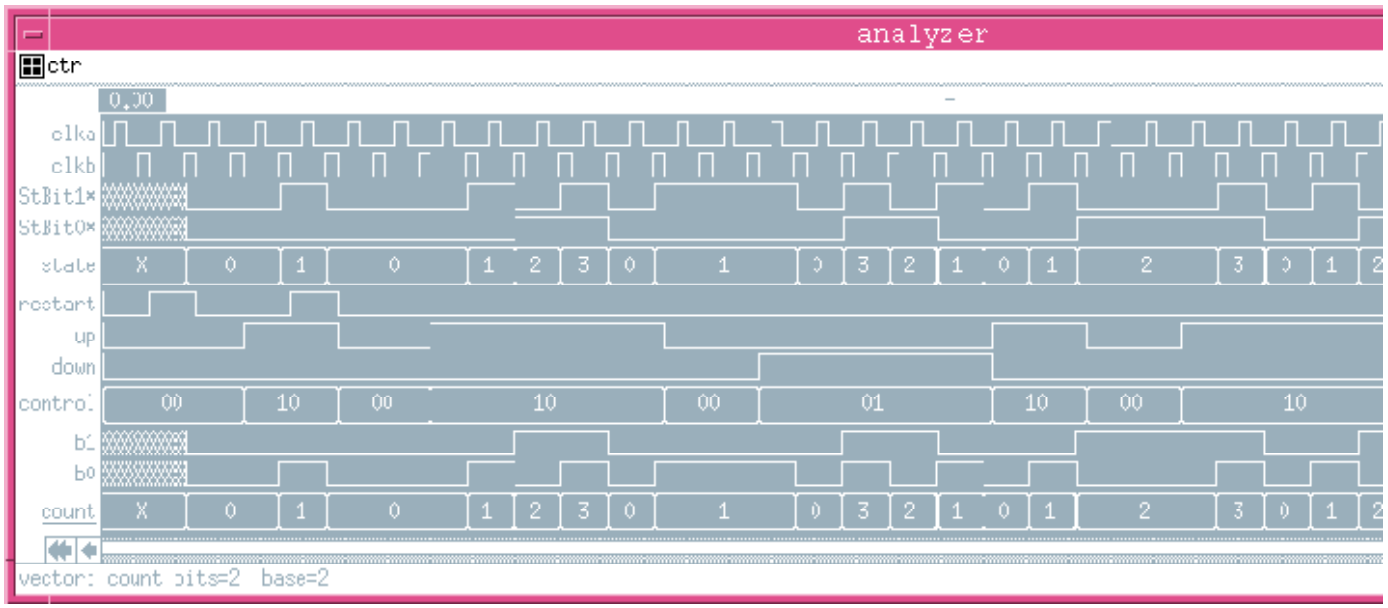

Figure 4: IRSIM simulation of the 2-bit up/down counter

Example 2: The *Bit Serial Adder* described in the *Pe*gtutorials can be simulated in *IRSIM* using the ***sadder.icmd*** script given below:

```
model linear

clock  clka         0  1  0  0
clock  clkabar   1  0  1  1
clock  clkb         0  0  0  1
clock  clkbbar   1  1  1  0

vector  operands  a b
vector  state     St1 St0
ana     phi1 phi2  reset  state a b  operands  sum

@ reset.icmd

V  reset    0  0  0  1  0  0  0  0  0  0  0  0
V  operands 10 01 11 00 01 10 11 10 01 11 00 01
R
```

The ***reset.icmd*** file is described below.

```
V   reset      0    1    0
V   control  00  00  00
R
```

The results of the *IRSIM* simulation of the *Bit Serial Adder* using the above script are depicted in Fig. 5.

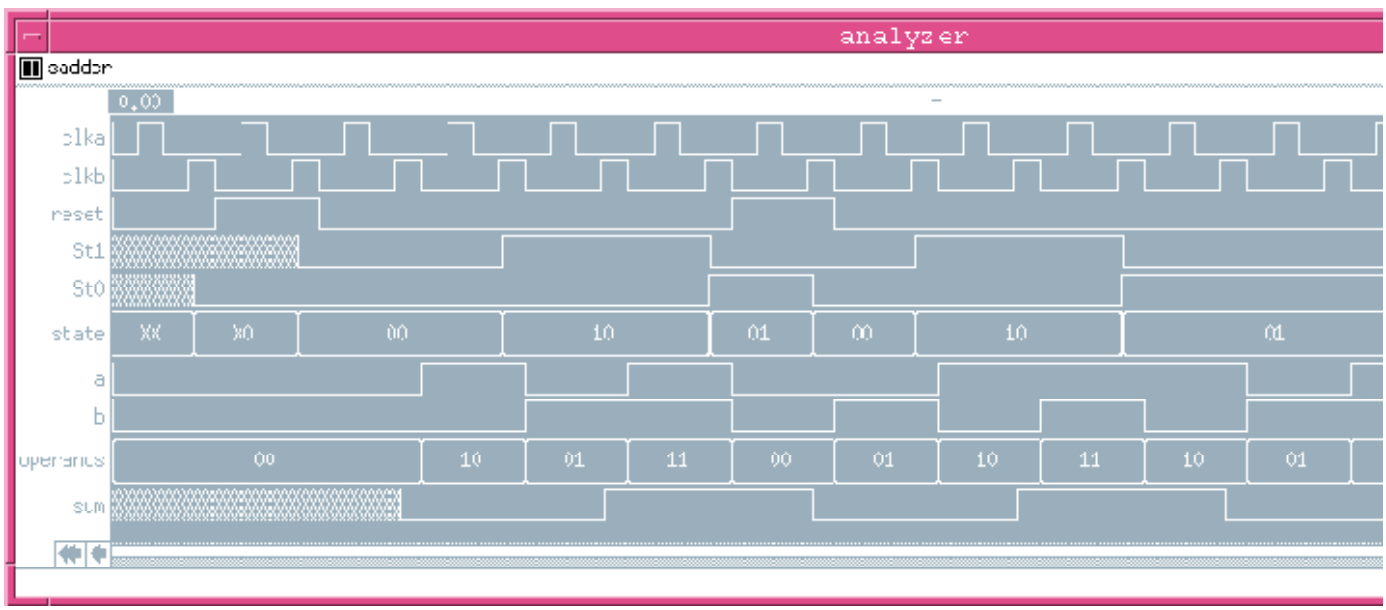

Figure 5: IRSIM simulation of the Bit Serial Adder

Return to EE6325 homepage