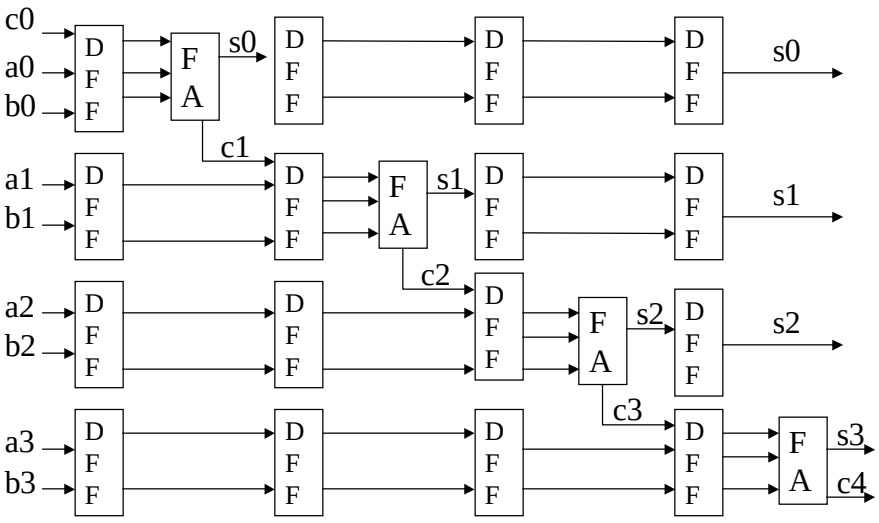
Bit-Pipelined Adder

The diagram below illustrates a bit pipelined adder (4 bits wide)

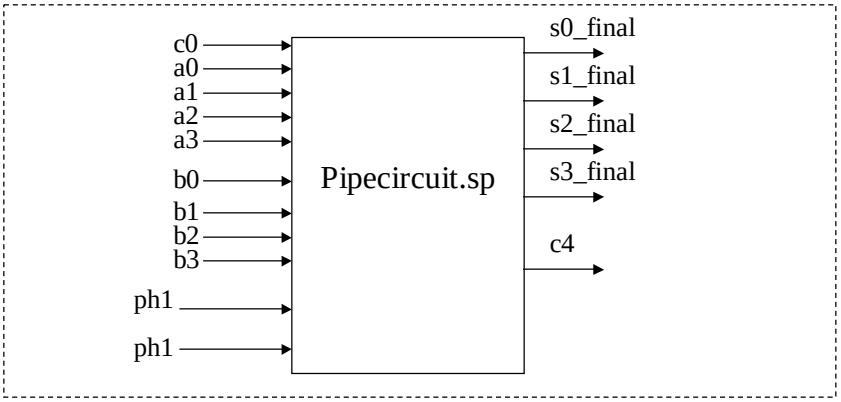


BR 6/00 1

Task

Build a domino pipeline with a 2-phase clock to solve this problem

testbench.sp



BR 6/00 2

Testbench

The *testbench.sp* provides the inputs and captures the outputs.

To generate a set of input values for the *testbench.sp*, a text file with a set of input vectors must be provided. A perl script is provided that can generate a set of random vectors:

```
% gen_random.pl 30 > testvec.txt
```

will generate a file with 30 random vectors in it.

This file has a simple format, you can generate vectors manually (strongly suggested) in order to test worse case vector sequences.

Testbench (cont)

To convert 'testvec.txt' to the format required by testbench.sp', do:

```
% rm –f testbench.a2d
% gen_test.pl  vec_filename  clock_per  clock_overlap  offset
```

The 'vec_filename' is the file that contains the input vectors.
The 'clock_per' is the clock period in ps.
The 'clock_overlap' is the overlap of the two clocks in ps.
Offset is a floating point number that indicates how late in the evaluation phase to check the outputs.

ie.

```
% gen_test.pl    testvec.txt    2000   200  0.5
```

The spice input file produced by this script is called 'testbench.d2a'. It also produces a parameter file called 'tb_parm.sp'.

Comparing Expected Results vs. Measured

During simulation, output results are captured in the file 'testbench.a2d'.

The script 'gen_test.pl' can also be used to compare expected vs. measured results. If the file 'testbench.a2d' exists when 'gen_test.pl' is run, then the vectors passed to 'gen_test.pl' are compared against the results in the 'testbench.a2d' file and both correct matches and mismatches are printed to the screen.

Rankings of Designs

I will take the fastest design (smallest clock period) and assign it a value of 1.0 .

I will take the most design with the lowest power and assign it a value of one.

Each design will be assigned a score based on your clock period and power value normalized to the fastest design and lowest power design.

The lowest possible (best) score is a '2.0' (design is both fastest and uses the least power). The higher your score, the worse your design. The upper third of the class will get 15 pts added to any test grade. The middle third will get 5 pts added to any test grade. The lowest third gets no extra points.

Misc

- You will need to experiment some with the third parameter to the 'gen_test.pl' script to determine what works best for you.
- Look the waveforms for PH1 and PH2 to make sure you understand how I define 'overlap'.
  - Negative values for the overlap parameter will work.
  - I assume Phi1 goes to the first stage, Phi2 to 2nd, and they alternate.
- The testbench currently does not measure power. I will correct this.
- If your design fails any test vector that I test it with, then your design is not eligible for any extra points. I will feel free to enter any test vectors in any sequence.
- You will have to provide me with parameters to use for the clock period, clock offset, and checking of output values.

Misc (cont)

- You can use any full adder design that you want. The DCVSL used earlier will work if you convert it to dual-rail domino. Feel free to look in the literature for other designs.
- The testbench does not provide complemented inputs. You must generate them.
  - Be careful – you can't just use a static inverter to generate the complement inputs.
- Do not modify the testbench – I am going to run submitted circuits using this testbench.
- A pipeline stage cannot be empty. It must at least have one domino buffer.
  - You may need to be concerned about racethru conditions.

Misc (cont)

- Do not use any GEO parameters in your models. This will cause worst case assumptions for source/drain capacitance.
- Be careful of charge sharing effects!!!
- The due date for this lab will not be extended. You are being given plenty of time to do this exercise.