FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Projeto de Circuitos VLSI

Multi-Stage Interconnection Network

Project Report

Group 8

Francisco Meireles
up202007382

Lucas Feijó
up202000145

Professor

João Canas Ferreira

May 15, 2024

1 Module characterization

1.1 Module Function

The module designed implements an 8x8 Beneš interconnection network (Fig. 1), enabling routing data from 8 input ports to 8 output ports with a routing algorithm controlled by control signals. The base for this 8x8 interconnection network is a 2x2 butterfly topology block that allows the connection between two inputs and two outputs be set on the fly, similar to a multiplexer, made possible by the use of 4 CMOS transmission gates. These analog gates function as switches, allowing current to flow bidirectionally based on a control signal S. On a transistor level, each transmission gate (TG) is composed of a parallel of a nMOS and pMOS controlled by a signal S and its complement, in a double rail logic [1], allowing the passage of both strong 0s and 1s.

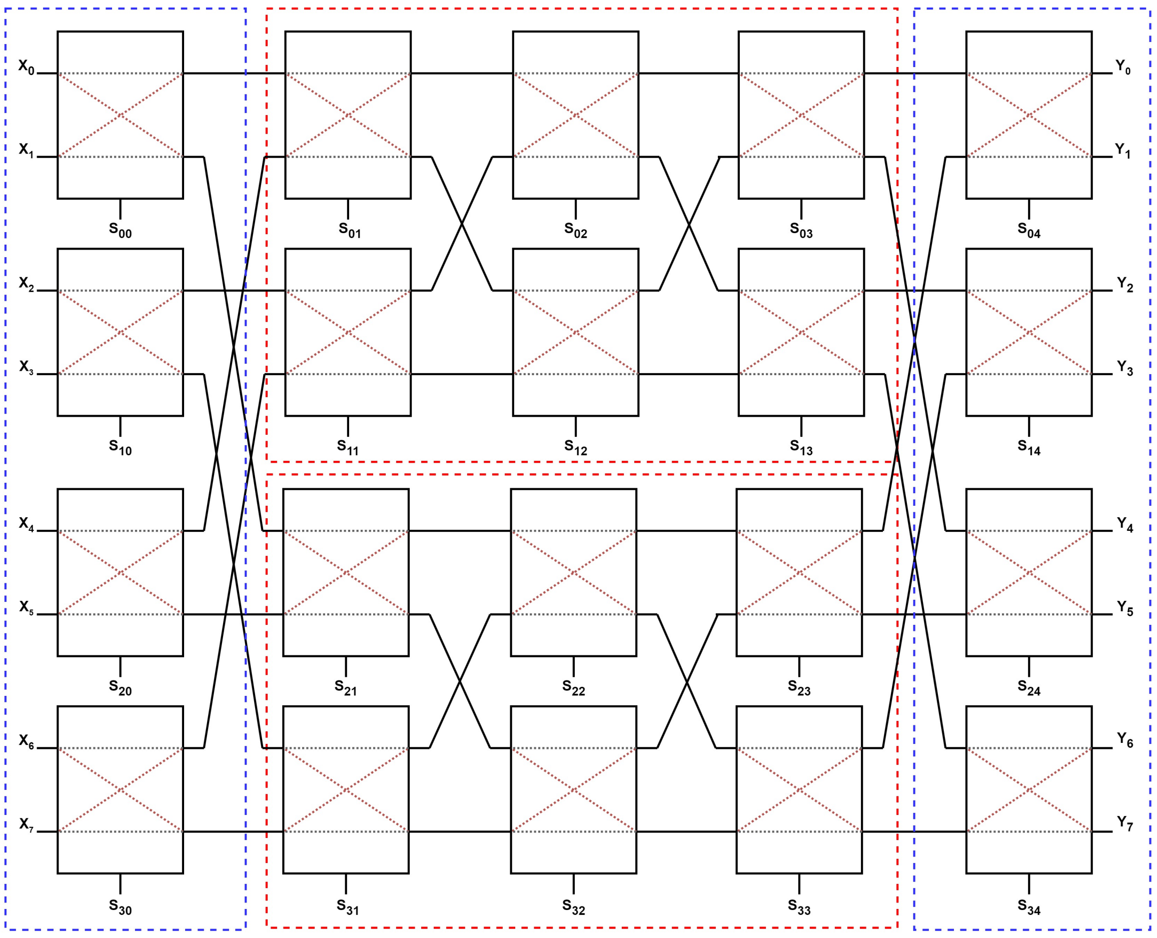

Figure 1: 8x8 Beneš interconnection network

The 8x8 module can be divided into three main parts: input and output, both composed of a column on 4 2x2 blocks, highlighted in blue on Figure 1, and intermediate interconnections, composed by two 4x4 Beneš network blocks, highlighted in red. Each 2x2 butterfly in the module has its own control signal S, with the overall routing being set by a 20 bit long control input. Any input signal can be routed to any output port with four possible routes between any input and output ports. The truth table for a 2x2 butterfly is shown in Table 1.

S	Y_0	Y_1
0	X_0	X_1
1	X_1	X_0

Table 1: 2x2 Butterfly truth table

1.2 Transistor Level Design

With the 2x2 butterfly as the base building block for the whole module, design started on characterizing and optimizing its functioning with delay and sizing in mind. After designing our TG and validating its correct functioning, a measurement algorithm was used to determine the best gate size and delay ratio. With the TG characterized, the scope was enlarged to encompass the whole butterfly construction. The need for a complement control signal made it necessary to include an inverter on each 2x2 block. Another measurement algorithm was used to determine the balanced inverter gate sizes for the load created by the four TGs designed. All sizing was made with multiples of the minimum width allowed by the gpdk045 technology, 120 nm. Table 2 shows the ratios used for sizing both inverter and transmission gates. The rest of the 8x8 Beneš module was designed using this 2x2 butterfly.

Component	**nMOS**	**pMOS**
Inverter	$2 \times W_{min}$	$3 \times W_{min}$
Transmission Gate	$3 \times W_{min}$	$1 \times W_{min}$

Table 2: Width multiples used for inverters and transmission gates on 2x2 butterflies

1.3 Module Area

A height of 7,325 μm and width of 12,935 μm makes the module have a total area of 94,748875 μm^2. Design approaches like clustering the TGs of each 2x2 block into single cells with multiple fingers, breaking the layout design into two main blocks, 2x2 and 4x4, with different layouts and space optimizations, and placing interconnections as close as possible within design rules made possible to make the layout as small as possible with our experience and knowledge.

1.4 Measurement Conditions

Measurements were done using a chain of two balanced inverters, with a nMOS width of W_{min} and pMOS width of $2 \times W_{min}$, to drive each input and an inverter 4 times the size of the driver as a load on each of the output ports. The Spectre analyses were of type transient, temperature of 27° C, and rise and fall times of 1 ps for the input signals. The propagation delay was measured between the input port of the module, after the driving inverter chain, and the output port, before the load inverter, as specified by the project guidelines. For each simulation done, all control signals have fixed values and only the input analyzed have transitions.

1.5 Module Performance

Using the the AT^2 criterion to evaluate the module's performance, with an area of 94.785.875 nm^2 and maximum propagation delay of 0,629795 ns, we got an AT^2 of 37.581.358,8349 $nm^2 ns^2$.

2 Simulation and Validation

2.1 Simulation

With the module layout finished with no DRC or LVS errors, Quantus was used to extract a RC model to be used on the testbench schematic and its netlist exported. With 20 control signals and 8 input ports combinations, simulating each possible scenario using ADE was unfeasible, so a Python[2] application was developed to generate thousands of relevant netlists and its possible input and control combinations, with its measurement specification files. To run autonomously and manage all the files created, a Bash script was also developed, making the measurement and validation process straightforward. The output files generated for each netlist contains the delay times between the input of interest and all outputs, validating the correct function of the circuit for the desired route defined with the control input while measuring the propagation delay. This propagation delay was measured between the input port of the module, after the driving inverter chain, and the output port, before the load inverter, as specified by the project guidelines.

2.2 Validation

As explained in the last section, our Python application generates netlists and measurement specification files with different input and control signal combinations. For each netlist created, it is verified the delay between the relevant input and all outputs, even ones that are not expected to have output signals, which results in a NaN (Not a Number) data type, indicating that with that particular control signal value there is no connection between the observed input and tested output. With this exhaustive testing for all relevant combinations, it is guaranteed that the module designed works as expected.

2.3 Maximum Propagation Delay

Determining the Maximum Propagation Delay was not an easy task. Initially we considered using our Python code to create a netlist for each possible configuration of the circuit, however considering that there are 20 control signals that can vary for each of the 8 inputs X there would be around 8,3 million (8×2^{20}) possible configurations, given the time it takes to run each simulation, and having in mind the hardware and software limitations, we considered this approach to be impractical. In the second approach we decided to analytically determine the longest possible physical path in the layout, which in this case would be between the input cell containing inputs X0 and X1, controlled by S00, and the output cell with outputs Y2 and Y3, controlled by S14, passing through cells controlled by signals S21, S22 and S33. We then analyzed each of the possibilities individually and concluded that the path that leads to the longest propagation delay is from X0 to Y3. So, once the path had been determined, using Python code where the signals S that influence the path from X0 to Y3 were given fixed values, ensuring that only this path would be tested, we created a netlist for each possible configuration, varying the remaining 15 control signals, resulting in 32.768 files (2^{15}). Once all the netlists had been created, we developed a Bash script that ran a simulation for each one of them and saved the highest delay value measured between the input and the output for the case of fall-to-fall and rise-to-rise. The values for the control signal bits that generates the maximum propagation delay can be found on Table 3.

Bit	**S00**	**S01**	**S02**	**S03**	**S04**	**S10**	**S11**	**S12**	**S13**	**S14**
Value	1	0	0	0	1	0	0	0	1	0
Bit	**S20**	**S21**	**S22**	**S23**	**S24**	**S30**	**S31**	**S32**	**S33**	**S34**
Value	0	0	1	0	0	1	0	1	0	0

Table 3: Control signal bit values for the maximum propagation delay

With the combination of the control signal and the inputs X1 and X4 with logic high values, the t_{phl} from X0 and Y3 is equal to 0,47075 ns and t_{plh} is 0,78884 ns, resulting in a maximum propagation delay of 0,629795 ns.

References

[1] Harris D. M. Weste N. H. *CMOS VLSI Design: A circuits and systems perspective*. Pearson, 2015.

[2] Francisco Meireles and Lucas Feijó. Vlsi. https://github.com/Meirelez/VLSI/tree/main , 2024. Accessed on 15-05-2024.