Nanoscale Implementation of G-Share Branch Predictor

Adithya Kommini, Wei-chung Chen

Dept. of Electrical and Computer Engineering, University of Massachusetts, Amherst, USA

***Abstract*— The performance of the processor depends on its ability to predict the conditional branches, so that it have maximum usability of the pipelining. In this project implementation of a G-Share branch predictor in N3ASIC is performed. The design is based on N3ASIC, a nanofabric using combination of crosspoint nanowire FETs and integration with metal interconnects[1]. The usage of N3ASIC will have very good impact on the area and the performance of the predictor when compared to the conventional CMOS implementation. These impacts have studied and compared in this project.**

Keywords—G-Share; Branch Prediction; N3ASIC; Nanoscale;

I. Introduction

The factors that determine computer performance is the degree to which the implementation can take advantage of instruction-level parallelism. The most important part of implementing the instruction level parallelism was able to know where the conditional branches points even before they are executed in the ALU. To do that there are many branch prediction techniques being proposed in the literature. The basic prediction technique was the Bimodal branch prediction technique, which makes a prediction based on the direction the branch went the last few times it was executed. This method don't use the global branch behavior in prediction, which results in local branch prediction. To have a high efficiency of branch prediction we uses the combined history of all recent branches in making a prediction. This technique will be referred to as *global* branch prediction. The local technique works well for branches with simple repetitive patterns. The global technique works particularly well when the direction taken by sequentially executed branches is highly correlated. In this project we tried to implement of the global branch prediction called Gshare branch predictor. This method uses a single shift register Global History Register (GHR), records the direction taken by the most recent conditional branches.

II. G-Share Branch Predictor

The Gshare branch predictor is implemented by using the global history stored in the GHR, a XOR gate and a Pattern History Table (PHT) as shown in figure.1. The GHR stored global history was XORed with the branch address to get an index which is used ti index the PHT. In this project the GHR implemented was a 4 bit shift register using Latches and Multiplexer, which shifts the global prediction bits after every execution of a conditional branch and stores the new branch behavior in the GHR. The GHR value was fed to a 4 bit XOR circuit which GHR value and the branch address to get the index of the prediction bit corresponding to that branch, which is stored in the PHT. The PHT was implemented by using 16 9TNWRAM cells which stores the prediction bits corresponding to the index, which in-turn depends on both the address and the global branch history.

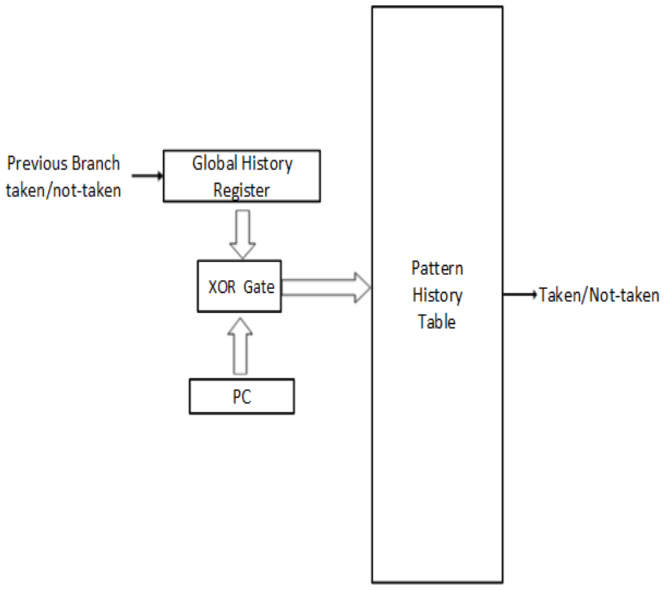

Fig. 1. Block Diagram G-Share Branch Predictor.

III. Designing a G-Share Branch Predictor

The designing of G-Share branch predictor in this project was done to have optimal layout implementation. Different blocks of the G-Share branch predictor was designed as follows.

A. *GHR Shift Register*

The GHR shift register used in this project is a 4bit shift register which shifts the bits by a bit when an executed branch outcome arrives at the GHR to accommodate the new value. The Shift register is implemented using a 2×1 multiplexer with a select line which decides to shift the bit or not, a buffer with a feedback which is used a latch to store the bit. The block diagram of the GHR was shown in the figure.2. The N3ASIC implementation of the shift register is designed as shown figure.3. The shift register works based on the shift signal which goes high on arrival of executed branch output from the ALU. If there is no outcome from the ALU shift will be low and the present bit in the latch is again given as input, otherwise the branch outcome was stored in the first bit of the shiftregister. The rest of the bits where the latch output from the previous bit is given as input to the next bit structure, are shifted by a bit and the last bit was discarded.

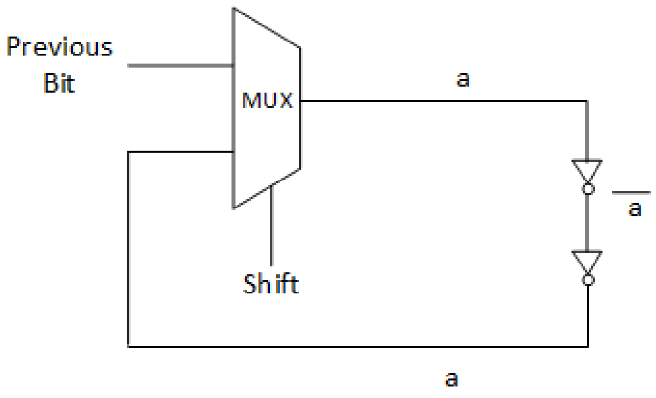

Fig.2. Block Diagram for single bit of Shift Register

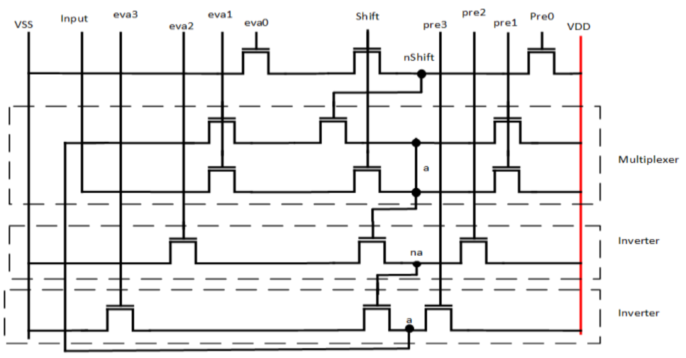

Fig.3. Dynamic Implementation for single bit of Shift Register

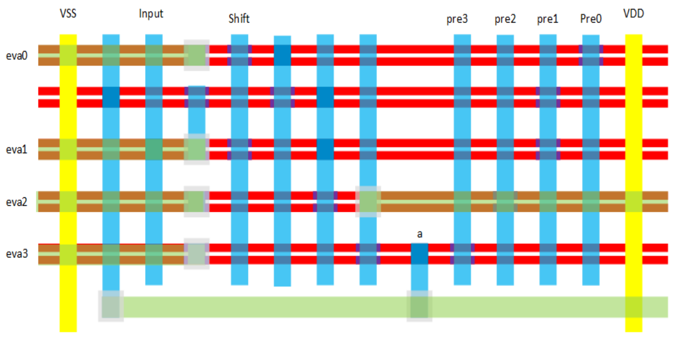

Fig.4. Layout of single bit of Shift Register for N3ASIC

B. *XOR Gate*

The 4 – Bit XOR gate was used to calculate the index to the PHT which is used to retrieve the prediction bit from the PHT. The index was calculated from the GHR and the address (last four bits) of the branch which is to be predicted. The XOR gate was implemented using four 1 bit XOR gates which is shown in the figure.5. This 1 bit XOR gate uses two NAND gates which are connected in parallel to each other to form a XNOR gate. In single precharge and evaluate cycle the XNOR operation was done on two input bits which are obtained from the GHR. In the next cycle an inverter is implemented on XNOR output using another precharge and evaluate cycle which gives an XOR operation output of the signals. The layout of the XOR gate was shown in Figure.6, using the nanowire design rules.

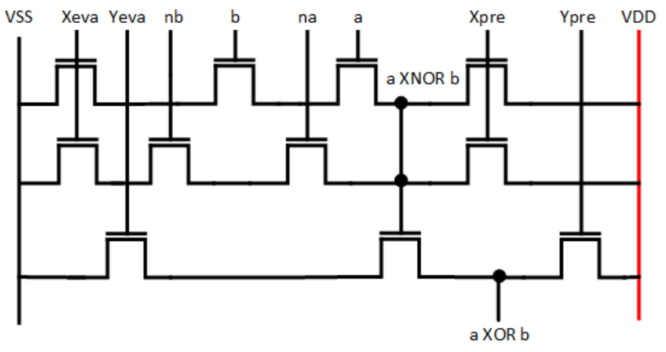

Fig.5. Dynamic Implementation of 1-bit XOR gate

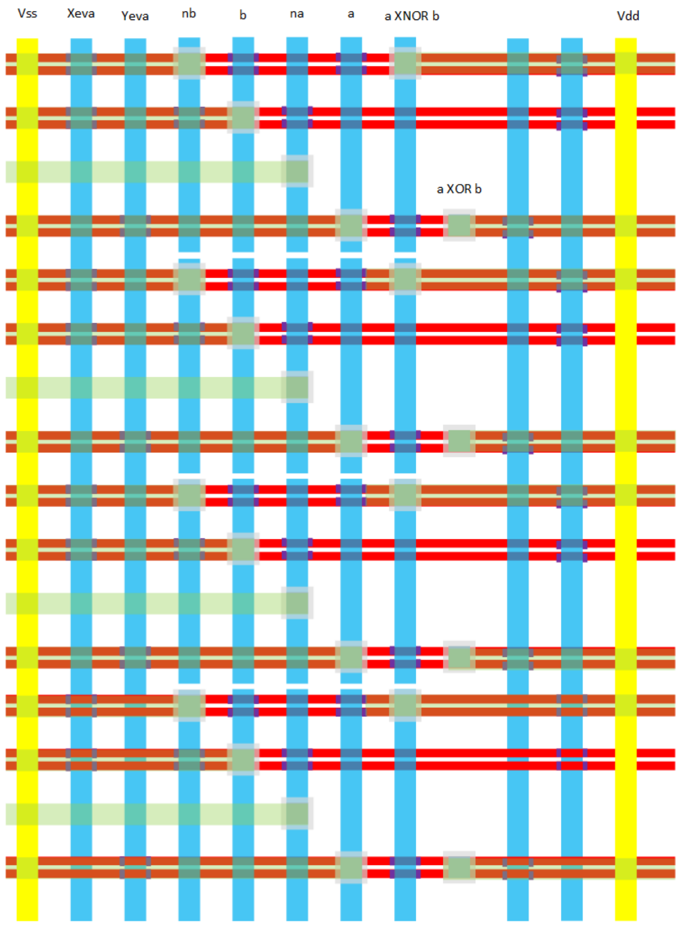

Fig.6. Layout Implementation of 1-bit XOR gate

C. *Decoder*

The index produced by the XOR is used to access the prediction bit from the PHT which is made of the 16 SRAM cells. To decode the index and to point to the corresponding SRAM we use a 4 to 16 decoder. This was implemented using 16 NOR gates like shown in the figure.7. The NOR gate was operated for a single precharge and evaluation cycle to get the decoded output. This decoder asserts the corresponding line pointed by the index given by the XOR gate, which activates the read operation for the corresponding SRAM. The layout of the single NOR gate used in decoder is shown in figure.8.

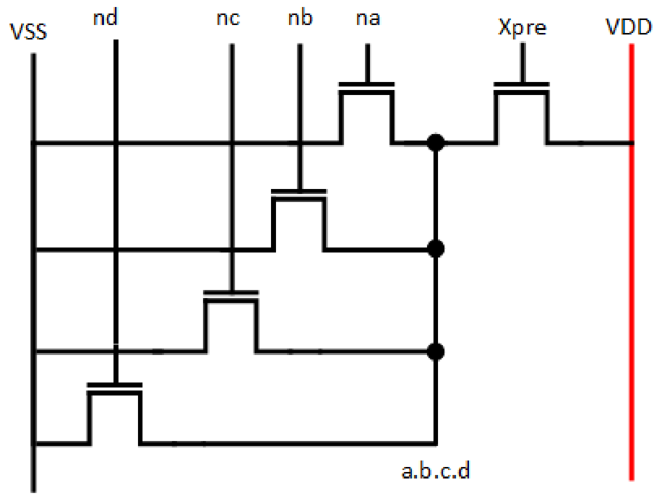

Fig.7. Dynamic Implementation of NOR gate.

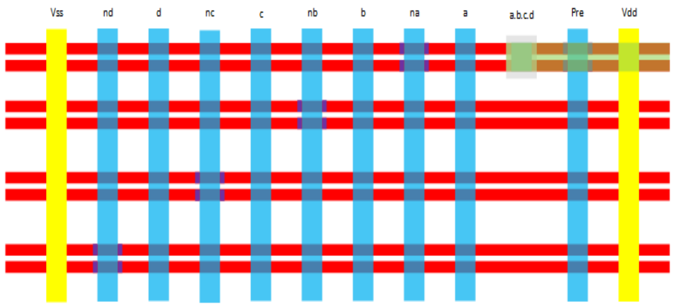

Fig.8. Layout Implementation of NOR gate.

D. SRAM

In this project we used a 9 transistor nanowire SRAM as shown in with additional transistor for read operation. The 9T Nanowire RAM circuit is as shown in fig. 9

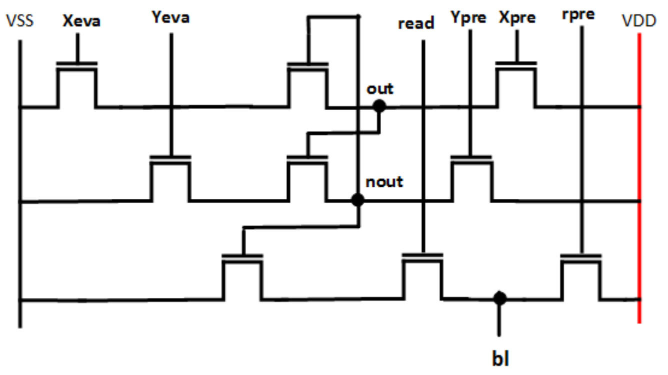

Fig.9. Dynamic Implementation of SRAM.

The storage element is implemented with a cross-coupled NAND structure using two 2C-xnwFETs. Complimentary states are stored in out and nout.

Write:

A pre-charge and evaluate transistor in each path controls the write and memory restore operations through non-overlapping clock signals xpre, ypre, xeva and yeva. The write operation is performed in two cycles of pre-charge and evaluate of nodes out and nout. To write a bit "1", the out node is pre-charged using xpre clock. However, the xeva signal is gated to hold the state of out at 1. This is followed by pre-charge and evaluation of nout. To write a 0 in out, while xeva is asserted to after pre-charging out, yeva is gated after pre-charging nout to store a state 1.

Read:

During read operation the bitline bl is pre-charged to VDD using the rpre transistor. The pre-charge is released and the read signal is asserted. Depending on the state stored in nout, transistor with nout input is turned ON or OFF to read the bit stored on out through the bitline.

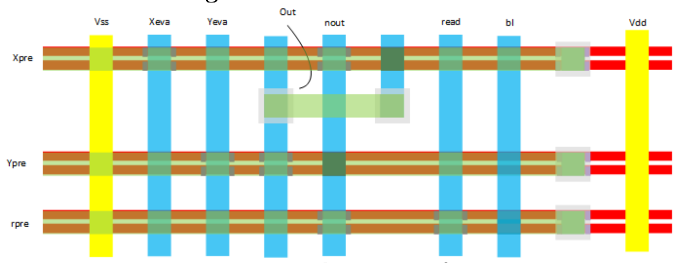

Fig.10. Layout Implementation of SRAM.

IV. Implemenatation and Results

The simulations are done in HSPICE using the N3ASIC design files. The corresponding CMOS implementation was done in Synopsys design compiler. Simulation results are shown in Fig 11,12,13. The area, power and delay calculatins done in the HSPICE for N3ASIC and Design Compiler for CMOS are compared in the Table.1. Observing the results we can see that N3SIC implementation has very good overall performance when compared to the CMOS 16nm implementation.

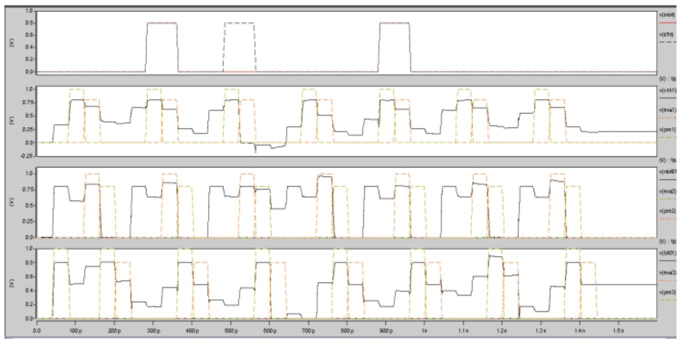

Fig.11. Simulation Results of shiftregister

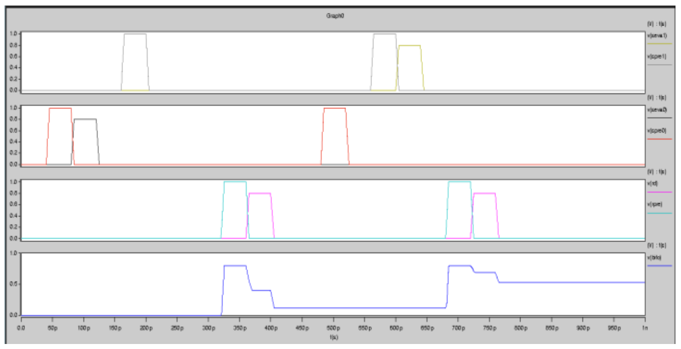

Fig.12. Simulation Results of SRAM

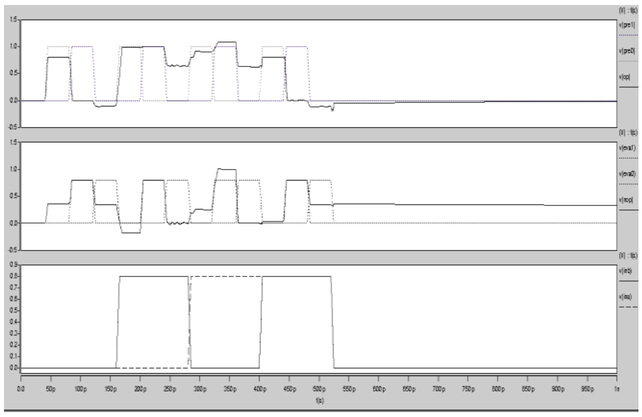

Fig.13. Simulation Results of XOR gate

TABLE I. COMPARISON BETWEEN N3ASIC AND CMOS

	N3ASIC	CMOS
AREA (Sq.µm)	2.14	3.69
Worst case Delay(pS)	13.2	22.4
Leakage Power(nW)	39.6	107.2

CONCLUSION

The nanoscale implementation of the G-Share branch predictor was done using the N3ASIC's. The simulation results are observed and compared with similar technology. The performance, area and delay characteristics of this implementation shows an improvement of about 3X in power, 1.7X in area and worst case time delay over the CMOS implementation of same feature size.

REFERENCES

[1] P. Panchapakeshan, *et. al*, "N3ASICs: Designing nanofabrics with fine-grained CMOS integration," in *NANOARCH*, 2011.

[2] M. Rahman, P. Narayanan, and C. A. Moritz, "N3asic-based nanowire volatile RAM," in *IEEE-NANO*, 2011..

[3] M. Rahman and C. A. Moritz, "Nanowire Volatile RAM as an Alternative to SRAM," *IEEE Transactions on Nanotechnology*.

[4] R. C. Bencher, H. Dai, and Y. Chen, "Gridded design rule scaling: taking the CPU toward the 16nm node," in *Proceedings of SPIE*, SanJose, CA, USA, 2009, p. 72740G-72740G-10.

[5] Y. G. S. Snider and R. S. Williams, "Nano/CMOS architectures using a field-programmable nanowire interconnect," *Nanotechnology*, vol.18, no. 3, p. 035204, Jan. 2007

[6] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.