Verilog HDL Model Based Thermometer-to-Binary Encoder with Bubble Error Correction

Zbigniew Jaworski
Institute of Microelectronics and Optoelectronics
Warsaw University of Technology
ul. Koszykowa 75, 00-662 Warszawa, Poland

***Abstract*—This paper compares several approaches to come up with the Verilog HDL model of the thermometer-to-binary encoder with bubble error correction. It has been demonstrated that implementations of different ideas to correct bubble errors yield circuits whose parameters tremendously vary in delay, area and power consumption. The shortest delay is achieved for the design synthesized from the model which mimics a human reading temperature on classic liquid-in-glass thermometer.**

***Index Terms*—thermometer-to-binary encoder, bubble error correction, FLASH ADC, HDL model, logic synthesis.**

I. Introduction

Nowadays, the use of automatic logic synthesis of Verilog or VHDL model to generate a digital circuit is the preferred design style, especially when complex algorithms are implemented [1]–[6]. The obvious advantages, among the others, are ease of targeting different technologies and simplicity to make changes to constraints in order to obtain fast or low-power circuit.

Modern commercial CAD software are powerful tools boasting 100+ million gate strength. However, the design performance resulting from HDL code synthesis is highly sensitive to modeling style. The Verilog or VHDL statements and model structures, which appear to an average engineer absolutely equivalent, are often processed by the logic synthesis tools in completely different way. As a consequence, the same CAD tool may infer a completely unalike logic implementation for the exactly the same algorithm but differently encoded.

A thermometer-to-binary encoder is a standard component of flash ADC [7]–[18]. One of the common design problem to be solved there is how to make the circuit resistant to bubble error [19]–[24]. The bubble error (*BER*) is defined as a missing '1' in input thermometer code. The BER may be a result of e.g. switching noise spreading through the comparators of the FLASH ADC. Whatever is the reason of a BER it results in erroneous output binary code.

The typical approach to encoder implementation is to use a ROM or devise a combinatorial circuit whose structure is derived directly from Boolean expression [7]–[17], [19]–[24]. However, these approaches do not exhibit the flexibility provided by automatic logic synthesis. This disadvantage is particularly burdensome in cases of higher ADC resolutions when number of comparators (and thus the number of bits in thermometer code) is large and the complexity of thermometer-to-binary encoder grows rapidly. So, in the real-life cases it is practically impossible to try out competitive implementations.

II. The Goal

The goal of this work was to investigate which thermometer-to-binary encoder implementation, encoded as a Verilog HDL model, yields fastest solution and is resistant to 2nd order bubble error (which is a cluster of two sebsequent '0'). Several experiments have been carried out to compare cost of implementation (layout area and power consumption) of different solutions.

A. Assumptions

All the implementations discussed in this paper have been developed under the following assumptions concerning the input thermometer code:

- all the bits set to '1' are correct,
- all the bits set to '0' above top '1' are correct,
- all the bits set to '0' below top '1' are incorrect.

It should be noted that the bubble error that modifies the top most '1' cannot be detected since there is no way to distinguish it from valid '0'. None of the existing solution [19]–[24] can correct this specific location of *BER*.

B. Encoder Structure

Typical solution to improve performance of a design, either in case of software or hardware, is the *divide and conquer* approach. In the case of the encoder, a simple decomposition technique has been employed [18]. Whenever it made a sense, the encoder was divided into several segments that independently process a sub-ranges of the input vector. Each segment produces a sub-code which are subsequently aggregated to produce the final output binary code. The modified encoder structure is shown in the Fig.1.

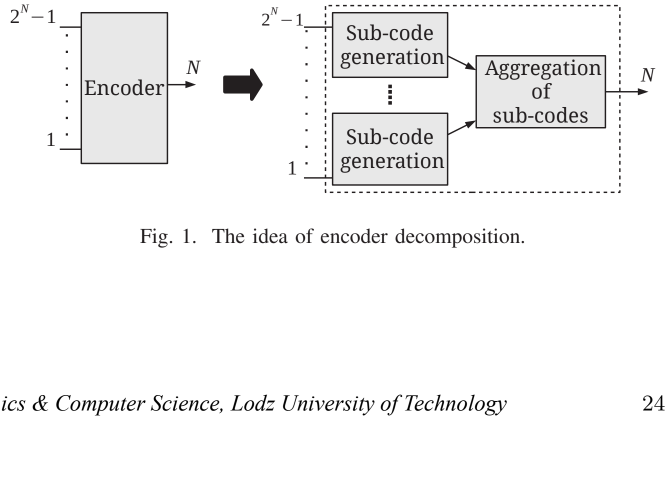

Fig. 1. The idea of encoder decomposition.

ISBN 978-83-63578-08-4

C. Bubble Error Correction

The two classic bubble error correction (BEC) approaches have been employed. The first one utilizes OR gates to form logical sum of three neighboring bits:

$$b_{corr}[i] = b[i] \lor b[i+1] \lor b[i+2] \tag{1}$$

where $n = 0 \ldots 2^n - 3$

The second method utilizes a modified *one-hot* encoder. It takes 3-input AND gate to convert thermometer code into one-hot code:

$$b_{corr}[i] = b[i] \land \neg b[i+1] \land \neg b[i+2] \tag{2}$$

where $n = 0 \ldots 2^n - 3$.
The side-effect of one-hot encoding is elimination of the bubble errors.

D. Logic Synthesis

All the proposed encoder implementations have been synthesized with a commercial CAD tool. An industrial 65 nm CMOS process has been chosen as the target technology. Since the FLASH ADCs are used for their speed the synthesis process has been carried out in such a way to determine which solution yields fastest circuits. In practice, synthesis has been performed several times for each model to find out the shortest timing constraint that yields nonzero worst slack.

Because realistic estimation of delays introduces by interconnects is crucial for design realized in nanometer technologies the synthesis tool was setup to use topological mode to calculate the parameters of nets. In this case, instead of statistical wire models, placement and routing are performed in order to obtain detail values of the parasitics R and C of the wiring.

III. Solution 1: The Highest ’1’

The first tested solution mimics the way a human is reading temperature on the classic liquid-in-glass thermometer. It logic terms it means the binary output code equals to the index of the most significant input bit which is set to ’1’.

In practice, it is achieved by scanning the complete range of input bits (*bottom-up* approach) and storing the index of each passed bit which is set to ’1’. As scanning does not stop when examined bit equals ’0’ and this solution is immune to all kinds of bubble error. The example implementation is presented in Listing I.

Since there may be an erroneous ’0’ present in the input thermometer code the sub-codes aggregation requires usage of `bin2 > 0` condition instead of `thermo[31] == 1'b1` which would be perfectly valid in case of error-free input code. An inevitable disadvantage of this solution is that the implementation of `bin2 > 0` condition requires a full comparator (6-bit in case of the example shown in Listing I). This may result in longer critical path as well as larger layout area and power consumption.

The five variants (decomposition into 1, 2, 4, 8 and 16 segments) of ”The highest ’1’” encoder have been synthesized. The results are shown in Table I. As can be observed, the fastest and smallest implementation (but most power-hungry) is obtained for the simplest solution.

LISTING I
MODEL OF ”THE HIGHEST ’1’” DECOMPOSED INTO 2 SEGMENTS.

```
module thermo2bin (thermo, bin);
input [62:0] thermo;
output [5:0] bin;

reg [5:0] bin, bin1, bin2;
integer i,j;

always @(thermo)
begin
  bin1 = 0;
  for (i=1; i<=32; i=i+1)
    if (thermo[i-1] == 1'b1) bin1 = i;
end

always @(thermo)
begin
  bin2 = 0;
  for (j=1; j<=31; j=j+1)
    if (thermo[j+31] == 1'b1) bin2 = j;
end

always @(bin1 or bin2)
if (bin2 > 0)
  bin = bin2 + 32;
else
  bin = bin1;

endmodule
```

TABLE I
RESULTS OF SYNTHESIS OF ”THE HIGHEST ’1’” ENCODER.

# of segments	delay $[ps]$	# of sdt. cells	sdt. cell area $[\mu m^2]$	power $[\mu W]$
1	290	267	1467	2406
2	355	231	1218	1685
4	460	230	1156	1479
8	415	246	1258	1586
16	297	295	1510	2666

IV. Solution 2: The Highest ’1’ with BEC

This solution is very similar to the previous one. The example Verilog model is shown in the Listing II. The additional procedural block implements the OR-based BEC.

The existence of the BEC pre-processing logic allows to use the simpler condition statement `thermo[31] == 1'b1` in the process responsible for sub-codes aggregation. However, additional process (the BEC implementation) may result in longer critical path. It is impossible to predict which effect is dominant. This may depend on parameters of standard cells from the particular target library or even algorithms used by the particular synthesis tool.

The five variants (decomposition into 1, 2, 4, 8 and 16 segments) of this encoder version have been synthesized. The results are shown in Table II.

As can be observed, this time the fastest design is obtained when encoder is split into 2 segments. This approach yields circuit with even shorter critical path than pure ”The Highest ’1’” concept. But implementation cost (layout area and total power consumption) is higher. It should be noted that this approach is resistant to 2nd order bubble error at most.

LISTING II
"THE HIGHEST '1' WITH BEC" DECOMPOSED INTO 2 SEGMENTS.

```
module thermo2bin (thermob, bin);
input [62:0] thermob;
output [5:0] bin;

reg [62:0] thermo;
reg [5:0] bin, bin1, bin2;
integer i, j, k;

always @(thermob)
begin
  for (k=0; k<=60; k=k+1)
    thermo[k] <= thermob[k] || thermob[k+1] || thermob[k+2];

  thermo[61] <= thermob[61] || thermob[62];
  thermo[62] <= thermob[62];
end

always @(thermo)
begin
  bin1 = 0;
  for (i=1; i<=32; i=i+1)
    if (thermo[i-1] == 1'b1) bin1 = i;
end

always @(thermo)
begin
  bin2 = 0;
  for (j=1; j<=31; j=j+1)
    if (thermo[j+31] == 1'b1) bin2 = j;
end

always @(bin1 or bin2)
if (thermo[31] == 1'b1)
  bin = bin2 + 32;
else
  bin = bin1;

endmodule
```

TABLE II
RESULTS OF SYNTHESIS OF "THE HIGHEST '1' WITH BEC"

# of segments	delay $[ps]$	# of sdt. cells	sdt. cell area $[\mu m^2]$	power $[\mu W]$
1	345	208	1093	1519
2	276	348	2022	4165
4	296	220	1211	2292
8	307	251	1412	2616
16	455	244	1153	1145

V. SOLUTION 3: ONE-HOT ENCODING

The one-hot encoding is a typical approach utilized in the thermometer-to-binary encoder implementations based on ROM usage. The immanent feature of this encoding is elimination of the bubble errors. The proposed solution encodes the input thermometer word into one-hot code in the first step. The obtained code contains only one '1' in the whole vector, which is also the highest '1'. Thus, the binary output can be calculated exactly in the same way as previously. The example implementation is presented in Listing III.

The five variants (decomposition into 1, 2, 4, 8 and 16 segments) of this encoder version have been synthesized. The results of are shown in Table III. As can be observed, the shortest critical path is obtained for the simplest version, without segmentation. The delay value is almost identical to the one obtained in the first solution ("the highest '1'") and slightly greater than attained for the 2nd proposition.

LISTING III
ONE-HOT ENCODING

```
module thermo2bin (thermob, bin);
input [62:0] thermob;
output [5:0] bin;

reg [62:0] thermo;
reg [5:0] bin, bin1, bin2;
integer i,j,k;

always @(thermob)
begin
  for (k=0; k<=60; k=k+1)
    thermo[k] <= thermob[k] && ~thermob[k+1] && ~thermob[k+2];

  thermo[61] <= thermob[61] && ~thermob[62];
  thermo[62] <= thermob[62];
end

always @(thermo)
begin
  bin1 = 0;
  for (i=1; i<=32; i=i+1)
    if (thermo[i-1] == 1'b1) bin1 = i;
end

always @(thermo)
begin
  bin2 = 0;
  for (j=1; j<=31; j=j+1)
    if (thermo[j+31] == 1'b1) bin2 = j;
end

always @(bin1 or bin2)
if (bin2 > 0)
  bin = bin2 + 32;
else
  bin = bin1;

endmodule
```

TABLE III
RESULTS OF SYNTHESIS OF "ONE-HOT ENCODING"

# of segments	delay $[ps]$	# of sdt. cells	sdt. cell area $[\mu m^2]$	power $[\mu W]$
1	287	295	1651	2916
2	304	284	1426	2391
4	410	257	1319	1808
8	455	274	1470	1694
16	490	281	1445	1199

VI. SOLUTION 4: SUM OF BITS WITH OR-BASED BEC

The next approach which has been examined is the sum of all the bits in the input code (a.k.a *fat tree*). Implementation of this idea should results in combinational logic built of adders. The bubble error correction has been realized by means of inserting the OR-based BEC between the input port and the classic implementation of the encoder. This additional Verilog procedural block transforms input vector into bubble-free internal code which is subsequently used to produce the binary output simply by summing up all the bits. The example implementation is presented in Listing IV.

The five variants (decomposition into 1, 2, 4, 8 and 16 segments) of this encoder version have been synthesized. The results are shown in Table IV. As can be seen, the shortest critical path is obtained for design decomposed into 16 segments. The shortest possible delay value is longer than those obtained in all the previous solutions.

LISTING IV
SUM OF BITS WITH OR

```
module thermo2bin (thermob, bin);
input   [62:0] thermob;
output  [5:0]  bin;

reg [62:0] thermo;
reg [5:0] bin, bin1, bin2;
integer i,j,k;

always @(thermob)
begin
  for (k=0; k<=60; k=k+1)
    thermo[k] <= thermob[k] || thermob[k+1] || thermob[k+2];

  thermo[61] <= thermob[61] || thermob[62];
  thermo[62] <= thermob[62];
end

always @(thermo)
begin
  bin1 = 0;
  for (i=0; i<=31; i=i+1)
     bin1 = bin1 + thermo[i];
end

always @(thermo)
begin
  bin2 = 0;
  for (j=32; j<=62; j=j+1)
    bin2 = bin2 + thermo[j];
end

always @(bin1 or bin2)
if (bin2 > 0)
  bin = bin2 + 32;
else
  bin = bin1;

endmodule
```

TABLE IV
RESULTS OF SYNTHESIS OF "SUM OF BITS WITH OR"

# of segments	delay $[ps]$	# of sdt. cells	sdt. cell area $[\mu m^2]$	power $[\mu W]$
1	1010	1104	6821	4953
2	910	1128	6624	4827
4	650	926	5578	5474
8	470	621	3668	4575
16	325	341	1674	2987

VII. SOLUTION 5: TRUTH TABLE

The last attempt to encoder realization was based on the truth table implementation (Listing V). Since only small subset of all possible combinations of input bit values are utilized there the default binary value has to be set in order to avoid D-latches. The possible options of the default output value are: `6'b000000`, `6'b111111` and `6'bzzzzzz`.

The truth table may be also implemented in the form of so called priority encoder (Listing VI). In this case the upper triangular part of the truth table is filled with 0s while the 1s appear on the diagonal. The lower triangular part is filled with *don't care* values (in Verilog represented as 'z').

In the both cases the OR-based BEC is to used to transforms input vector into bubble-free internal code which is then subsequently converted into the binary output.

LISTING V
MODEL VERSION C: DIRECT TRUTH TABLE IMPLEMENTATION

```
module thermo2bin (thermo, bin);
input  [62:0] thermo;
output [5:0] bin;

reg [5:0] bin;
integer k;
reg [62:0] thermo;

always @(thermob)
begin
  for (k=0; k<=60; k=k+1)
    thermo[k] <= thermob[k] || thermob[k+1] || thermob[k+2];

  thermo[61] <= thermob[61] || thermob[62];
  thermo[62] <= thermob[62];
end

always @(thermo)
casez (thermo)
  63'h0000000000000000 : bin =  0;
  63'h0000000000000001 : bin =  1;
  63'h0000000000000003 : bin =  2;
  ...............................
  ...............................
  ...............................
  63'h1fffffffffffffff : bin = 61;
  63'h3fffffffffffffff : bin = 62;
  63'h7fffffffffffffff : bin = 63;
  //To avoid D-latches default bin value has to be set.
  //Possible options are: 6'b000000, 6'b111111, 6'bzzzzzz
  default : bin =  6'b000000;
endcase

endmodule
```

LISTING VI
MODEL VERSION D: PRIORITY ENCODER

```
module thermo2bin (thermob, bin);
input  [62:0] thermob;
output [5:0]  bin;

integer k;
reg [62:0] thermo;
reg [5:0] bin;

always @(thermob)
begin
  for (k=0; k<=60; k=k+1)
    thermo[k] <= thermob[k] || thermob[k+1] || thermob[k+2];

  thermo[61] <= thermob[61] || thermob[62];
  thermo[62] <= thermob[62];
end

always @(thermo)
casez (thermo)
  {63'b0               }: bin =  0;
  {62'b0,          1'b1}: bin =  1;
  {61'b0, 1'b1,    1'bz}: bin =  2;
  {60'b0, 1'b1,    2'bz}: bin =  3;
  ...............................
  ...............................
  ...............................
  { 2'b0, 1'b1,   60'bz}: bin = 61;
  { 1'b0, 1'b1,   61'bz}: bin = 62;
  { 1'b1,         62'bz}: bin = 63;
endcase

endmodule
```

The three variants of truth table based design and priority encoder based implementations have been synthesized. The parameters of the obtained circuits are shown in Table V and Table VI respectively. The results shown there prove that the priority encoder allow for faster and smaller circuits.

TABLE V

Results of synthesis of truth table implementation

default case	delay $[ps]$	# of sdt. cells	sdt. cell area $[\mu m^2]$	power $[\mu W]$
bin = 6'b000000	401	703	4218	5767
bin = 6'b111111	375	704	4339	6320
bin = 6'bzzzzzz	644	509	2557	2157

TABLE VI

Results of synthesis of priority encoder

delay $[ps]$	# of sdt. cells	sdt. cell area $[\mu m^2]$	power $[\mu W]$
360	309	1417	2222

VIII. Summary of Results

The five different approaches to implement thermometer-to-binary encoder with be bubble error correction have been examined. Whenever it made a sense, a simple decomposition technique (*divide and conquer* approach) has been employed.

All the Verilog models have been synthesized with a commercial CAD tool. An industrial 65 nm CMOS process has been used as the target technology. The synthesis procedure has been carried out in such a way that allowed to obtain the fastest possible circuit for every variant.

The Table VII presents parameters of the three best designs: ”the highest '1'”, ”the highest '1' with BEC” and ”one-hot encoding”.

TABLE VII

Results of synthesis: the fastest model variants

model variant	parts	delay $[ps]$	cells	cell area $[\mu m^2]$	power $[\mu W]$
highest '1' (native)	1	290	267	1467	2406
highest '1' w/ BEC	2	276	348	2022	4165
one-hot encoding	1	287	295	1651	2916

All of these circuits were synthesized from the Verilog models which implement algorithms that mimics human reading temperature on classic liquid-in-glass thermometer yield. In all these cases the length of the critical path is relatively similar and belongs to the range of 276…290 ps. Much more significant are differences in power consumption and layout area.

The first solution has a unique feature – it is resistant to all kinds of bubble errors. It utilizes the simplest idea and does not use any special BEC circuitry. In addition, the cost of implementation is the lowest.

The two other approaches exhibit slightly shorter critical paths but need more silicon area and consume more energy. What's most important, they are resistant to 2nd order bubble error at most.

So, there is a dilemma. Should the fastest solution be chosen or maybe the one most resistant to the bubble errors? The answer depends on the priorities in a particular project.

It should also be emphasized that parameters of the synthesized circuits considerably dependent on the features of particular standard cells from a specific target library. Thus, in case of other technologies (or different technology modules e.g. high performance, low-leakage, etc.) it may turn out that shortest delay is achieved for a different model variant than reported in this paper. Analogous effects might be expected when different synthesis tool is used.

IX. Conclusions

As has been demonstrated in this paper, in real-life projects, when the number of comparators in the FLASH ADC is large, finding an optimal implementation of thermometer-to-binary encoder is not so easy.

A routine task like converting thermometer code into binary word seems to be a relatively easy to realize. However, with larger number of the input bits complexity of the resulting logic grows rapidly, both in therms of the number of used logic gates as well as the length of interconnections. The critical path, layout area and power consumption depends heavily on the actual logic structure and parasitics introduced be the wiring.

Evidently, full-custom approach to such a task is burdensome and automatic logic synthesis seems to be a sensible alternative. However, there are pitfalls here, too. Various VHDL or Verilog HDL statements and/or model structures may result in completely different parameters of the synthesized circuits. The fact an average engineer is usually not aware of.

In order to get the most from automatic logic synthesis designers have to understand how CAD tools infer logic. It should be clear that even most powerful software cannot improve poor algorithms and/or poor encoding. It is the engineer role to come up with clever ideas and specify it in such way that CAD tools can generate optimal implementation.

References

[1] M. Muzal and Z. Jaworski, “Hardware implementation of fuzzy logic based diagnostic algorithm for implantable cardioverter defibrillator,” in *Proceedings of the 11th International Conference Mixed Design of Integrated Circuits and Systems MIXDES'2004*. Wydawnictwo Politechniki dzkiej, 2004, pp. 579–584.

[2] A. Bardossy, A. Blinowska, W. Kumicz, J. Ollitrault, M. Lewandowski, A. Przybylski, and Z. Jaworski, “Fuzzy logic-based diagnostic algorithm for implantable cardioverter defibrillators,” *Artificial Intelligence in Medicine*, no. 60, pp. 113–121, 2014.

[3] A. Wojtasik, Z. Jaworski, W. Kumicz, A. Wielgus, A. Wakanis, and D. Sarna, “Fuzzy logic controller for rate-adaptive heart pacemaker,” *Applied Soft Computing*, vol. 4, pp. 259–270, 2004.

[4] M. Banaszkiewicz, S. Gadomski, A. Gaat, and Z. Jaworski, “The rosettabis asic for the rosetta space mission,” in *Proceedings of the 7th International Conference: Mixed Design of Integrated Circuits and Systems MIXDES 2000*, A. Napieralski, Ed. Technical University of d, 2000, pp. 587–590.

[5] Z. Jaworski, W. Kumicz, M. Sadowski, D. Sarna, A. Wakanis, A. Wielgus, and A. Wojtasik, “Vlsi implementations of fuzzy logic controllers for rate-adaptive pacemakers,” in *Microtechnologies in Medicine and Biology, 1st Annual International, Conference On. 2000*, 2000, pp. 475–478.

[6] A. Bardossy, A. Blinowska, Z. Jaworski, W. Kumicz, J. Ollitrault, A. Penciolelli, D. Sarna, A. Wakanis, A. Wielgus, and A. Wojtasik, “Application of fuzzy logic to pacemaker control,” *Biocybernetics and Biomedical Engineering*, vol. 1, pp. 7–23, 2003.

[7] E. Sail and M. Vesterbacka, “Comparison of two thermometer-to-binary decoders for high-performance flash adcs,” in *NORCHIP Conference, 2005. 23rd*, Nov 2005, pp. 253–256.

[8] S. Ali, R. Raut, and M. Sawan, “Digital encoders for high speed flash-adcs: Modeling and comparison,” in *Circuits and Systems, 2006 IEEE North-East Workshop on*, June 2006, pp. 69–72.

[9] E. Sail and M. Vesterbacka, “Thermometer-to-binary decoders for flash analog-to-digital converters,” in *Circuit Theory and Design, 2007. ECCTD 2007. 18th European Conference on*, Aug 2007, pp. 240–243.

[10] G. Madhumati, K. Rao, and M. Madhavilatha, “Comparison of 5-bit thermometer-to-binary decoders in 1.8v, 0.18 um cmos technology for flash adcs,” in *2009 International Conference on Signal Processing Systems*, May 2009, pp. 516–520.

[11] P. Kanodiya and A. Naik, “Analysis and design of flash analog to digital converter for ultra wide band application,” in *Engineering (NUiCONE), 2011 Nirma University International Conference on*, Dec 2011, pp. 1–5.

[12] A. Kale, P. Palsodkar, and P. Dakhole, “Comparative analysis of 6 bit thermometer-to-binary decoders for flash analog-to-digital converter,” in *Communication Systems and Network Technologies (CSNT), 2012 International Conference on*, May 2012, pp. 543–546.

[13] X. Hu, L. Zhao, S. Liu, J. Wang, and Q. An, “A stepped-up tree encoder for the 10-ps wave union tdc,” in *Real Time Conference (RT), 2012 18th IEEE-NPSS*, June 2012, pp. 1–5.

[14] P. Palsodkar, P. Dakhole, P. Palsodkar, and A. Kale, “Highly digitalized flash analog to digital (fadc) converter using mux based decoder topology,” in *Emerging Technology Trends in Electronics, Communication and Networking (ET2ECN), 2012 1st International Conference on*, Dec 2012, pp. 1–6.

[15] A. Chunn and R. Sarin, “Comparison of thermometer to binary encoders for flash adcs,” in *India Conference (INDICON), 2013 Annual IEEE*, Dec 2013, pp. 1–4.

[16] M. Pasha and M. Vesterbacka, “A modified switching scheme for multiplexer based thermometer-to-binary encoders,” in *NORCHIP, 2014*, Oct 2014, pp. 1–4.

[17] C. Vudadha, G. Makkena, M. Nayudu, P. Phaneendra, S. Ahmed, S. Veeramachaneni, N. Muthukrishnan, and M. Srinivas, “Low-power self reconfigurable multiplexer based decoder for adaptive resolution flash adcs,” in *VLSI Design (VLSID), 2012 25th International Conference on*, Jan 2012, pp. 280–285.

[18] Z. Jaworski, “Choosing the optimal hdl model of thermometer-to-binary encoder,” in *Mixed Design of Integrated Circuits Systems (MIXDES), 2015 22nd International Conference*, June 2015, pp. 297–300.

[19] Y.-J. Chuang, H.-H. Ou, and B.-D. Liu, “A novel bubble tolerant thermometer-to-binary encoder for flash a/d converter,” in *VLSI Design, Automation and Test, 2005. (VLSI-TSA-DAT). 2005 IEEE VLSI-TSA International Symposium on*, April 2005, pp. 315–318.

[20] N. Agrawal and R. Paily, “An improved rom architecture for bubble error suppression in high speed flash adcs,” in *Student Paper, 2008 Annual IEEE Conference*, Feb 2008, pp. 1–5.

[21] M. Rahman, K. Baishnab, and F. Talukdar, “A novel rom architecture for reducing bubble and metastability errors in high speed flash adcs,” in *Electronics, Communications and Computer (CONIELECOMP), 2010 20th International Conference on*, Feb 2010, pp. 15–19.

[22] B. V. Hieu, S. Beak, S. Choi, J. Seon, and T. Jeong, “Thermometer-to-binary encoder with bubble error correction (bec) circuit for flash analog-to-digital converter (fadc),” in *Communications and Electronics (ICCE), 2010 Third International Conference on*, Aug 2010, pp. 102–106.

[23] B. V. Hieu, S. Choi, J. Seon, Y. Oh, C. Park, J. Park, H. Kim, and T. Jeong, “A new approach to thermometer-to-binary encoder of flash adcs- bubble error detection circuit,” in *Circuits and Systems (MWSCAS), 2011 IEEE 54th International Midwest Symposium on*, Aug 2011, pp. 1–4.

[24] S. Kumar, M. Suman, and K. Baishnab, “A novel approach to thermometer-to-binary encoder of flash adcs-bubble error correction circuit,” in *Devices, Circuits and Systems (ICDCS), 2014 2nd International Conference on*, March 2014, pp. 1–6.