INTERNATIONAL JOURNAL OF CIRCUITS, SYSTEMS AND SIGNAL PROCESSING
DOI: 10.46300/9106.2021.15.180 Volume 15, 2021

A solution to design semi-static Null Convention Logic cell libraries

Lac Truong Tri[1,2], Toi Le Thanh[1,2,3], Trang Hoang[1,2]
[1]Ho Chi Minh City University of Technology (HCMUT), 268 Ly Thuong Kiet Street, District 10, Ho Chi Minh City, Vietnam
[2]Vietnam National University Ho Chi Minh City, Linh Trung Ward, Thu Duc District, Ho Chi Minh City, Vietnam
[3]Ho Chi Minh City University of Food Industry (HUFI), 140 Le Trong Tan Street, Tay Thanh Ward, Tan Phu District, Ho Chi Minh City, Vietnam
Correspondence: hoangtrang@hcmut.edu.vn

Received: April 29, 2021. Received: October 16, 2021. Accepted: November 5, 2021. Published: November 18, 2021.

Abstract—The Null Convention Logic (NCL) based asynchronous circuits have eliminated the disadvantages of the synchronous circuits, including noise, glitches, clock skew, power, and electromagnetic interference. However, using NCL based asynchronous designs was not easy for students and researchers because of the lack of standard NCL cell libraries. This paper proposes a solution to design a semi-static NCL cell library used to synthesize NCL based asynchronous designs. This solution will help researchers save time and effort to approach a new method. In this work, NCL cells are designed based on the Process Design Kit 45nm technology. They are simulated at the different corners with the Ocean script and Electronic Design Automation (EDA) environment to extract the timing models and the power models. These models are used to generate a *.lib file, which is converted to a *.db file by the Design Compiler tool to form a complete library of 27 cells. In addition, we synthesize the NCL based full adders to illustrate the success of the proposed library and compare our synthesis results with the results of the other authors. The comparison results indicate that power and delay are improved significantly.

Keywords—NCL Cell Library, Threshold Gate, Asynchronous method, Null Convention Logic.

I. Introduction

THE synchronous circuits are pretty popular, and they often use a clock signal to control their operations [1]. These circuits have played a significant role and have dominated the semiconductor industry [2]. This industry has continuously reduced the feature size of transistors and wires for decades. As a result, a high-density number of transistors was integrated into a single chip [3]. However, the semiconductor industry must face clock pulse related issues such as noise, power consumption, clock skew, electromagnetic interference, and the complexity of the clock network layout. These issues are considered as the technological challenges that the semiconductor industry must encounter in the future [4].

In contrast to the synchronous circuit models, the asynchronous circuit models do not employ a clock signal because they communicate through a local handshake protocol to eliminate the clock-related issues as presented above [5]. Among the asynchronous circuit models, NCL is a Quasi-Delay-Insensitive (QDI) logic model chosen to design asynchronous circuits and realize commercial applications because of its delay-insensitive advantages [6]. In recent years, researches based on NCL have shown outstanding performance compared with studies based on the synchronous method [5], [7]-[10]. Additionally, researchers study NCL for various purposes such as synthesis of QDI combinational circuits using NCL based on elemental Gates [1], comparison of NCL threshold gate models [5], Efficient Muller C-element implementation for asynchronous designs [11], a solution to replace bus for asynchronous designs [12], Design and detection hardware trojans in asynchronous circuits [13], low power circuits [14]-[17], and some relevant studies can be found in [18]-[21]. In most of NCL based researches mentioned above, authors synthesized their designs in three approaches. The first approach is to use a full-custom design flow, which is not feasible for some complex structures. The second is to synthesize designs based on conventional synchronous libraries [16]. Design based on such a library makes the NCL based asynchronous design not achieve optimal performance. The last approach is to use mapping tools to convert from synchronous to asynchronous designs [22]. However, these tools are difficult to optimize for large-scale structures. In addition, the semi-custom design of the QDI circuits always uses threshold gates that are not available in the commercial standard cell library.

This lack of the NCL cell library would be a significant barrier to the research and development of the NCL based asynchronous design method.

In the state-of-the-art researches of NCL cell library designs, there are several suggested flows to design NCL based asynchronous libraries [23]-[24]. These flows are pretty complex and use some own tools of the authors. These tools could cause difficulties for users to install or use if any errors appear. Therefore, in this paper, we propose a simple flow to design the NCL cell library with threshold gates using only the main commercial tools and use the semi-static structure as an example to analyze and experiment. This flow helps researchers by themselves to generate new NCL cell libraries and update new cells easily.

The remains of this article are organized as follows: Section II presents an overview of NCL, the standard NCL cell library design flow, and the cell characterization. Results and discussions are given in Section III. Finally, Section IV provides conclusions of the proposed solution to design the NCL cell library.

II. Materials and Methods

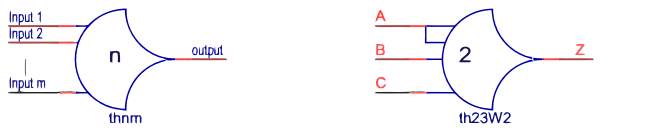
A. Null Convention Logic

NCL is not only a QDI logic model but also a symbolically complete logic model. NCL is a new logic design model that does not use a clock signal and intends for asynchronous circuits [25]. The NCL-based asynchronous circuits execute correctly regardless of component and wire delays and use dual-rail logic [9]. Therefore, an NCL logic signal is formed by two rails. Table I shows the conversion of a conventional logic signal to a dual-rail signal [7]. Both A1 and A0 rails cannot be in the '1' state simultaneously because these two rails are mutually exclusive.

Table. I Dual rail signal

	Code	
Dual-rail logic	A1	A0
DATA0	0	1
DATA1	1	0
NULL	0	0
ILLEGAL	1	1

NCL-based circuits use a set of threshold gates, including 27 threshold gates shown in Table II [7], [9]. Fig. 1(a) depicts the general symbol of the thnm threshold gate, where n is the threshold value that means at least n of m inputs transition to '1' state before the output transitions to '1' state, and m is the total number of inputs. A different kind of threshold gate denoted as thnmWz1z2…zm is a weighted threshold gate, where the input weights are z1, z2, …, and zm. An example of the weighted threshold gate, th23W2, is shown in Fig. 1(b).



(a) Thnm threshold gate (b) Th23W2 threshold gate

Fig. 1 The threshold gate

Table. II 27 threshold gates

No	NCL gates	Boolean Functions
1	th12	X + Y
2	th22	XY
3	th13	X + Y + Z
4	th23	XY + YZ + ZX
5	th33	XYZ
6	th23w2	X + YZ
7	th33w2	XY + XZ
8	th14	X + Y + Z + W
9	th24	XY + XZ + XW + YZ + YW + ZW
10	th34	XYZ + XYW + XZW + YZW
11	th44	XYZW
12	th24w2	X + YZ + YW + ZW
13	th34w2	XY + XZ + XW + YZW
14	th44w2	XYZ + XYW + XZW
15	th34w3	X + YZW
16	th44w3	XY + XZ + XW
17	th24w22	X + Y + ZW
18	th34w22	XY + XZ + XW + YZ + YW
19	th44w22	XY + XZW + YZW
20	th54w22	XYZ + XYW
21	th34w32	X + YZ + YW
22	th54w32	XY + XZW
23	th44w322	XY + XZ + XW + YZ
24	th54w322	XY + XZ + YZW
25	thxor0	XY + ZW
26	thand0	XY + YZ + XW
27	th24comp	XZ + YZ + XW + YW

As presented above, NCL-based asynchronous circuits are formed by threshold gates. The typical paradigm of a static CMOS threshold gate with latency comprises five function blocks (reset, set, hold null, hold data, and an inverter), as depicted in Fig. 2. In this structure, the reset block complements the hold data block [6], and Fig. 3 shows their general

structures. According to these structures, threshold gates with the same number of inputs have the same reset block and hold data block. Similarly, the set block complements the hold null block, but their actual structures depend on the number of inputs and the threshold value.

In Fig. 3, the reset block is active when all inputs are in the '0' state. In contrast, the hold data block is active when at least one or more inputs are in the '1' state. Note that the reset block and the hold data block are always in their standard forms.

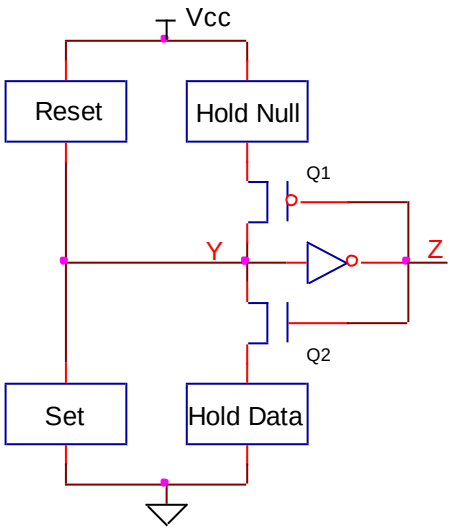


Fig. 2 Common paradigm of a static CMOS threshold gate

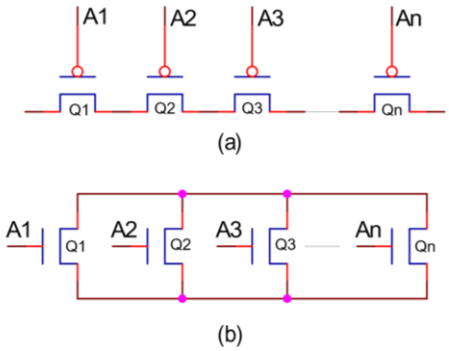


Fig. 3 General structure of Reset block (a) and Hold data block (b)

Similar to the typical paradigm of the static threshold gate, Fig. 4 shows a standard paradigm of the semi-static threshold gate [25]. This gate includes three function blocks (a reset block, a set block, and a weak feedback inverter at the output). When both set and reset networks are off, the logic level on node Y is kept unchanged at this inverter output. In addition, noise on node Y will influence the weak inverter if it is too small.

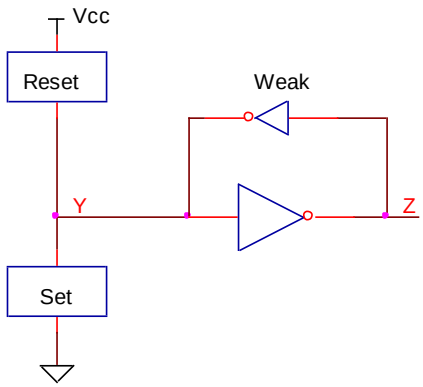


Fig. 4 Common model of a semi-static CMOS threshold gate

In many applications related to real-time computing, such as signal processing, the input data flow is continuous at the minimum speed. In these cases, a feedback mechanism is not essential to maintain the state information. Therefore, we can remove the weak feedback inverter from the semi-static structures. As a result, a new paradigm is formed and is called a dynamic threshold gate. Fig. 5 depicts the general structure of the dynamic threshold gates [25].

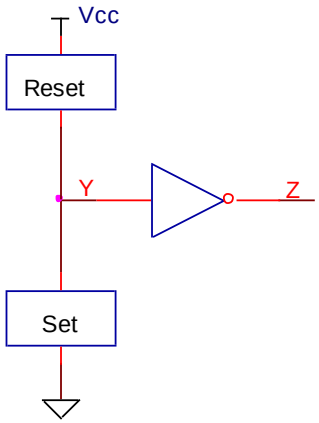


Fig. 5 The common paradigm of a dynamic CMOS threshold gate

B. The proposed flow to design a standard NCL Cell Library

This subsection presents a simple flow to design the NCL cell library with threshold gates using only the main commercial tools. The proposed flow depicted in Fig. 6 comprises eight steps from the schematic design step to the RTL synthesis test step. In this work, the semi-static NCL cell library is implemented by our flow because the semi-static NCL cell library is one of the most fundamental libraries for university students and researchers. This library that comprises 27 cells are threshold gates.

Firstly, to design a cell circuit schema, we begin with the general paradigm of the semi-static NCL cells, as depicted in Fig. 4. For example, designing the th23 threshold gate, in this cell, the reset block is in its standard form shown in Fig. 3. Their circuit diagram depends on the number of cell inputs. As the th23 gate has three inputs, the circuit diagram of the reset block will comprise three PMOS transistors connected in series. Subsequently, we design the circuit diagram of the set block.

This block is only active when any two of the three inputs (i.e. (A and B) or (A and C) or (B and C)) go to the high level. For this condition, its switching expression also describes the cell function, as presented in Table II and explicitly mentioned by (1). To reduce the number of NMOS transistors used to generate the set block, we convert (1) to (2). Finally, we design the weak feedback inverter. Because of the weak inverter, the transistor size in this inverter must be smaller than that of the other transistors. Therefore, we keep the pair of transistors that make up this inverter at the standard size and change the width of the remaining transistors. This phase has to be closely coordinated with the simulation phase to test the cell operation. Fig. 7 is the complete circuit of th23, and its symbol is in Fig. 8.

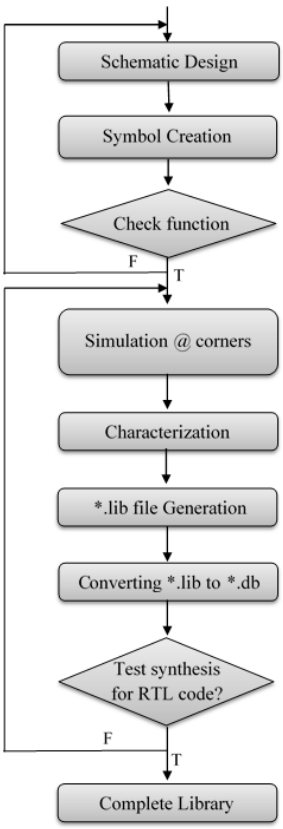


Fig. 6 Standard NCL Cell Library Design Flow chart

$$F_{SET} = AB + AC + BC \quad (1)$$

$$F_{SET}(A, B, C) = A(B + C) + BC \quad (2)$$

After designing the schematic circuit of the cell, we simulate it to check its function. If the simulation results of the cell are correct, this process will move to the simulation step at the corners. Otherwise, we have to recheck the schematic circuit design step.

Next step, we implement the cell simulation at corners to measure the leakage power and the input capacitances. Subsequently, we carry out the cell characterization to extract the power models and the timing models to create a *.lib file (this step will be covered in detail in subsection 2.3). This file complies with Synopsys standards. We use the Library Compiler tool of Synopsys to convert a *.lib file to a *.db file which is one of the vital files in the libraries. It contains the essential parameters of threshold gates.

Finally, the design synthesis step is implemented to check if the cell library works appropriately. This step will write a piece of RTL code and synthesize it at the gate level. If the synthesis results are sound, we will complete the NCL cell library.

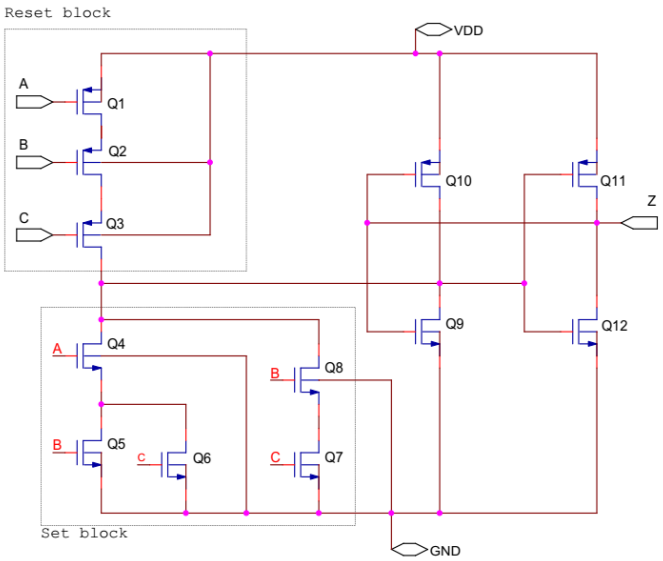


Fig. 7 The schematic circuit of th23

C. NCL cell characterization

Cell characterization is one of the most important steps in our flow because, in this step, cell models are generated to form the library. We perform characterization for all cells in Table II, and the quantities of cell rise delay, cell fall delay, rise transition, fall transition, rise power, fall power, leakage power, and input capacitance. Twenty-seven cells in the library will employ the same load capacitance C_{Load} (1.4 fF, 2.54 fF, 4.61 fF, 8.37 fF, 15.2 fF, 27.6 fF, 50.0 fF), the same fall time, and the same rise time of the input voltage waveform (0.01 ns, 0.0192 ns, 0.0368 ns, 0.0707 ns, 0.136 ns, 0.261 ns, 0.5 ns). To simulate all the cases, we must implement the tasks manually because there are no options and powerful instructions in the graphic user interface to implement repetitive tasks, which is one of the significant disadvantages of the ADE. Additionally, there is no method to characterize an ordinary cell. Therefore, in this section, Ocean language is used to support automatically executing simulations within Cadence because it is one of the powerful script languages. In addition to the Ocean script, the calculator of Virtuoso to perform cell characterization is also used. The above-mentioned parameters such as load capacitance, fall time, and rise time of the input voltage must be determined clearly in the *.ocn file to execute the simulation of the 49 times and compute the dynamic power and the timing paradigm, including the rise transition, the fall transition, the cell fall delay, and the cell rise delay. We do not use the Ocean script to assist in measuring the input capacitance and the leakage power because it is only used to measure a range of values. The testbench circuit is used to characterize the th23 cell depicted in Fig. 8.

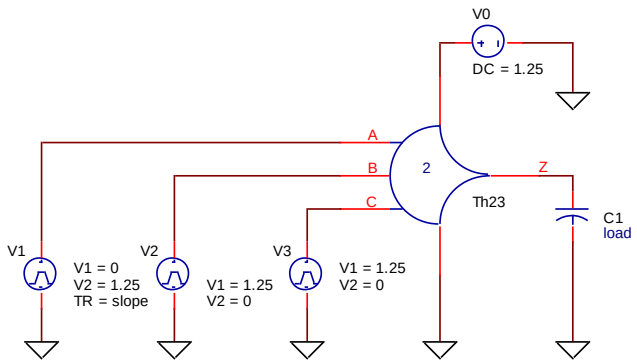


Fig. 8 The testbench circuit

Pin capacitance can be determined at all inputs and outputs. In most cases, it is only specified at input pins. That means that the output capacitance of a cell is equal to zero [26]. The input capacitance value is calculated according to (3), which shows the relation between voltage, current, and capacitance.

$$I = C\frac{dV}{dt} \tag{3}$$

By providing pulse voltage to the input pin and taking the current measurements at the same point, we will calculate the input capacitance value (4).

$$C_{input} = \frac{\int_t^{t+\Delta t} Idt}{\int_t^{t+\Delta t} dV} \tag{4}$$

Where I is the current at an input pin, and is generated by charging and discharging the charge via the input capacitance.

The cell timing paradigms are used to give precise timing for different cell scenarios in the design environment. Non-linear delay models are used to create a *.lib file because these models are accurate even if used for the sub-micrometer technology [26]. The timing paradigms are calculated for every timing arc of the cell. The timing and delays are table paradigms, which have to define detail for all the cells in the library. The transition time at the output pin and the latency via the cell for different combinations of the input transition time at the cell input and total output capacitance at the cell output pin are recorded by the table paradigms [26]. Fig. 9, Fig. 10, and Fig. 11 depict determining time values of the timing paradigms (transition time and latency). Percentages (30%, 70%, 10%, 90%) are threshold values determined clearly at the top of the liberty file [27].

Cell rise delay is the period of time from when the input reaches 70% of VDD at the first falling edge until the output reaches 70% of VDD at the first rising edge in case the timing arc of the cell being considered is negative unate. When the timing arc of the cell being considered is positive unate, cell rise delay is the period of time from when the input reaches 30% of VDD at the first rising edge until the output reaches 70% of VDD at the first rising edge.

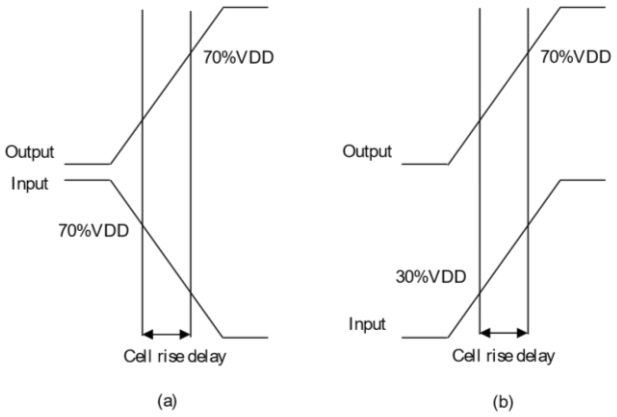


Fig. 9 Cell rise delay
a. Timing arc is negative unate
b. Timing arc is positive unate

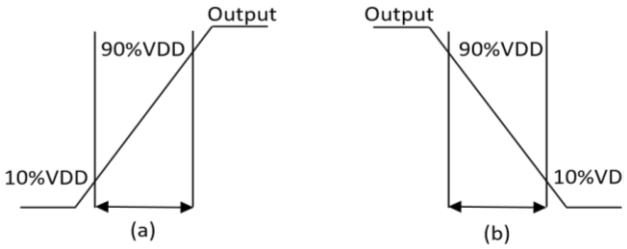


Fig. 10 Transition time at output pin
a. Rise transition,
b. Fall transition

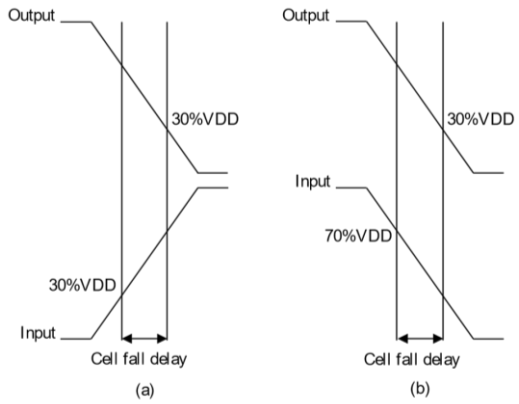


Fig. 11 Cell fall delay
a. Timing arc is negative unate
b. Timing arc is positive unate

Cell fall delay is the period of time from when the input reaches 30% of VDD at the first rising edge until the output reaches 30% of VDD at the first falling edge in case the timing arc of the cell being considered is negative unate. When the timing arc of the cell is positive unate, cell fall delay is the period of time from when the input reaches 70% of VDD at the first falling edge until the output reaches 30% of VDD at the first falling edge.

Rise transition is the period of time that the output is between 10% VDD and 90% VDD at the rising edge. Similarly, fall transition is the period of time that the output is between 90% VDD and 10% VDD at the falling edge of the output.

Dynamic power comprises fall power and rise power. Fall power is calculated in case the output changes from high to low. Similarly, rise power is calculated in case the output changes from low to high. The dynamic power is calculated according to (5).

$$P_{dynamic} = \frac{Vdd}{T} * \int_{t}^{t+\Delta t} i("/V0/PLUS")dt \quad (5)$$

Most normal cells only dissipate power when the output changes. However, other powers are consumed as the cells are supplied with the voltage but are not active because the leakage current is not zero. The tunneling current through the gate oxide of metal-oxide-silicon devices or the sub-threshold current causes the leakage [26]. The leakage power is calculated by (6).

$$P_{Leakage} = \sum I_{Leakage} V_{DD} \quad (6)$$

To calculate the leakage power, we first list all input combinations of that cell and then calculate the leakage power of every case by connecting the voltage supply line to the ground when inputs are low (pulled-down) or connecting the voltage supply line to VDD when inputs are high (pulled-up). The leakage power of a standard cell is equal to the average of all cases.

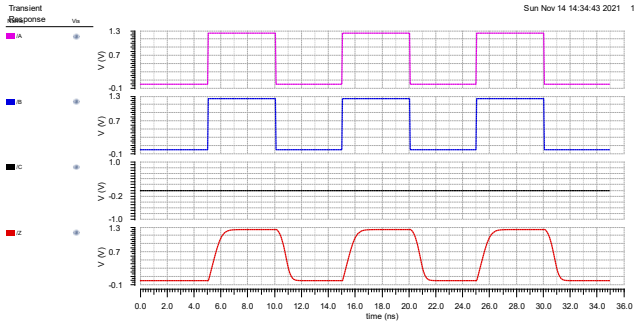
III. Results and Discussions

In this section, we present and discuss our implemented results. These results are from the processes mentioned in Section II, including checking the cell function, performing cell characterization, and checking the feasibility of the library created in this work by using this library to synthesize the RTL code.

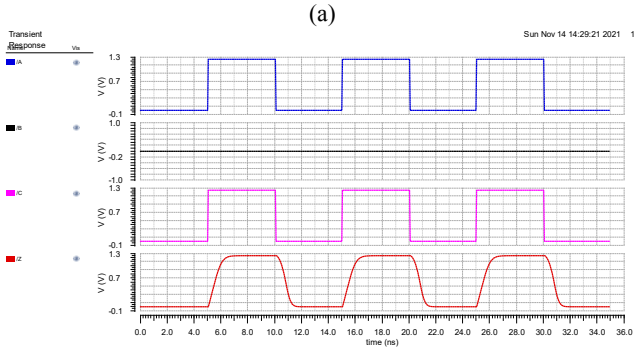
A. Function Verification

It is essential to check cell functionality before performing cell characterization. The simulation results to test its function are shown in Fig. 12.

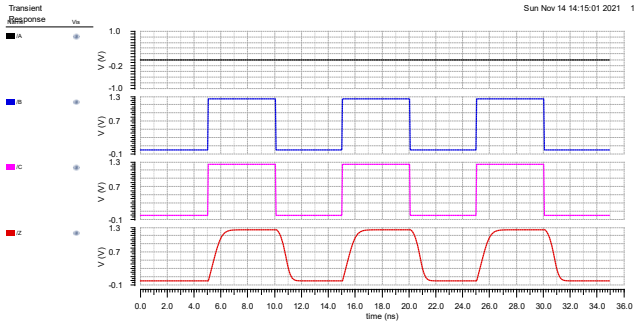
Theoretically, when two of the three inputs transition to high, the th23 gate output will become high. When all three inputs transition to low, the output will become low. Fig. 12 indicates that the circuit works correctly.



(a)



(b)



(c)

Fig. 12 Function test results of th23

B. The Simulation Results of The Cell Characterization

To perform cell characterization, we use an Ocean script to help measure 49 cases, as mentioned in subsection 2.3. Fig. 13 is the simulation result of those 49 cases (with Pin A supplied Vpulse, pin B, and pin C connected GND). Similarly, Fig. 14 and Fig. 15 show the simulation results for the remaining cases. The tables (from Table III to Table XI) show the parameters of cell fall, cell rise, rise transition, fall transition, rise power, and fall power.

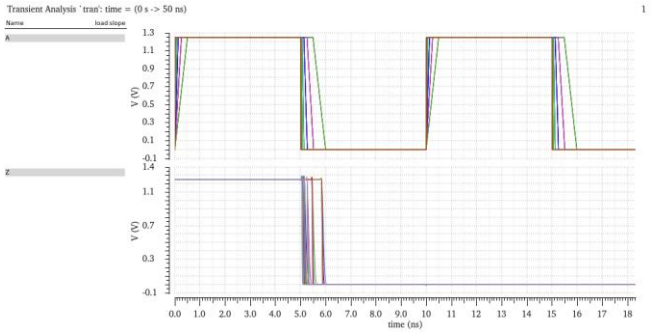


Fig.13 The simulation result with Pin A supplied Vpulse, pin B and pin C connected GND

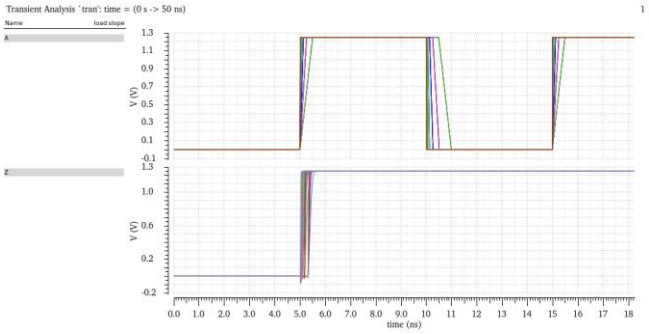


Fig.14 The simulation result with Pin A supplied Vpulse, pin B connected GND and pin C connected VDD

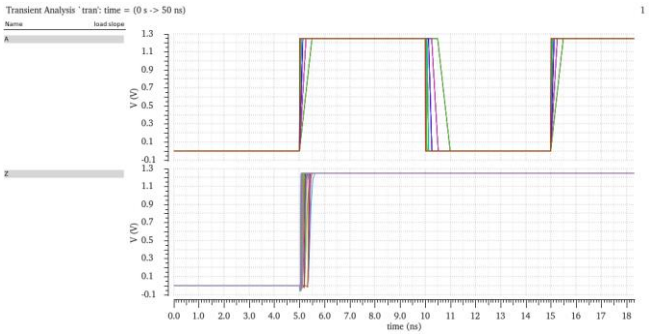


Fig.15 The simulation result with Pin A supplied Vpulse, pin B connected VDD and pin C connected GND

Table. III Cell fall delay (A = Vpulse, B = 0, C = 0)

C(fF) / T(ns)	1.4	2.54	4.61	8.37	15.2	27.6	50.0
0.0100	0.047721	0.051123	0.056087	0.064640	0.078109	0.100256	0.139117
0.0192	0.051265	0.054632	0.059808	0.068274	0.081698	0.104053	0.142892
0.0368	0.058168	0.061524	0.067025	0.075287	0.088713	0.111123	0.149553
0.0707	0.071483	0.074836	0.080299	0.088626	0.102009	0.124362	0.163031
0.1360	0.096827	0.100234	0.105587	0.113952	0.127595	0.150270	0.188948
0.2610	0.143624	0.147365	0.153165	0.162265	0.176282	0.199035	0.238323
0.5000	0.230087	0.234278	0.240559	0.250491	0.264524	0.289438	0.329877

Table. IV Fall transition (A = Vpulse, B = 0, C = 0)

C(fF) / T(ns)	1.4	2.54	4.61	8.37	15.2	27.6	50.0
0.0100	0.013900	0.016811	0.021346	0.029757	0.043895	0.069059	0.114756
0.0192	0.014002	0.016879	0.021740	0.030000	0.044036	0.068971	0.115041
0.0368	0.013800	0.016684	0.021933	0.029988	0.043836	0.069094	0.114003
0.0707	0.013970	0.016822	0.021704	0.029697	0.043950	0.069113	0.114992
0.1360	0.014565	0.017425	0.022306	0.030290	0.044187	0.069216	0.115021
0.2610	0.015798	0.018737	0.023540	0.032378	0.046032	0.070812	0.115854
0.5000	0.017632	0.021103	0.026286	0.034507	0.049036	0.074417	0.118143

Table. V Fall power (A = Vpulse, B = 0, C = 0)

C(fF) / T(ns)	1.4	2.54	4.61	8.37	15.2	27.6	50.0
0.0100	-0.007374	-0.008214	-0.007862	-0.008769	-0.009346	-0.010344	-0.011857
0.0192	-0.008123	-0.008355	-0.008085	-0.008907	-0.009568	-0.010386	-0.011742
0.0368	-0.008099	-0.007531	-0.008130	-0.008375	-0.009375	-0.010168	-0.011902
0.0707	-0.007844	-0.008153	-0.008658	-0.008828	-0.009598	-0.010329	-0.012073
0.1360	-0.008629	-0.008523	-0.008943	-0.009358	-0.009934	-0.010934	-0.012474
0.2610	-0.009517	-0.009710	-0.010004	-0.010408	-0.011113	-0.012049	-0.013656
0.5000	-0.011564	-0.011762	-0.012072	-0.012580	-0.013115	-0.014272	-0.015982

Table. VI Cell rise delay (A = Vpulse, B = 1, C = 0)

C(fF) / T(ns)	1.4	2.54	4.61	8.37	15.2	27.6	50.0
0.0100	0.027142	0.030337	0.035469	0.044172	0.059193	0.085993	0.134614
0.0192	0.031166	0.034304	0.039538	0.048181	0.063247	0.090165	0.138635
0.0368	0.039026	0.042153	0.047300	0.055988	0.070931	0.097577	0.146058
0.0707	0.054174	0.057322	0.062532	0.071182	0.086213	0.113037	0.161667
0.1360	0.080656	0.084025	0.089421	0.098094	0.113101	0.140310	0.188495
0.2610	0.125880	0.129741	0.135662	0.144906	0.160427	0.187132	0.235360
0.5000	0.204777	0.209490	0.216373	0.226670	0.242921	0.270683	0.319712

Table. VII Rise transition (A = Vpulse, B = 1, C = 0)

C(fF) / T(ns)	1.4	2.54	4.61	8.37	15.2	27.6	50.0
0.0100	0.012276	0.015657	0.021374	0.031827	0.050972	0.086437	0.151521
0.0192	0.012348	0.015576	0.021446	0.031795	0.051005	0.086529	0.150661
0.0368	0.012369	0.015603	0.021332	0.031763	0.050719	0.085898	0.151283
0.0707	0.012764	0.016033	0.021790	0.031999	0.051046	0.086372	0.151468
0.1360	0.014621	0.017837	0.023186	0.033260	0.051467	0.086964	0.149707
0.2610	0.017467	0.020945	0.026307	0.035853	0.053400	0.088585	0.150784
0.5000	0.021595	0.025247	0.031147	0.040324	0.057607	0.090918	0.152751

Table. VIII Rise power (A = Vpulse, B = 1, C = 0)

C(fF) / T(ns)	1.4	2.54	4.61	8.37	15.2	27.6	50.0
0.0100	-0.004976	-0.006487	-0.009027	-0.014407	-0.023360	-0.039414	-0.068467
0.0192	-0.005459	-0.005967	-0.009023	-0.014235	-0.023256	-0.039376	-0.068439
0.0368	-0.005345	-0.006027	-0.009412	-0.014567	-0.023474	-0.039480	-0.068651
0.0707	-0.005090	-0.006576	-0.009404	-0.014566	-0.023591	-0.039698	-0.068782
0.1360	-0.005756	-0.007288	-0.010099	-0.014959	-0.024064	-0.040292	-0.069327
0.2610	-0.006570	-0.008152	-0.010935	-0.015954	-0.024956	-0.041194	-0.070301
0.5000	-0.008108	-0.009709	-0.012552	-0.017596	-0.026657	-0.042895	-0.072109

Table. IX Cell rise delay (A = Vpulse, B = 0, C = 1)

C(fF) / T(ns)	1.4	2.54	4.61	8.37	15.2	27.6	50.0
0.0100	0.022135	0.024993	0.029759	0.038170	0.053040	0.079869	0.127608
0.0192	0.026270	0.029058	0.033892	0.042262	0.057137	0.083943	0.131823
0.0368	0.034121	0.036963	0.041762	0.049957	0.064770	0.091618	0.140064
0.0707	0.048448	0.051240	0.055948	0.064297	0.079343	0.106043	0.154212
0.1360	0.072991	0.076111	0.081000	0.089391	0.104396	0.131241	0.179645
0.2610	0.115896	0.119205	0.124750	0.132874	0.148199	0.175149	0.223204
0.5000	0.191446	0.195533	0.201511	0.211033	0.226793	0.253453	0.303216

Table. X Rise transition (A = Vpulse, B = 0, C = 1)

C(fF) / T(ns)	1.4	2.54	4.61	8.37	15.2	27.6	50.0
0.0100	0.010412	0.013628	0.019376	0.030104	0.049639	0.085393	0.150592
0.0192	0.010471	0.013628	0.019419	0.030120	0.049639	0.085234	0.150678
0.0368	0.010479	0.013749	0.019549	0.029790	0.049318	0.085410	0.149227
0.0707	0.011212	0.014305	0.020355	0.030554	0.049924	0.085234	0.150859
0.1360	0.012752	0.015928	0.021441	0.031317	0.050740	0.086075	0.151126
0.2610	0.015283	0.018250	0.023544	0.034054	0.052583	0.086882	0.151611
0.5000	0.018656	0.021799	0.027380	0.037474	0.054876	0.090628	0.151523

Table. XI Rise power (A = Vpulse, B = 0, C = 1)

C(fF) / T(ns)	1.4	2.54	4.61	8.37	15.2	27.6	50.0
0.0100	-0.003901	-0.005776	-0.008394	-0.013355	-0.022164	-0.038665	-0.067837
0.0192	-0.004229	-0.006071	-0.008479	-0.013393	-0.022343	-0.038555	-0.067655
0.0368	-0.004137	-0.005929	-0.008705	-0.013330	-0.022550	-0.038506	-0.067769
0.0707	-0.004376	-0.005813	-0.009005	-0.013704	-0.022853	-0.038816	-0.067950
0.1360	-0.004942	-0.006510	-0.009253	-0.014193	-0.023277	-0.039134	-0.068491
0.2610	-0.005748	-0.007324	-0.010054	-0.015008	-0.023987	-0.040125	-0.069336
0.5000	-0.007284	-0.008821	-0.011605	-0.016593	-0.025620	-0.041694	-0.070962

C. The Synthesis Results of The RTL Code

In this section, we use the full adder model [17] as an example for testing the library. This model includes two gates th23 and two gates th34w2, and we synthesize it using the Design Compiler tool and the library generated by our proposed flow. The typical parameters of the library are temperature (25°C), voltage (1.25 V), process (ff).

```
////////////////////////////////////////////////////////////
// Created by: Synopsys DC Ultra(TM) in wire load mode
// Version   : L-2016.03-SP1
// Date      : Mon Jul 26 05:24:33 2021
////////////////////////////////////////////////////////////


module test ( a0, a1, b0, b1, c0, c1, s0, s1, cout0, cout1 );
  input a0, a1, b0, b1, c0, c1;
  output s0, s1, cout0, cout1;


  NCL2W111X1 U5 ( .A(a1), .B(c1), .C(b1), .Y(cout1) );
  NCL3W2111X1 U6 ( .A(cout1), .D(c0), .C(b0), .B(a0), .Y(s0) );
  NCL2W111X1 U7 ( .A(c0), .B(b0), .C(a0), .Y(cout0) );
  NCL3W2111X1 U8 ( .A(cout0), .D(b0), .C(a1), .B(c1), .Y(s1) );
endmodule
```

Fig. 16. The netlist file after synthesis

```
Number of ports:                           10
Number of nets:                            10
Number of cells:                            4
Number of combinational cells:              4
Number of sequential cells:                 0
Number of macros/black boxes:               0
Number of buf/inv:                          0
Number of references:                       2

Combinational area:                54.000000
Buf/Inv area:                       0.000000
Noncombinational area:              0.000000
Macro/Black Box area:               0.000000
Net Interconnect area:      undefined  (No wire load specified

Total cell area:                   54.000000
Total area:                 undefined
```

Fig. 17 The area report result

```
Global Operating Voltage = 1.25
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000pf
    Time Units = 1ns
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1pW


  Cell Internal Power  =   2.8885 uW   (89%)
  Net Switching Power  = 351.5618 nW   (11%)
                         ---------
Total Dynamic Power    =   3.2401 uW  (100%)

Cell Leakage Power     =   2.7550 nW
```

Fig. 18 The power report result

```
Timing Path Group (none)
-----------------------------------
Levels of Logic:                2.00
Critical Path Length:           0.12
Critical Path Slack:          uninit
Critical Path Clk Period:        n/a
Total Negative Slack:           0.00
No. of Violating Paths:         0.00
Worst Hold Violation:           0.00
Total Hold Violation:           0.00
No. of Hold Violations:         0.00
-----------------------------------
```

Fig. 19 The delay report result

The synthesis results in figures (From Fig. 16 to Fig. 19) show that the full adder is synthesized successfully with the library created by our proposed flow. In addition, we make comparisons between our full-adders and the full adders in [17]. Table XII and XIII show that our power and delay results are better than the results in [17]. In terms of area of the designs without registers, our designs and the designs in [17] have the same area because both designs use the same semi-static structure. In case of registers, the area of our designs is larger than that in [17]. However, this difference can be improved because the actual area depends on the layout. About the power and delay, our designs show better results when compared to designs in [17]. These synthesis results are outstanding thanks to the optimization support of the Design Compiler tool. Furthermore, the designs in [17] follow a full-custom flow, which can be challenging to optimize for complex designs. Therefore, using semi-custom flow helps researchers save much time and effort.

Table. XII 1-bit full adder comparison results

Design	**Area (Transistor Count without registers)**	**Area (Transistor Count with registers)**	**Power (mW)**	**Delay (ns)**
Ours	54	315	0.034	0.72
[17]	54	193	7.46	5.097

Table. XIII 4-bit full adder comparison results

Design	**Area (Transistor Count without registers)**	**Area (Transistor Count with registers)**	**Power (mW)**	**Delay (ns)**
Ours	216	913	0.095	1.04
[17]	216	602	20.44	10.58

IV. Conclusion

In this paper, the semi-static NCL cell library is implemented based on our proposed flow. This library could be used for the synthesis of the NCL-based asynchronous designs at universities. The flow given in this work would be easy for researchers to implement. This flow also could help them update new cells and approach new design methods quickly. In addition, with this flow, we could solve the problem of the lack of standard NCL cell libraries that makes it difficult for students and researchers. The complete cell library includes 27 semi-static NCL cells, which are designed using 45nm CMOS technology and are used for the synthesis of the NCL based asynchronous designs by the Design Compiler tool of Synopsys.

Acknowledgment

We acknowledge the support of time and facilities from Ho Chi Minh City University of Technology (HCMUT), VNU-HCM for this study.

References

[1] D. L. d. Oliveira, O. Verducci, V. L. V. Torres1, R. Moreno2 and L. d. A. Faria., "Synthesis of QDI combinational circuits using null convention logic based on basic gates," Adv. Sci. Technol. Eng. Syst., vol. 3, no. 4, pp. 308–317, 2018.

[2] S. M. Nowick and M. Singh, "Asynchronous design-part 1: Overview and recent advances," IEEE Des. Test, vol. 32, no. 3, pp. 5–18, 2015.

[3] W. W. Kai1, N. B. Ahmad2, M. H. b. Jabbar, "Variable Body Biasing (VBB) based VLSI Design Approach to Reduce Static Power," International Journal of Electrical and Computer Engineering, Vol.7, No.6, December2017, pp. 3010~3019.

[4] J. Wu, "Null Convention Logic applications of asynchronous design in nanotechnology and cryptographic security," Ph.D. dissertation, Dept. Elect. Eng., Missouri Univ., 2012.

[5] A. A. Sakib, A. A. Akib and S. C. Smith, "Implementation of FinFET Based Static NCL Threshold Gates: An Analysis of Design Choice," 2020 IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS), 2020.

[6] A. A. Sakib and S. C. Smith, "Implementation of Static NCL Threshold Gates Using Emerging CNTFET Technology," ICECS 2020 - 27th IEEE Int. Conf. Electron. Circuits Syst. Proc., no. January 2021, 2020.

[7] B. G. Fawzy, M. M. Abutaleb, M. I. Eladawy, and M. Ghoneima "Strong Indication Full-Adder Circuit for NULL Convention Logic Automation Flows," Isc. 2018 - 18th Int. Symp. Commun. Inf. Technol., no. Iscit, pp. 416–421, 2018.

[8] S. Wei, E. Deng, J. Di, W. Kang and W. Zhao, "Nonvolatile NULL Convention Logic Pipeline using Magnetic Tunnel Junctions," in IEEE Transactions on Nanotechnology. 2021

[9] A. Caberos, S. Huang, and F. Cheng, "Area-efficient CMOS implementation of NCL gates for XOR-AND/OR dominated circuits," Asia Pacific Conf. Postgrad. Res. Microelectron. Electron., vol. 2017-Octob, pp. 37–40, 2018.

[10] P. Metku1, K. K. Kim, and M. Choi, "Novel area-efficient null convention logic based on CMOS and Gate Diffusion Input (GDI) hybrid," J. Semicond. Technol. Sci., vol. 20, no. 1, pp. 127–134, 2020.

[11] J. M. Emmert and S. A. VanDewerker, "EMC: Efficient Muller C-Element Implementation for High Bit-width Asynchronous Applications," 2021 IEEE International Midwest Symposium on Circuits and Systems (MWSCAS), 2021, pp. 816-819, 2021.

[12] M. Howard, N. Mize and J. Di, "Investigation and Comparison of Bus Alternatives for Asynchronous Circuits," Conf. Proc. - IEEE southeastcon, vol. 2018-April, pp. 1–2, 2018.

[13] K. K. Ponugoti, S. K. Srinivasan and S. C. Smith, "Hardware Trojan Design and Detection in Asynchronous NCL Circuits," 2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS), 2020, pp. 1-4, 2020.

[14] H. J. Lee and Y. Kim, "Low power Null Convention Logic circuit Design based on DCVSL," 2013 IEEE 56th International Midwest Symposium on Circuits and systems, 2013.

[15] K. Haque, C. Jakob and P. Beckett, "Low Power Spatial Computing Using Null Convention Logic," 2015 IEEE International Conference on Data Science and Data Intensive Systems, 2015, pp. 325-329, 2015.

[16] D. L. Oliveira, O. Verducci, L. A. Faria and T. Curtinhas, "A novel K convention logic (NCL) gates architecture based on basic gates," Proc. 2017 IEEE 24th Int. Congr. Electron. Electr. Eng. Comput. INTERCON 2017, pp. 1–4, 2017.

[17] A. Vakil, K. P. Jayadev, S. Hegde and D. Koppad, "Comparitive analysis of null convention logic and synchronous CMOS ripple carry adders, " 2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT), 2017, pp. 1-5.

[18] F. A. Parsan and S. C. Smith, "CMOS implementation comparison of NCL gates," Midwest Symp. Circuits Syst., pp. 394–397, 2012.

[19] P. Metku, K. K. Kim, Y. Kim and M. Choi, "Low-Power Null Convention Logic Multiplier Design Based on Gate Diffusion Input Technique," 2018 International SoC Design Conference (ISOCC), pp. 233-234, 2019.

[20] M. T. Moreira, P. A. Beerel, M. L. L. Sartori, and N. L. V. Calazans, "NCL synthesis with conventional EDA tools: Technology mapping and optimization," IEEE Trans. Circuits Syst. I Regul. Pap., vol. 65, no. 6, pp. 1981–1993, 2018.

[21] R. A. Guazzelli, M. T. Moreira, and N. L. V. Calazans, "A comparison of asynchronous QDI templates using static logic," LASCAS 2017 - 8th IEEE Lat. Am. Symp. Circuits Syst. R9 IEEE CASS Flagsh. Conf. Proc., pp. 1–4, 2017.

[22] R. B. Reese, S. C. Smith and M. A. Thornton, "Uncle - An RTL Approach to Asynchronous Design," 2012 IEEE 18th International Symposium on Asynchronous Circuits and Systems, May 2012.

[23] C. H. M. Oliveira, M. T. Moreira, R. A. Guazzelli, and N. L. V. Calazans, "ASCEnD-FreePDK45: An open source standard cell library for asynchronous design," 2016 IEEE International Conference on Electronics, Circuits and Systems (ICECS), Dec. 2016.
[24] M. T. Moreira and N. L. V. Calazans, "Design of Standard-Cell Libraries for Asynchronous Circuits with the ASCEnD Flow," 2013 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Aug. 2013.
[25] G. E. Sobelman and K. Fant, "CMOS circuit design of threshold gates with hysteresis," Proc. - IEEE Int. Symp. Circuits Syst., vol. 2, pp. 61–64, 1998.
[26] J. Bhasker and R. Chadha, Static timing analysis for nanometer designs: A practical approach. 2009.
[27] Y. Lai, "Synopsys Liberty User Guides and Reference Manual Suite," 2017, vol. 2, pp. 1069–1072, 2009.

Lac Truong Tri was born in Ho Chi Minh city, Vietnam. He received the Bachelor of Engineering in Electronics-Telecommunication Engineering major from Ho Chi Minh City University of Technology. He continues to study Master of Science degree in Electronics Engineering from Ho Chi Minh City University of Technology. His research interests are Null Convention Logic, Low power, and Asynchronous Design Method.

Toi Le Thanh was born in Tay Ninh City, Vietnam. He gave his M.S. degree from the Ho Chi Minh City University of Technology, Vietnam, in 2006. Since 2003, he has been a teacher at Ho Chi Minh City University of Food Industry (HUFI), Vietnam. His studies interests include Null Convention Logic, asynchronous circuit design, and low power circuit design.

Trang Hoang was born in Nha Trang city, Vietnam. He received the Bachelor of Engineering and Master of Science degree in Electronics-Telecommunication Engineering from Ho Chi Minh City University of Technology in 2002 and 2004. He received a Ph.D. degree in Microelectronics-MEMS from CEA-LETI and University Joseph Fourier, France, in 2009. From 2009–2010, he did postdoctorate research in Orange Lab-France Telecom. Since 2010, he has been a lecturer at Faculty of Electricals–Electronics Engineering, Ho Chi Minh City University of Technology. His field of research interest is in the domain of FPGA implementation, Speech Recognizer, IC architecture, MEMS, fabrication.