VLSI 설계를 위한 동시수행 하드웨어 자원 할당 및 바인딩 알고리듬

정회원 최 지 영*, 인 치 호**, 김 희 석***

A Simultaneous Hardware Resource Allocation and Binding Algorithm for VLSI Design

Ji-young Choi*, Chi-ho Lin** Hi-seok Kim*** *Regular Members*

요 약

본 논문에서는 VLSI 설계를 위한 동시수행 하드웨어 할당 및 바인딩 알고리듬을 제안한다.
제안된 알고리듬은 스케쥴링 결과를 입력으로 받아들이고, 각 기능 연산자에 연결된 레지스터 및 연결 구조가 최대한 공유하도록 제어스텝마다 연산과 기억 소자의 상호 연결 관계를 고려하여 기능 연산자, 연결 구조 및 레지스터를 동시에 할당 및 바인딩을 한다. 또한 레지스터 할당은 그래프 컬러링을 이용하여 최적의 레지스터 할당을 수행한다.
제안된 알고리듬은 실험 결과를 통해 기존의 기능 연산자와 레지스터의 수를 미리 정했거나, 분리하여 수행한 방식들과 비교함으로서 본 논문의 효율성을 보인다.

ABSTRACT

This paper proposes a simultaneous hardware resource allocation and binding algorithm for VLSI design.

The proposed algorithm works on scheduled input graph and simultaneously allocates binds functional units, interconnections and registers by considering interdependency between operations and storage elements in each control step, in order to share registers and interconnections connected to functional units, as much as possible. Also, the register allocation is especially executes the allocation optimal us-ing graph coloring techniques. Therefore the overall resource is reduced.

This paper shows the effectiveness of the proposed algorithm by comparing experiments to determine number of functional unit in advance or to separate executing allocation and binding of existing system

Ⅰ. 서 론

The required ASIC (Application Specific Integrated Circuit) that a special purpose and small production is essential CAD technology for reducing IC design to produce turn-around time and cost because necessary issue over the industrial of the miniaturization and quality.[1-4]

It is target purpose of CAD technology that a process from behavioral description of designed IC chip to design automation for chip manufacturing. The modern logic design (from RT level circuit description to gate level circuit

* 청주대학교 전자공학과 고수준논리합성 연구실(m7515103@kebi.com),
** 세명대학교 컴퓨터과학과 (ich410@venus.semyung.ac.kr),
*** 청주대학교 전자공학과 (khs8391@chongju.ac.kr)
논문번호 : 00159-0509, 접수일자 : 2000년 5월 9일
※ 본 연구는 과학기술부·한국과학재단지정 청주대학교 정보통신 연구센터의 지원에 의한 것입니다.

design)[5] or the layout design (form the technology library mapped each circuit element to placement and routing) is commonly used.[6] But the study on the high level synthesis that documentation feature of design flow, according to integration and complexity of VLSI be shorten design time, to reduce debugging time and error of various design automation for evaluation chip performance design of the beginning stages, is lacking.[7,8]

The definition of high-level synthesis was consisted of scheduling, allocation, binding from the behavioral description of designed to create structure of RT register transfer level for limiting constraints and satisfied target function. The scheduling consist of assigning behavioral description each operation to control step.[9] But it had developed algorithms solution of the limited application field for very scheduling process is considerable items conditional branch, pipeline, loop and so on. The typical methods is the easiest ASAP(As Soon As Possible) to each operation is scheduled to occur at the earliest possible time. ALAP scheduling is defined as scheduling each operation as late as possible. The algorithm is very similar to the ASAP scheduling algorithm except that we work from the bottom of the data flow graph toward the top and we start with the last control step and work toward the first control step. and Force-Direct Scheduling is common used scheduling technology under time constraint. Finally list scheduling method sequentially determined execution time of operation as priority function. The design flow for VLSI shown in Figure 1.

The allocation is assigned so that minimize area of implement hardware, to operation as functional unit, to variable as register, between operation and register as interconnection assigned to bus and multiplexer.[10] The typical methods is greedy allocation[11], left edge algorithm, clique partitioning and so on. The greedy allocation which is assigned hardware resource for executive variable and operation each at the ti-me interval. The clique partitioning which is applied allocation

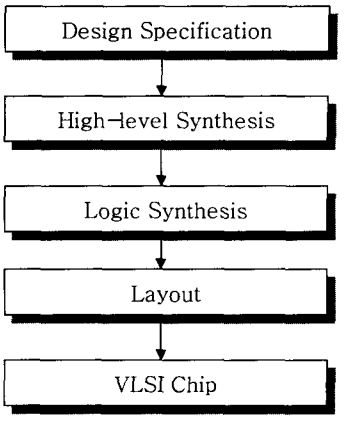


Fig. 1 Design flow for VLSI

of operation and memory as the approach method for using the graph theory.

As the existed method of a hardware allocation, HAL[12] system is together executes allocation of functional unit and scheduling to suppose one each type of functional unit, assigned allocation and binding to register and interconnection to use clique partitioning method. Splicer system[13] is together executed scheduling number of function unit which fixed the states in advance, the number of register minimized interconnection using the method of branch and bound. But this systems cannot to obtain optimal design. Also, REAL[14] system allocates minimum register using the left-edge algorithm in this case not considering mutual exclusion. But does not consider influence interconnection, not deal with the allocation of function unit and interconnection. The existed methods that the first, the number and type of the register and functional unit is fixed in advance, the second, it execute to separate the allocation and binding. Therefore, the existing approach methods do not optimal methods.

Consequently this paper proposes a simultaneous hardware resource allocation and binding for VLSI de-sign. The proposed algorithm works on the scheduled input graph and simultaneously allocates and binds registers, functional units and interconnections in stages by considering interdependency between operations and storage

element in each control step, in order to share registers and interconnections that are connected to functional units, as much as possible. Therefore it resolved the existing problem at the same time and optimized total area cost. Also the register allocation assigns for using graph coloring technique.

The structure of this paper is introduction of the first section, section 2 describes overall contents of the propose a simultaneous hardware resource allocation and binding algorithm, section 3 is adequate of this algorithm by comparing the result of our experiments with those of existing system, section 4 is composed of the conclusion of this paper.

Ⅱ. The proposed a simultaneous hardware resource allocation and binding algorithm

As this paper, the proposed a simultaneous hard-ware resource allocation and binding algorithm is shown in Figure 2. The input works on the scheduled input graph and functional units calculated in order to allocate and bind for mobility of all operation at preprocessing. The mobilities of operation are computed by investigating data dependency. From the first control step allocate register, functional unit, interconnection in stages, after calculating

```
Input_mobility( );
 while(control step) {
        Register_allocate( );
        /* allocates and binds register */
        Function_allocate( );
        /* allocates and binds
        function unit */
        Mobility_modify( );
        /* modification of mobility */
 Inter_allocate_merge( );
 /* allocates and merges
 interconnection */
 }
```

Fig. 2 The simultaneous hardware resource allocation and binding algorithm

mobilities of operation. At this point, providing that allocation and binding of functional units finished, the mobilities of existing operation modify at the next control steps. The interconnection merging executes, after allocation and binding, as control step on the whole

The overall flow of the resource allocation and binding is shown in Figure 3. The registers allocation and binding, functional units allocation and binding, mobility modification, interconnection binding can sequentially put into the form of a diagram.

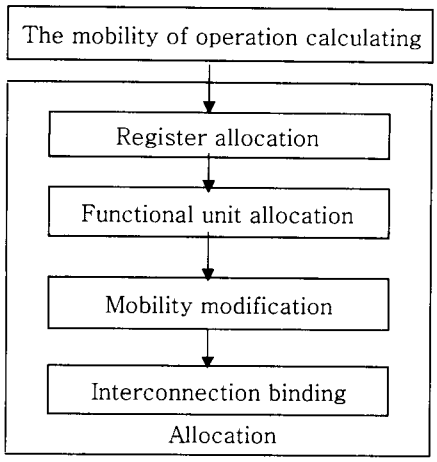


Fig. 3 The overall flow of the simultaneous hardware resource allocation and binding

1. The allocation and binding of registers

The register allocation and binding is executed in stages each control step, alike the next description.

The first register classify to allocate type. Namely, variable or constant, before the control step executed output of functional unit, classify. The second, it allocate that a register, according to classify type. In this case variable, if the first control step, new register assign and if the next control step, register allocate that reused before the control step.

Namely. The overlapped register allocation but existing another control step executes optimal register allocation using graph coloring techniques.

After the life-time composed according to arrange input created interference graph. When it suppose usable register number is K, if the node dont exist with degree(n) < k (n: node, k: usable register number) insert node of stack in position in stead of the spill. The coloring execute optimal coloring that node dont coloring after it suppose that color is able to use at the stack when the node pop at stack, if color not useable. The coloring algorithm shown in Figure 5. In this case constant, it excluded register allocation. In this case of the output of functional units that was performed at the previous control step, investigate whether it is the input of the other operation, allocation at register after considering the types of the functional units and the type of operation receiving input. If it is not the input of other operation, allocation at register after considering the type of the function operator. The allocation and binding algorithm of register shown in Figure 4.

If the loop exists, allocate such as Figure 6, the register that is used at the beginning and ending of a loop. That is, each variable V1, V2, V3 is allocated as register R1, R2, R3 the first control step, At the same time they are allocated as register R1, r2, R3 which or the same register at the last control step.

Table 1 shows allocation of the registers which is come under control step1 and control step2.

```
Register_allocate( ) {
Input_type_search( );
/* class type of input value */
if(control step >1) then {
Reg_chart_search( );
Variable_allocate( );
/* allocates the register as variable */
In_fun_allocate( );
/* register allocated as the
output of functional unit */
Register_chart( );
}
}
```

Fig. 4 The allocation and binding algorithm of register

```
if(node) {
color_stack_pop( ); /* Pop stack */
if(degree(n) > k) {
Non_coloring( );
/* Not coloring */
Spill_code( );
/* Insert spill code */
else
Coloring( ); /* Coloring */
}
}
```

Fig. 5 The coloring algorithm

Table 1. Register_chart

CS	Register				
1	R1	R2	R3		
2	R1	R2	R3	R4	R5

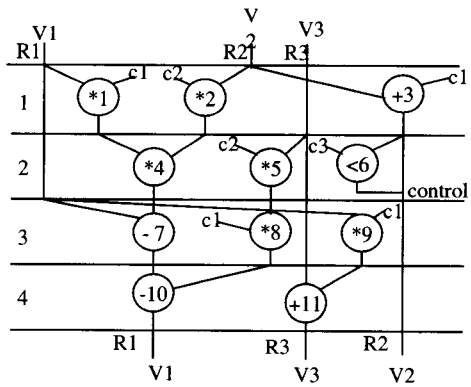


Fig. 6 Register allocation in existence loop

2. The allocation and binding of functional units

After performing of allocation and binding of register, performance the allocation of the functional unit about each operation which are being, now, at the control step. That is, to choose the functional unit, in the cell library the functional unit, which are satisfied with the computed performance time of each computed of operation and has the smallest area, investigate the five variable of self distributed number, relative distributed number, self fixed number,

relative fixed number, and self mobility that will allocate or bind the functional unit. First, self distributed number is at the control step such as operation which is going to allocate the functional units and represents the number of operation which has the same type. Relative distributed number represents operation which is going to allocate the functional units and maximum number of operation which is at the other control step, has the same type. Also , Self fixed number is the maximum of operation of which mobility is zero when it is investigated the mobility of operation which exist at the same control step and has the same type, Relative fixed number is the number of maximum of which mobility is zero per a control step when it is investigated the mobility of operation which has the same type and is at the other control step with the operation that is going to allocate functional units. Self mobility is t-he mobility of operation that is going to allocate functional units.

An example of self distributed number used when it is allocated and binding shown in Figure 7. The self distributed number of multiplication operation which is at the first control step is one.

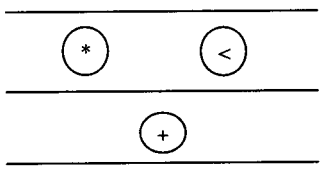


Fig. 7 A example of self distributed number

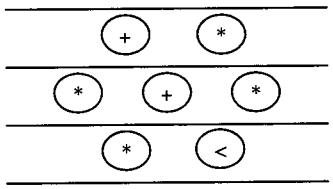


Fig. 8 A example of relative distributed number

An example of relative distributed number shown in Figure 8. The relative distributed number, if multiplication operation which is at the first control step is determined as follows. The number of multiplication operation which is at the second control step is two, the number of multiplication operation which is the third control step is one, that has the maximum value, so the relatives distributed number of multiplication is two.

An example of self fixed number shown in Figure 9. It is possible to execute at the second control step. The multiplication operation which is at the first control step can be also performed at the control step, so the mobility is one. Therefore the self fixed number is zero.

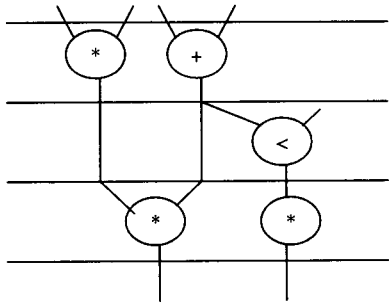


Fig. 9 A example of self fixed number

We showed you the examples of functional five variable from the Figure 7 to the Figure 9. Secondly, as the example which allocate the practical functional units, we are going to look into the process which allocate functional units to the comparator of the figure 6. Initial input of comparator was scheduled at the second control step. But on investigation the variables of comparator, It is possible to allocate the functional units which occupies the two control steps (the second control step and the third control step) at the comparator.

That is, it is the multi-cycling. The multi-cycling is that one puts on many control step. Therefore, choose the functional units of which, delay time is the same as correction time on the cell-library, has a small as much as possible The delay time is time except for the connection structure time and the register delay time on the two control step. The allocation algorithm of functional units shown in Figure 10.

The overall functional units and registers table (Function_register_chart) shown in Table 2. In the case of the first control step, allocate the functional units after considering self distributed number, self fixed number, relative distributed number, relative fixed number and next from the control step, investigate the functional unit for operation of existence first of all. If, there is a suitable functional units at the operation allocated the function unit, allocated them at once, or not, allocate functional units after first investigate the five variable above.

```
Function_allocate( ) {
if (control step ==1)
then investigate fore variable( );
if(same operation && same mobility) {
Existed_function_allocate( );
Function_unit_allocate( );
Function_register_chart( );
}
}
```

Fig. 10 The allocation and binding algorithm of function unit

Table 2. Function_register_chart

	FU1(*)			FU2(*)			FU3(+)			FU4(<)			FU5(-)		
	op	A	b	op	a	B	op	a	b	op	a	b	op	a	b
S1	1	R1	c1	2	c2	R2	3	R3	c1						
S2	4	R4	R5	5	c2	R3				6	R2	C3			
S3	8	c1	R5	9	R1	c1							7	R1	R4
S4							11	R3	R5				10	R1	R4

3. A step modification the mobility

When you allocate the functional unit for operation, in the case of using the multi-cycling which uses many control step, the mobility arbitrate for all operation of dependence the operation. plural control step is the same with the delay correction time on the library.

For example, Figure 11(a) is *1 allocates and binds functional units for using multiple control step, self distributed number 1 and self mobility 2, self fixed number, relative distributed number, relative fix number, all 0. There by the +2 of input received output of *1 is mobility conversed from 1 to 0. The result of allocation and binding shown in Figure 11(b).

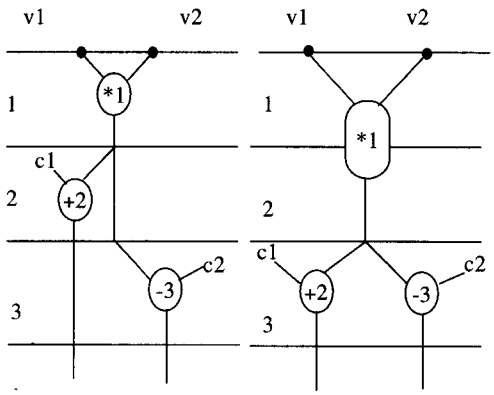


Fig. 11 A example of modification the mobility
(a) Before the modification of mobility
(b) After the modification of mobility

4. The binding of interconnection

Perform the allocation of interconnection after performing the allocation of functional units. In this case first control step, allocate new multiplexer for each operation. Second, from control step, investigate the types of functional units and input, and then find out the same type as much as possible finally allocate the multiplexer. Third, investigate the input number of multiplexer. If, there is one, omit multiplexer, or not investigate the control step of each multiplexer. If the input value is the same, it will be merged even thr-ough the control step is duplicated or not. At this time, the multiplexer which has been merged stand face to face with bus. As an example, seeing the figure 6, we can find out that *5 which is at the second control step and *2 which is at the first control step have been allocate and bind for the same functional unit (FU2). Therefore, the interconnection of *5 can allocate and bind as the same interconnection of *2.

At this time, for interconnection of two distributed allocation and binding is executed for considering type. The overall result of allocation

and binding for registers, functional units, interconnections shown in Figure 12.

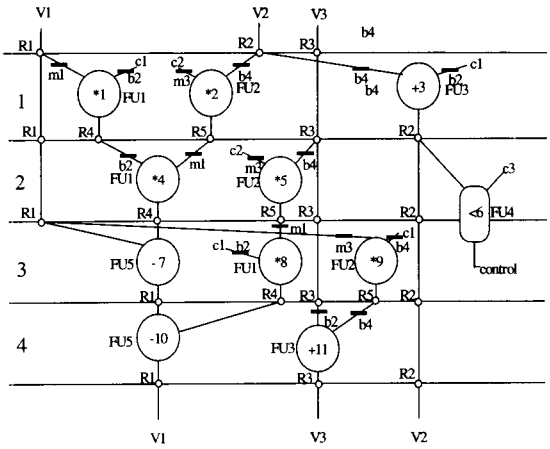


Fig. 12 The overall result of a simultaneous hardware resource allocation and binding algorithm

Ⅲ. The result of experiments

This paper executed allocation algorithm result is compared by HAL system result, for exactly comparison, using HAL, Splicer system application extraction result is received input.

1. The Differential Equation

The result of comparing the area cost about the differential equation with HAL, Splicer, REAL system shown in Table 3.

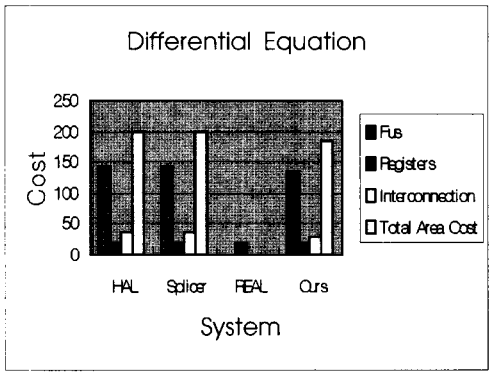
This algorithm show the effectiveness to reduce total area cost than HAL system allow to execute separate allocation and binding about high-level synthesis. The area cost of register is almost three system sameness but functional unit reduce remarkable 7.9 %. The Splicer system is similar to the whole HAL system. Especially the register is reduced 1.5%. When our algorithm is comparing the three system, total area cost is reduce 8.5%

2. The Fifth-Order Elliptic wave filter.

The HAL, Splicer, REAL system comparative result of area cost of the fifth elliptic wave filter to adopt as the standard benchmark model for

Table. 3 Comparing experiment of differential equation

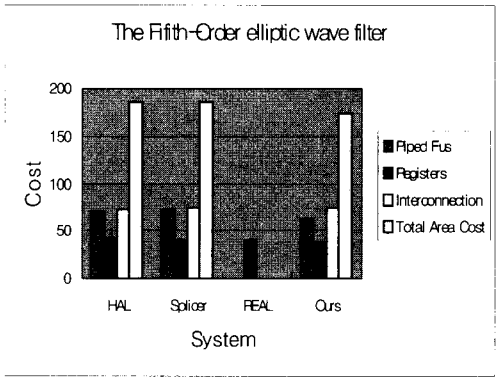
	HAL	Splicer	REAL	Ours
FUs	145	145	-	135
Register	18	17	18	18
Interconnection	37	35	-	30
Total Area Cost	200	197	-	183



High Level synthesis Workshop as benchmark model shown in Table 4. The HAL system is the piped functional units area cost was reduced 11.4%, But inter-connection area was many or few increased, interconnection of layout design is not the weakness by high-level design is not considered, Also register area cost was reduced 9.6%, consequently total area cost is reduced. The Splicer and REAL system is the same cost ratio register like the differential equation. The total area cost reduced 6%. However, REAL system is not considering the functional units and interconnection units.

Table. 4 Comparing experiment of fifth-order elliptic wave filter

	HAL	Splicer	REAL	Ours
FUs	70	72	-	62
Register	42	40	40	38
Interconnection	73	74	-	74
Total Area Cost	185	186	-	174



Ⅳ. Conclusion

This paper shows algorithm that perform a hard-ware resource allocation and binding at the same time.

A simultaneous hardware resource allocation and binding algorithm is implemented AVION SYS, performs with the characters as follows. First, from control step, allocate registers and functional units by stages and then perform interconnection merge after performing interconnection binding.

It simultaneously executed to allocate and bind for dependency relation. as it shown comparison for the sake performance estimating, the experiment of application of the differential equation is reduced register is almost three system sameness but functional unit reduce remarkable 7.9 %. The Splicer system is similar to the whole HAL system. Especially the register is reduced 1.5%. When our algorithm is comparing the three system, total area cost is reduce 8.5%. The experiment of the fifth-order elliptic wave filter, standard benchmark model, but interconnection area cost increased 1%, while, the register area cost reduced 9.6% (Especially the HAL system).

Consequently it showed effectiveness of this algorithm, that is small area cost, compared to other existing systems which take the number of functional units in advance or allow to execute separate allocation and binding. Finally, the hardware cost functional unit values was shows effectiveness, which minimum for the ultimate purpose of high-level synthesis techniques.

Also, after this study project will precede the study for the anticipation and estimation which based on a simultaneous hardware resource allocation and binding algorithm for VLSI design.

Reference

[1] Breuer, M. A., 1975, Digital System Design Automation, Computer Science Press, Inc.

[2] Rubin, S. M., Computer Aided for VLSI Design, Addison_Wesley.

[3] Shiva, S. G., Jan. 1983, Automatic Hardware Synthesis, Proceedings of the IEEE, Vol.71(1), 76-87

[4] Gajski, Daniel D., 1988, Silicon Compilation, Addison-Wesley

[5] Brayton, R. K. Sangiovanni-Vincentelli, A. L. and G. D. hatchtel, Feb. 1990 Multi-level Logic Synthesis, Proceedings of the IEEE, Vol.78(2), 264-300.

[6] Kuh, E. S. and Ohtuski, T., Feb 1990, Recent Advance in VLSI Layout, Proceedings of the IEEE, Vol78(2)

[7] McFarland, M. C., Paker,A. C. and Camposano, R., Feb 1990 The High-Level Synthesis of the Digital System Proceedings of the IEEE, Vol. 78(2), 301-318.

[8] Hitchcock, C. Y., and D. E. Thomas, 1983, A Method for Automatic DataPath synthesis, Proc. of the 20th Design Automatic Conference(DAC), 484-489.

[9] Camposano, R., Oct. 1990., From Behavior to Structure: High-Level Synthesis, IEEE Design & Test of Computer, 8-19.

[10] James R. Armstrong, F. Gall Gray., 1993, Structured Logic Design With VHDL, 374 -393

[11] Daniel D. Gajski, Nikil D. Dutt, and Allen C-H Wu., 1992, High -Level Synthesis : introduction to chip and system design, 272 -283.

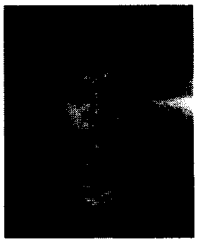
[12] Paulin, P., Knight. J. and Girczyc, E., 1986, HAL: A Multi-Paradigm Approach to Automatic Data Path Syn thesis, Proc. of 23rd

DAC, 263-270.

[13] Pangrle, B., 1988, Splicer : A Heuristic Approach to Connectivity Binding, Proc. of the 25th Design Automation Conf, 536-541.

[14] Kurdahi, F. J. and A. C. Parker, 1987, REAL : A Program for register allocation, in Proc. of the 14th Design Automation Conf, 210-215.

최 지 영(Ji-young Choi)



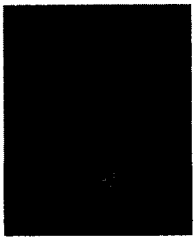
1997년 2월 : 세명대학교 전자계산학과 졸업 (이학사)

1999년 2월 : 세명대학교 전산정보학과 졸업 (이학석사)

2000년 3월~현재 : 청주대학교 전자공학과 박사과정

<주관심 분야> CAD, 알고리듬, 상위레벨합성, 저전력 설계

인 치 호(Chi-ho Lin) 정회원



1985년 : 한양대학교 전자공학과 졸업(공학사)

1987년 : 한양대학교 전자공학과 졸업(공학석사)

1996년 : 한양대학교 전자공학과 졸업(공학박사)

1992년~현재 : 세명대학교 컴퓨터과학과 부교수

<주관심 분야> VLSI CAD, ASIC 설계, CAD 알고리듬 등

김 희 석(Hi-seok Kim)

1977년 2월 : 한양대학교 전자공학과 졸업 (공학사)

1980년 2월 : 한양대학교 전자공학과 졸업 (공학 석사)

1985년 8월 : 한양대학교 전자공학과 졸업(공학박사)

1987년 9월~1988년 9월 : 미국 University of Colorado at Boulder 객원 교수

1996년 8월~1997년 7월 : 미국 University of Callifornia at Irvine 객원 교수

1981년 3월~현재 : 청주대학교 전자공학과 교수

<주관심 분야> CAD, 컴퓨터 구조, 컴퓨터 알고리듬