ISSN : 1226-7244 (Print)
ISSN : 2288-243X (Online)
논문번호 23-03-10
j.inst.Korean.electr.electron.eng.Vol.27,No.3,273~279,September 2023
https : //doi.org/10.7471/ikeee.2023.27.3.273
61

# 이미지처리에서 디지털 필터를 구현하기 위한 가변모드 동기 발생기의 설계

# Design of a Variable-Mode Sync Generator for Implementing Digital Filters in Image Processing

정 세 민*†, 한 시 연*†, 강 봉 순*★

Semin Jung*†, Si-Yeon Han*†, Bongsoon Kang*★

## Abstract

The use of line memory is essential for image filtering in image processing hardware. After input data is stored in line memory, filtering is performed after synchronization to use the stored data. A sync generator is used for synchronization, and in the case of a conventional sync generator, the input sync signal is delayed by one row of the input image. If a signal delayed by two rows is required, it is necessary to connect two modules. This approach increases the size of the hardware and cannot be designed efficiently. In this paper, we propose a sync generator that generates multiple types of delayed signals by adding a finite state machine. The hardware design was coded in Verilog HDL, and performance is verified by applying it to image processing hardware using field programmable gate array board.

## 요 약

이미지처리 하드웨어에서 이미지 필터링을 진행할 때 line memory의 사용은 필수적이다. line memory에 입력 데이터를 저장한 후 저장된 데이터를 사용하기 위해 동기를 맞춘 후 필터링을 진행한다. 이때 동기를 맞추기 위해 동기 발생기를 사용한다. 기존 동기 발생기의 경우 입력 동기 신호를 입력으로 들어오는 이미지의 1행만큼 지연시킨다. 만약 2행만큼 지연된 신호를 얻기 위해서는 모듈 2개를 연결하여 사용해야 한다. 해당 방식으로 하드웨어 설계 시 하드웨어의 크기가 커져 효율적으로 설계할 수 없다. 따라서 본 논문에서는 finite state machine을 추가하는 방식을 사용하여 여러 종류의 지연 신호를 생성하는 동기 발생기를 제안한다. 하드웨어 설계는 Verilog HDL로 코딩하였으며, field programmable gate array 보드를 이용하여 이미지처리 하드웨어에 적용하여 성능을 검증하였다.

*Key words : Sync generator, Finite state machine, Auto detection, Verilog, Xilinx*

* Dept. of Electronics Engineering, Dong-A University
★ Corresponding author
E-mail : bongsoon@dau.ac.kr, Tel : +82-51-200-7703
† These authors contributed equally to this work.
※ Acknowledgment
This work was supported by the Ministry of Education of the Republic of Korea and the National Research Foundation of Korea(NRF-2023R1A2C1004592)
Manuscript received Aug. 25, 2023; revised Sep. 17, 2023; accepted Sep. 20, 2023.
This is an Open-Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License(http://creativecommons.org/licenses/by-nc/3.0) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.

## Ⅰ. 서론

이미지처리 하드웨어에서 이미지 필터링을 진행할 때 line memory의 사용은 필수적이다[1]. 예를 들어 3×3 윈도우를 이용하여 필터링을 진행할 때 이미 지나간 데이터는 사용할 수 없으므로 2개의 line memory를 이용하여 데이터를 저장한다[2]. 저장된 데이터를 사용하기 위해서는 연산되는 데이터의 동기를 맞춰준 후 필터링을 진행한다. 이때 sync generator를 이용하여 입력으로 들어온 동기 신호인 Horizontal Active Video(HAV), Vertical Active Video(VAV)의 1H, 2H delay 신호를 생성한다. H delay는 Horizontal delay를 의미하며, 해당 신호가 필요한 이유는 이미지 필터링을 진행할 때 3×3 윈도우는 3-line의 데이터를 연산하기 때문에 Horizontal delay 신호를 이용하여 연산되는 데이터와 동기를 맞춰주어야 하기 때문이다. 기존의 Fixed-Mode Sync Generator(FMSG)는 입력으로 들어온 i_hav 그리고 i_vav 신호의 1H delay 신호를 출력으로 내보낸다. 따라서 2H delay 신호를 만들어 주기 위해서는 FMSG 모듈 2개를 사용하는 방식으로 지연 신호를 생성한다[3]-[5]. 해당 방식으로 신호를 생성하면 하드웨어의 크기가 모듈의 크기만큼 증가하게 된다. 만약 5×5, 7×7 윈도우[6]를 사용한다면, 필요한 지연 신호가 증가하고 FMSG 모듈 4, 6개를 사용하므로 효율적인 하드웨어를 구현할 수 없다. 그리고 FMSG의 경우 지연 신호를 생성할 때 i_hav 그리고 i_vav 신호뿐만 아니라 이미지의 width와 blank 정보가 필요하기 때문에 유동적인 사용이 어렵다.

본 논문에서는 모듈을 연결하여 사용하는 방식이 아닌 내부에서 Finite State Machine(FSM)을 추가하여 다양한 종류의 지연 신호를 생성하는 Variable-Mode Sync Generator(VMSG)를 제안한다. 제안하는 VMSG의 경우 모듈을 연결하여 사용하는 방식이 아닌 FSM[7]을 추가하는 방식으로, 추가되는 FSM을 구성할 하드웨어 자원을 요구하기 때문에 모듈을 추가하는 방식보다 하드웨어 자원을 적게 사용한다. 또한 입력 이미지가 변경될 때마다 이미지의 width와 blank 정보를 변경해야 하는 FMSG와 다르게, 입력 i_hav 신호를 이용하여 이미지의 width, blank 정보를 검출하는 auto detection 기능을 추가하여 들어오는 입력 신호에 따라 이미지 정보를 검출하여 유동적으로 사용할 수 있다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 FMSG와 VMSG의 동작 방식을 설명하며, 실제 이미지 필터링 하드웨어에 적용하여 Register Transfer Level (RTL) 검증을 진행한다. Ⅲ장에서는 Xilinx Vivado 2023.1을 이용해 구현된 하드웨어 합성 결과를 비교한다. 마지막으로 Ⅳ장에서 결론을 서술하였다.

## Ⅱ. 본론

### 1. Fixed-Mode Sync Generator

FMSG는 i_hav 그리고 i_vav를 이용하여 HAV 상승, VAV 상승, 하강 에지를 검출하는 신호를 생성한다. 또한 i_width, i_blank 그리고 두 신호를 합한 1H total delay를 바탕으로 VAV counter를 이용하여 vav_flag 신호를, HAV counter로 total_flag, width_flag 그리고 blank_flag 신호를 생성한다. 생성된 신호들은 지연 신호를 생성하는 FSM에서 입력으로 사용된다.

가. VAV generator state machine (FMSG)

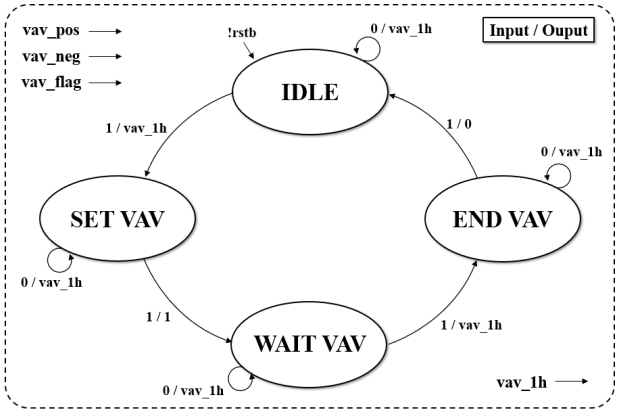


Fig. 1. State diagram of VAV generator (FMSG).
그림 1. VAV 발생기 상태도 (FMSG)

그림 1은 VAV generator의 상태도이다. IDLE, WAIT VAV 상태에서 vav_pos와 vav_neg 신호를 입력으로 사용하며, 입력 신호가 high일 때 상태를 천이한다. SET VAV 상태는 vav_flag 신호를 입력으로 받아 vav_flag가 high일 때 vav_1h가 1로 출력되고, END VAV 상태는 vav_flag가 high일 때 vav_1h를 0으로 출력한다. 해당 경우를 제외하고 출력은 피드백되는 구조이다. vav_1h는 i_vav 신호가 1H delay 된 신호이다.

나. HAV generator state machine (FMSG)

그림 2는 HAV generator의 상태도이다. IDLE 상태에서 입력 hav_pos가 high일 때 상태를 천이시킨다. WAIT HAV 상태는 출력 신호를 1H delay 시키기 위해

1H total delay를 알려주는 total_flag를 입력으로 사용하여 해당 신호가 high라면 hav_1h 신호가 1로 출력된다. WIDTH 상태에서 width_flag가 high라면 hav_1h를 0으로 출력하고 BLANK 상태로 천이한다. BLANK 상태에서 blank_flag 그리고 vav_flag가 결합한 신호의 상태가 x1이면 IDLE, 00이면 state를 유지, 10일 때 WIDTH 상태로 천이하며 hav_1h를 1로 출력한다. 출력값을 변경하는 3가지 경우를 제외하고 출력은 피드백되는 구조이다. hav_1h는 i_hav가 1H delay 된 신호이다.

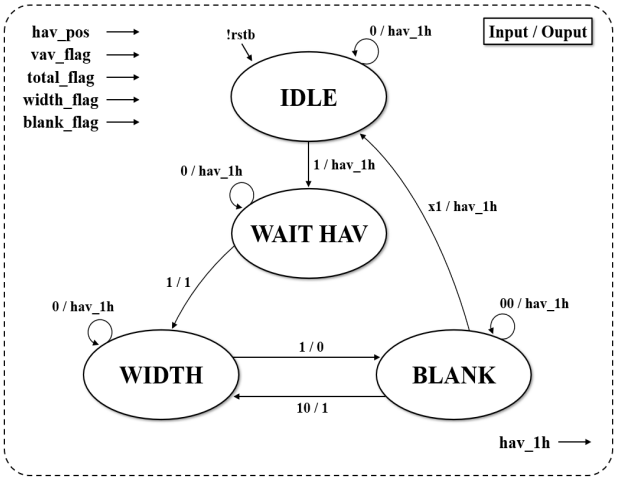


Fig. 2. State diagram of HAV generator.
그림 2. HAV 발생기 상태도

다. FMSG의 동작방식

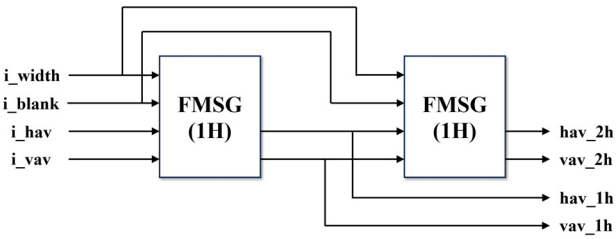


Fig. 3. Block diagram of FMSG (1H).
그림 3. FMSG (1H)의 하드웨어 블록도

그림 3은 1H, 2H delay 신호를 생성하기 위한 FMSG의 동작 원리를 블록도로 나타낸 것이다. FMSG 모듈은 입력으로 i_hav 그리고 i_vav와 이미지 정보인 i_width 그리고 i_blank를 받는다. 첫 번째 모듈에서 i_hav 그리고 i_vav 신호의 1H delay 된 신호를 생성하고 생성된 출력 신호를 두 번째 모듈의 입력으로 넣어주어 2H delay 신호를 생성한다.

그림 4는 그림 3에서 사용된 FMSG의 내부 구조이다. 입력 신호를 받아 flag 신호를 생성한 후 VAV, HAV generator에서 입력으로 사용하여 hav_1h 그리고 vav_1h 신호를 생성한다.

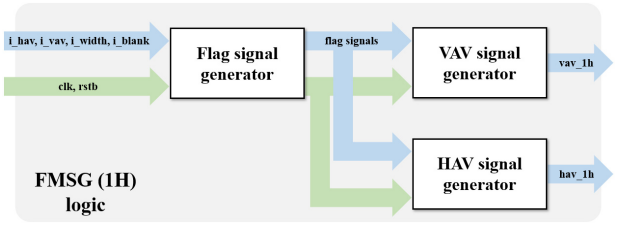


Fig. 4. Pipelined architecture of FMSG.
그림 4. FMSG의 하드웨어 구조

## 2. Variable-Mode Sync Generator

제안하는 VMSG에서 에지 신호를 생성하는 방식은 FMSG와 동일하다. 하지만 이미지의 정보를 받아 동작하는 것이 아닌 i_hav 신호를 이용하여 이미지 정보를 검출하여 flag 신호를 생성한다. 그리고 추가 지연 신호를 생성할 때 VAV의 경우 내부 FSM의 선택 신호를 추가하고, HAV의 경우 FSM을 추가하는 방식을 사용한다.

가. Auto detection state machine

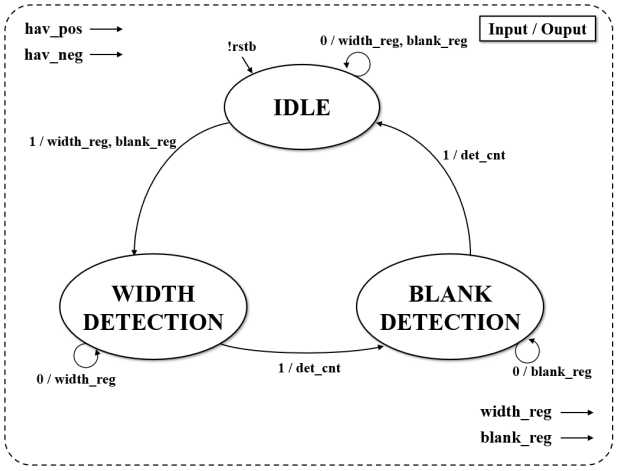


Fig. 5. State diagram of auto detection.
그림 5. Auto detection 상태도

Auto detection의 경우 그림 5의 상태도를 통해 동작 원리를 확인할 수 있다. IDLE 상태에서 입력 hav_pos가 high면 WIDTH 상태로 천이하고 counter를 이용해 width의 크기를 측정한다. hav_neg가 high일 때 측정을 종료하며, BLANK 상태로 천이하여 blank를 측정한다. 측정된 이미지 정보는 레지스터에 저장되고 flag 신호를 생성할 때 사용된다.

나. VAV generator state machine (VMSG)

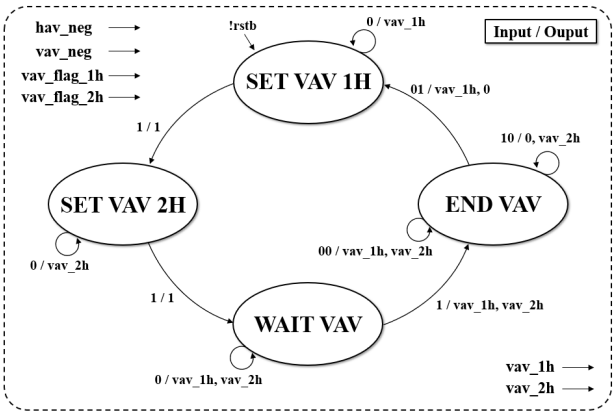


Fig. 6. State diagram of VAV generator (VMSG).
그림 6. VAV 발생기 상태도 (VMSG)

그림 6은 VAV generator의 상태도이다. SET VAV 1H, SET VAV 2H 상태에서 입력 신호인 hav_neg, vav_flag_1h가 high일 때 vav_1h, vav_2h가 1로 출력된다. WAIT VAV 상태는 vav_neg 신호를 기다리며, 해당 신호가 들어오면 END VAV 상태로 천이 후 입력 vav_flag_1h 그리고 vav_flag_2h가 결합한 신호의 상태가 10이면 vav_1h 신호를 0으로 출력하며 상태를 유지하고, 01일 때, vav_2h 신호를 0으로 출력하며, SET VAV 1H 상태로 천이한다. END VAV state에서 사용하는 flag 신호는 1H, 2H total delay 된 시점을 알려주는 신호이다.

다. HAV generator state machine (VMSG)

VMSG에서 HAV의 지연 신호를 생성하기 위해 사용하는 FSM의 구조는 FMSG와 동일하다. 해당 FSM은 그림 2에서 확인할 수 있다. 하지만 사용하는 flag 신호를 달리하며, 동일한 구조의 FSM 1개를 추가로 사용한다. 각각의 FSM에서 hav_1h, hav_2h 신호가 출력된다. 단, flag 신호를 생성할 때 사용되는 HAV counter의 경우 2H delay 신호를 생성하는 FSM에서 제어한다.

라. VMSG의 동작방식

Fig. 7. Block diagram of VMSG (Up to 2H).
그림 7. VMSG (Up to 2H)의 하드웨어 블록도

그림 7은 1H, 2H delay신호를 생성하는 VMSG를 블록도로 나타낸 것이다. auto detection 기능이 추가되었기에 i_hav 그리고 i_vav 두 입력만을 받는다. 또한 FSM을 추가하는 방식으로 지연 신호를 추가로 생성하기 때문에 하나의 모듈에서 모든 지연 신호를 생성하는 것을 볼 수 있다.

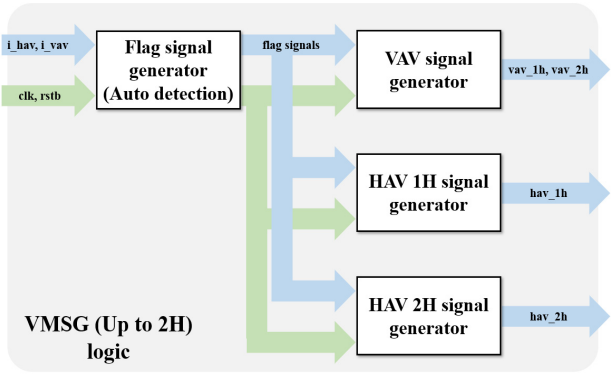


Fig. 8. Pipelined architecture of VMSG.
그림 8. VMSG의 하드웨어 구조

그림 8은 그림 7에서 사용된 VMSG의 내부 구조이다. 입력 신호를 통해 이미지 정보를 검출하고 해당 신호를 이용하여 flag 신호를 생성한다. flag 신호는 지연 신호를 생성하는 FSM에서 입력으로 사용되며, VAV generator에서는 vav_1h, vav_2h 신호를 생성하고 HAV 1H generator를 이용하여 hav_1h 신호를, HAV 2H generator로 hav_2h 신호를 생성한다.

## 3. RTL 검증

제안하는 VMSG의 성능을 검증하기 위해 3×3 LPF를 사용하는 이미지처리 하드웨어에 적용하여 256×256 Circular Zone Plate (CZP)[8]와 Lena 이미지의 필터링을 진행하였다. 사용한 LPF의 계수는 수식 1에서 볼 수 있으며, RTL 검증을 위해 Mentor Graphic 사의 ModelSim을 이용하였다[9]. 그림 9는 VMSG의 기능을 검증한 결과를 보여준다. 상단의 좌측 이미지는 입력 CZP이고, 우측 이미지는 필터링된 CZP의 RTL 검증 결과이다. 중간의 1차원 파형은 입력과 필터링된 CZP의 128행을 나타낸 것이다. 필터링된 파형(붉은색)을 보면 중앙의 저주파 신호는 거의 손실 없이 통과하지만, 좌우의 가장자리 신호들은 고주파 신호이기에 LPF 통과 후에 감쇄되는 것을 확인할 수 있다. 하단의 좌측 이미지는 입력 Lena이고 우측 이미지는 필터링된 Lena의 RTL 검증 결과이다. LPF에 의해 필터링된 이미지는 확대한 모

자의 깃(붉은 원)과 챙 부분(푸른 원)에서 에지 성분들이 감소된 것을 볼 수 있다.

$$LPF = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad (1)$$

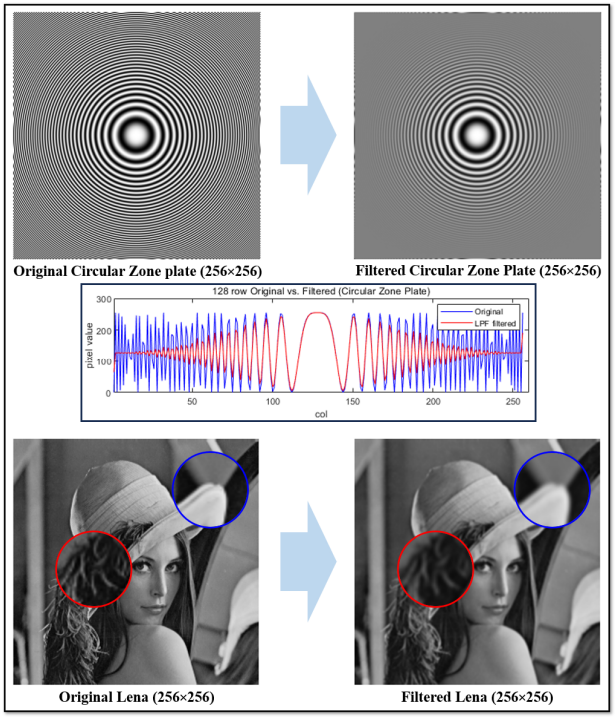


Fig. 9. ModelSim verification of LPF filtering.
그림 9. ModelSim을 이용한 LPF 필터링 검증

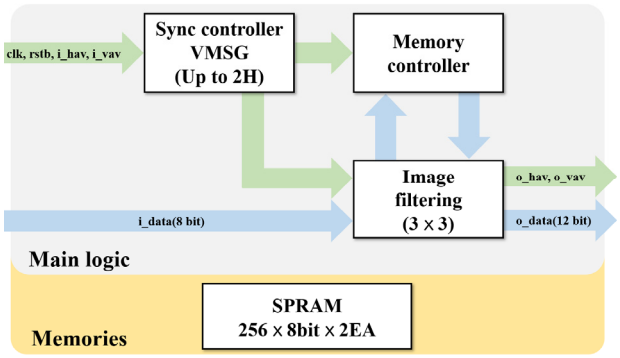


Fig. 10. Piplined architecture of LPF filtering.
그림 10. LPF 필터링 하드웨어 구조

그림 10은 이미지 필터링 하드웨어의 구조를 보여준다. 3×3 윈도우를 사용하기에 2개의 SPRAM line memory를 사용한다. Sync controller에서는 1H, 2H delay 신호를 생성하기 위해 본 논문에서 제안하는 VMSG를 사용하였다. 생성된 신호를 이용하여 Memory controller에서 line memory에 저장된 데이터의 동기를 맞춰준 후 Image filtering에서 수식 1의 LPF를 이용한 연산을 진행하여 그림 9의 필터링된 이미지를 출력한다.

## Ⅲ. 하드웨어 합성 결과

구현된 하드웨어는 Verilog HDL[10]로 코딩하였으며, Xilinx 사의 xczu7ev-ffvc1156 목표 보드[11]를 이용해 Field Programmable Gate Array 환경에서 구현하였다. FMSG와 제안하는 VMSG의 성능을 비교하기 위하여, 표 1에서 1H delay 신호를 생성하는 모듈(1H)부터 1H~4H delay를 생성하는 모듈(Up to 4H)까지 총 4가지 모듈에 대한 하드웨어 합성 결과를 나타냈다. 1H와 Up to 4H 모듈의 Register의 수와 LUT의 수를 비교하였을 때, FMSG에서는 사용하는 Register의 수가 52에서 141로 증가하였고 LUT의 수도 86에서 335로 증가하였다. 마찬가지로, VMSG에서도 사용하는 Register의 수가 59에서 89로 증가하였고 LUT의 수도 80에서 172로 증가하였다. 필터링 윈도우 크기가 2×2의 경우(1H 사용)에는 FMSG와 VMSG의 하드웨어 크기는 거의 비슷하지만, 윈도우 크기가 5×5로 증가한 경우(Up to 4H 사용)에는 FMSG가 VMSG에 비해 Register의 수는 약 158%(=141/89) 더 필요로 하며, LUT의 수는 약 195%(=335/172) 더 사용하는 것을 볼 수 있다. 따라서 사용하는 윈도우 크기에 비례하여 FMSG를 사용하는 경우는 하드웨어 크기가 선형적으로 더욱 증가하는 것을 알 수 있다. 이 결과를 그림 11에 그래프로 나타내었는데, FMSG의 Register와 LUT 하드웨어 자원 증가분의 기울기는 VMSG의 기울기보다도 더욱 크다는 것을 볼 수 있다. 따라서 사용하는 윈도우 크기가 증가함에 따라 필요한 하드웨어 자원의 상승 폭이 FMSG가 VMSG 보다 더욱 큰 것을 확인하였다.

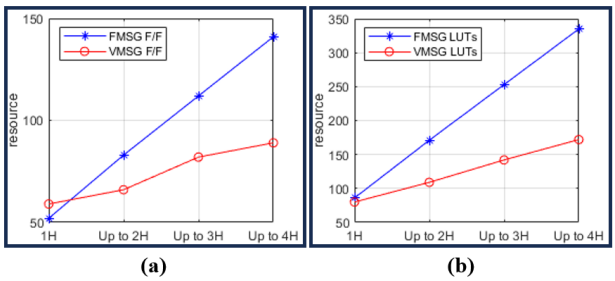


Fig. 11. Compare hardware resource of sync generator. (a) Register and (b) Look-up Tables.
그림 11. 동기 발생기 하드웨어 자원 비교 (a) 레지스터, (b) 룩업 테이블

Table 1. Vivado synthesis result of sync generator.
표 1. Vivado를 이용한 동기 발생기 합성 결과

(a) Fixed-Mode Sync Generator (image size: 256×256).

| Xilinx Vivado 2023.1 | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Device | xczu7ev-ffvc1156-2 | | | | | | | | |
| Logic Utilization | Avaliable | 1H | | Up to 2H | | Up to 3H | | Up to 4H | |
| | | Used | Util. | Used | Util. | Used | Util. | Used | Util. |
| CLB Register(#) | 460,800 | 52 | 0.01% | 83 | 0.02% | 112 | 0.02% | 141 | 0.03% |
| CLB LUTs(#) | 230,400 | 86 | 0.04% | 171 | 0.07% | 253 | 0.11% | 335 | 0.15% |
| Minimum period(ns) | - | 1.431 | | 1.446 | | 1.432 | | 1.484 | |
| Maximum Freq.(MHz) | - | 698.812 | | 691.562 | | 698.324 | | 673.854 | |

(b) Variable-Mode Sync Generator (image size: 256×256).

| Xilinx Vivado 2023.1 | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Device | xczu7ev-ffvc1156-2 | | | | | | | | |
| Logic Utilization | Avaliable | 1H | | Up to 2H (2×2) | | Up to 3H | | Up to 4H | |
| | | Used | Util. | Used | Util. | Used | Util. | Used | Util. |
| CLB Register(#) | 460,800 | 59 | 0.01% | 66 | 0.01% | 82 | 0.02% | 89 | 0.02% |
| CLB LUTs(#) | 230,400 | 80 | 0.04% | 109 | 0.05% | 142 | 0.06% | 172 | 0.07% |
| Minimum period(ns) | - | 1.432 | | 1.433 | | 1.452 | | 1.485 | |
| Maximum Freq.(MHz) | - | 698.324 | | 697.836 | | 688.705 | | 673.400 | |

Table 2. ivado synthesis result of LPF filtering.
표 2. Vivado를 이용한 LPF 필터링 하드웨어 합성 결과

| Xilinx Vivado 2023.1 | | | |
|---|---|---|---|
| Device | xczu7ev-ffvc1156-2 | | |
| Logic Utilization | Avaliable | Use FMSG | Use VMSG |
| | | Used(Util.) | Used(Util.) |
| CLB Register(#) | 460,800 | 383(0.08%) | 377(0.08%) |
| CLB LUTs(#) | 230,400 | 410(0.18%) | 330(0.14%) |
| Block RAMs(#) | 312 | 1(0.32%) | 1(0.32%) |
| Min. period(ns) | - | 1.590 | 1.588 |
| Max. Freq.(MHz) | - | 628.930 | 629.722 |

* The EDA tool was supported by the IC Design Education Center (IDEC), Korea.

표 2는 그림 10에 해당하는 3×3 LPF를 이용한 이미지 필터링을 수행하는 하드웨어에 FMSG를 사용한 경우와 VMSG를 사용한 경우에 대한 하드웨어 합성 결과를 보여준다. 동작 주파수는 약 630MHz로 유사하지만, 하드웨어 자원은 VMSG의 경우가 FMSG를 사용하는 경우보다 Register는 6 그리고 LUT는 80의 자원을 덜 사용한다. 만약 5×5(Up to 4H 경우)를 사용하면 사용 메모리는 두 경우 동일한 2가 되지만, VMSG는 표 1에서 Register는 52 그리고 LUT는 163의 자원을 덜 사용하는 것을 볼 수 있다. 이는 VMSG를 사용함으로써 얻을 수 있는 장점이라고 할 수 있다.

## Ⅳ. 결론

본 논문에서는 이미지처리에서 디지털 필터 구현을 위한 가변 가능한 VMSG를 제안하였다. 기존 FMSG와 다르게 이미지의 크기 정보를 입력으로 받는 방식이 아닌, auto detection 기능을 추가하여 이미지의 width, blank 정보를 자동으로 검출하기 때문에 유동적으로 사용할 수 있다는 장점이 있다. 또한 더 많은 지연 신호가 필요할 때, 모듈을 연결하는 방식인 FMSG와 달리 FSM을 추가해주는 방식을 사용하였다. 그 결과 Xilinx Vivado를 이용하여 하드웨어를 구현하였을 때 사용하는 윈도우의 크기가 증가할수록 성능이 우수함을 검증하였다. 마지막으로 이미지처리 하드웨어에 FMSG와 VMSG를 적용한 후 하드웨어를 구현하였을 때도 동작 주파수는 유사하지만, 논문에서 제안하는 VMSG를 적용한 경우가 요구하는 하드웨어 자원은 더 적은 것을 확인하였다. 향후 VMSG를 활용하여 up/down scaling 이미지처리 하드웨어를 구현할 예정이다.

## References

[1] H.-S. Park, “Single SRAM Based Line Memory System for Providing Column Vectors of 2D Filtering,” *Journal of KIIT,* vol.12, no.4, pp.137-143,

2014. DOI: 10.14801/kiite.2014.12.4.137
[2] M. Azizabadi and A. Behrad, "Design and VLSI implementation of new hardware architectures for image filtering," *2013 8th Iranian Conference on Machine Vision and Image Processing,* 2013. DOI: 10.1109/IranianMVIP.2013.6779960
[3] Ngo. D, S. Lee, U.-J. Kang, T. M. Ngo, G.-D. Lee and B. Kang, "Adapting a Dehazing System to Haze Conditions by Piece-Wisely Linearizing a Depth Estimator," *Sensors*, vol.22, no.5, pp.1957-1981, 2022. DOI: 10.3390/s22051957
[4] U.-J. Kang and B. Kang, "Hardware Implementation of Fog Feature Based on Coefficient of Variation Using Normalization," *Journal of the lorea Institute of Information and Communication Enginerring,* vol.25, no.6, pp.819-824, 2021.
[5] S. Lee and B. Kang, "A 4K-Capable Hardware Accelerator of Haze Removal Algorithm using Haze-relevant Features," *Journal of information and communication convergence engneering,* vol.20, no.3, pp. 212-218, 2022.
DOI: 10.56977/jicce.2022.20.3.212
[6] A. Sengupta and M. Rathor, "Obfuscated hardware accelerators for image processing filters-Application specific and functionally reconfigurable processors," *IEEE Transactions on Consumer Electronics,* vol.66, no.4, 2020. DOI: 10.1109/TCE.2020.3027760
[7] Shambhu Sharan and S.P. Tiwari, "Products of Mealy-type Rough Finite State Machine," *2012 National Conference on Computing and Communication Systems,* 2012.
DOI: 10.1109/NCCCS.2012.6412988
[8] W. Jang, J. Kim, S. Lee, B. Kwak and B. Kang, "Hardware Implementation of 2D-CZP Pattern for System verification," *2007 Autumn Annual Conference of IEIE,* vol.30, no.2, pp.597-598, 2007.
[9] Mentor Graphics, "ModelSim® User's Manual," http://microsemi.com/document-portal/doc_view/131619-modelsim-user
[10] *IEEE Std 1364-2005 (Revision of IEEE Std 1374-2001),* "IEEE Standard for Verilog Hardware Description Language," 2006.
DOI: 10.1109/IEEESTD.2006.99495
[11] Xilinx, "ZCU106 Board User Guide," https://docs.xilinx.com/v/u/en-US/ug1244-zcu106-eval-bd

## BIOGRAPHY

**Semin Jung** (Member)

2023 : BS degree in Electronics Engineering, Dong-A University.
2023~present : MS degree course in Electronics Engineering, Dong-A University.

**Si-Yeon Han** (Member)

2023 : BS degree in Electronics Engineering, Dong-A University.
2023~present : MS degree course in Electronics Engineering, Dong-A University.

**Bongsoon Kang** (Member)

1985 : BS degree in Electronics Engineering, Yonsei University.
1987 : MS degree in Electrical Engineering, University of Pennsylvania.
1990 : Ph.D degree in Electrical Engineering, Drexel University.
1989~1999 : Senior Staff Researcher, Samsung Electronics.
1999~present : Prof. of Dept. Electronics Engineering, Dong-A University.