|  |
| --- |
| **기말과제 보고서:**  **RC카 원격 조종 컨트롤러 설계 및**  **충돌감지 시스템** |



|  |  |
| --- | --- |
| **제출자** | 21700384 송태웅 |
| 21700794 홍준형 |
| **날짜** | 2022. 12. 18 |
| **수강과목** | 디지털시스템설계 |
| **담당교수** | 이강 |

목차

[1. 서론 3](#_Toc122287233)

[A. 실험 목표 3](#_Toc122287234)

[B. 하드웨어 구성 3](#_Toc122287235)

[C. Preliminary 4](#_Toc122287236)

[2. 본론 9](#_Toc122287237)

[A. 설계 9](#_Toc122287238)

[B. 시뮬레이션 11](#_Toc122287239)

[C. 합성 결과 14](#_Toc122287240)

[3. 결론 및 토의 15](#_Toc122287241)

[A. 분석 및 완성도 15](#_Toc122287242)

[B. 개선 및 추가 연구 내용 15](#_Toc122287243)

[C. 응용 가능성 16](#_Toc122287244)

[4. 부록 17](#_Toc122287245)

[A. Reference 17](#_Toc122287246)

[B. Demo 17](#_Toc122287247)

[C. 조원별 역할 17](#_Toc122287248)

[D. 개인 소감문 17](#_Toc122287249)

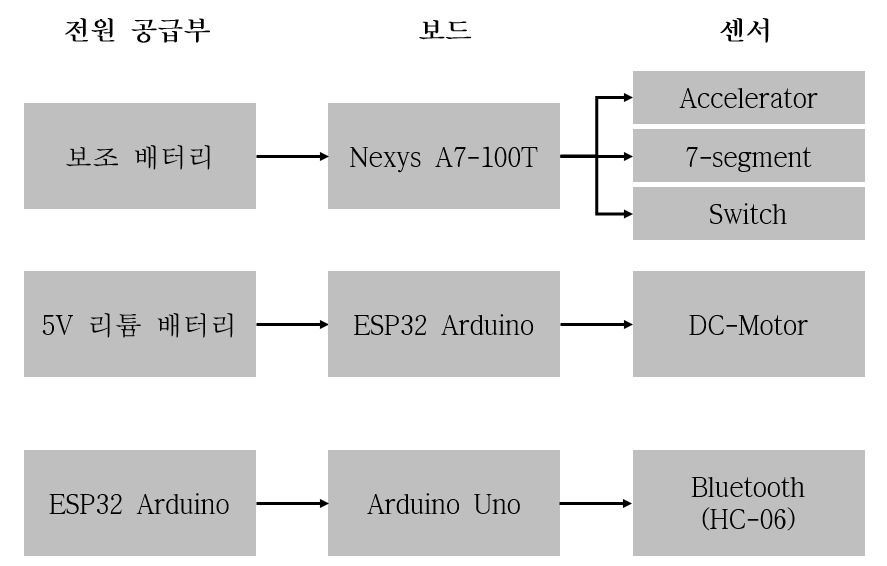
[E. 소스코드 깃허브 17](#_Toc122287250)

# **서론**

## 실험 목표

1. 블루투스 모듈 및 RC Car 키트를 통해 RC Car 원격 조종 컨트롤러를 설계 및 구현한다.
2. Accelerator를 이용하여 충돌감지 및 긴급제동 시스템을 구현한다.

## 하드웨어 구성



**그림 1. 하드웨어 연결도**

|  |  |
| --- | --- |
|  |  |
|  |  |

**그림 2. 차량 사진**

**상면(좌상), 후면(우상), 전면(좌하), 측면(우하)**

## Preliminary

* + 1. Flash Memory Programming

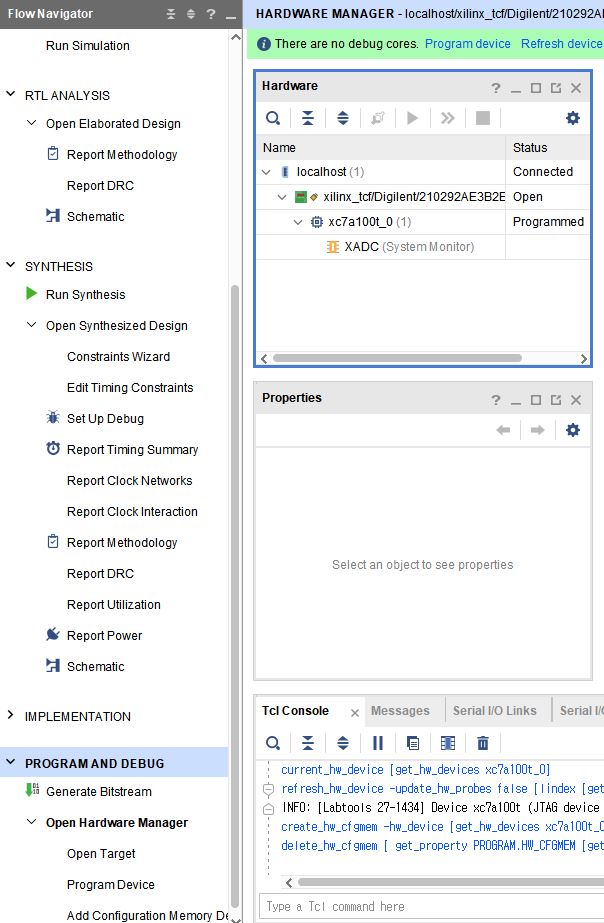
Xilinx FPGA의 연결 스위치에 대한 프로그래밍 구조에는 SRAM(Static RAM) 방식이 있다. SRAM 프로그래밍 구조의 특징은 연결점을 제어하기 위한 배열 정보가 SRAM 셀에 저장되어 있다는 것이다. 이는 보드 전원이 OFF되면 프로그래밍 정보가 소실되는 휘발성 기억 소자이다. 따라서, RC Car 원격 조종을 위해서는 FPGA에 심겨진 코드 정보가 비휘발성을 가질 필요가 있다.

FPGA의 코드 정보가 비휘발성을 가지기 위해서 FPGA에 내장된 Flash Memory에 프로그래밍하였다. Flash Memory는 비휘발성으로 전원이 OFF되어도 프로그래밍 정보가 잔존하여 있기 때문에 보조 배터리를 이용하여 RC Car를 작동시키기에 적합하다.

SRAM 프로그래밍에서 Flash Memory 프로그래밍으로 전환하는 방법은 Bitstream 파일을 mcs 파일로 변환하고, 이를 Flash Memory에 저장하는 것이다.

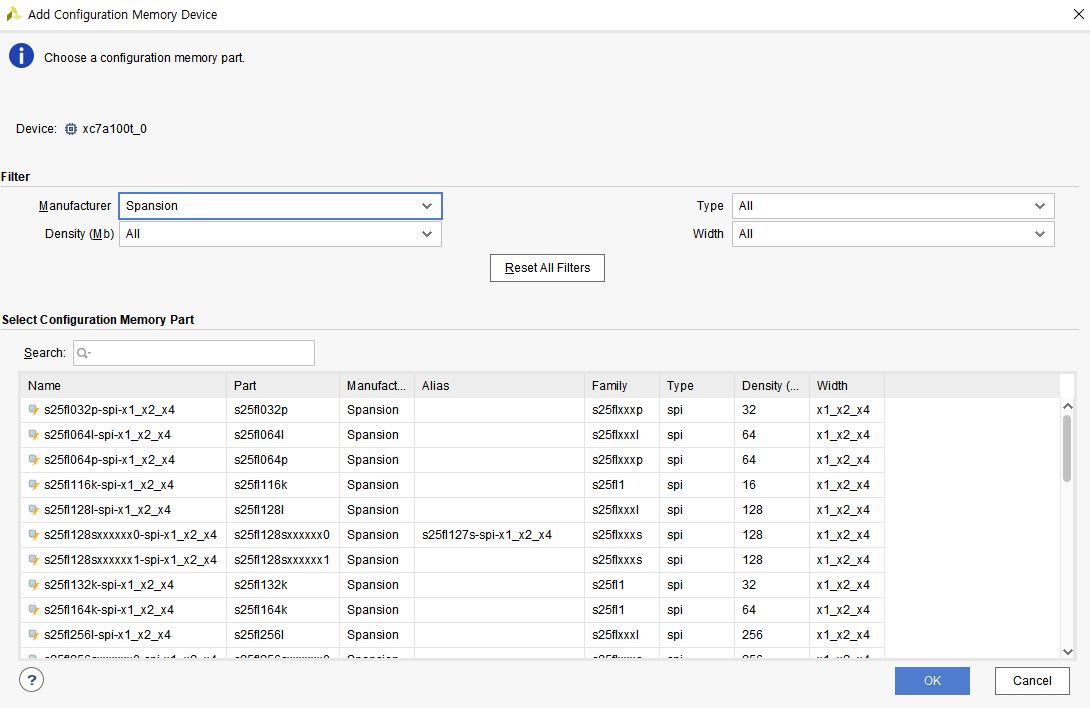
* + 실습
    - Add Configuration Memory Device 선택
      * s25FL128s (128 Mb) SPI Flash Memory (Spansion)

1. Add Configuration Memory Device

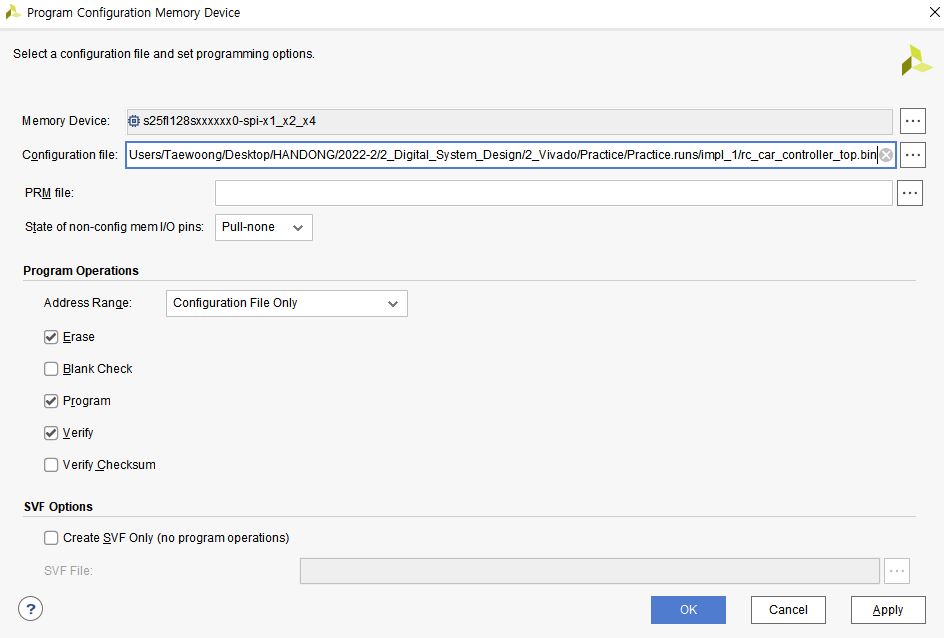


1. Select *Manufacturer* as *Spanion*

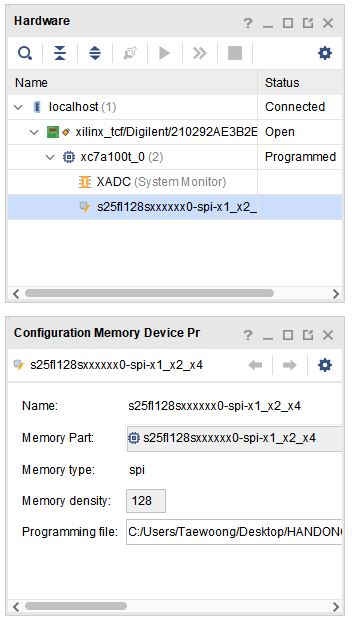
Select Configuration Memory Part and Click OK



1. Select configuration file as *your own Top module binary file*



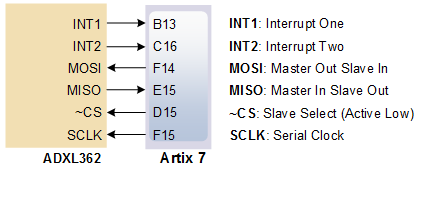
1. Flash memory file is created



* + 1. Accelerator

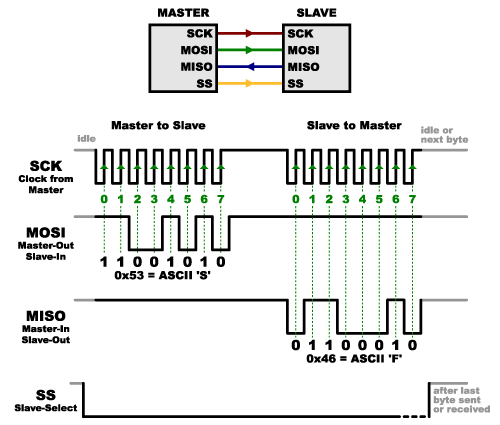
Nexys A7은 ADXL362 accelerator가 포함되어 있다. ADXL362는 입력 신호를 모든 데이터 속도에서 센서의 전체 대역폭을 샘플링하여 12비트 출력 해상도를 제공한다. FPGA는 SPI 인터페이스를 통해 ADXL362와 통신할 수 있으며, SPI 통신은 UART 통신보다 빠른 통신을 지원한다. ADXL362는 측정 모드에 있는 동안 X 데이터, Y 데이터, Z 데이터 레지스터에 가속도 데이터를 지속적으로 측정하고 저장한다. FPGA와 accelerator 사이의 인터페이스는 **그림 3**과 같다.

ADXL362는 **그림 3**에서 SPI 통신 방식을 이용하여 Slave Device로 작동한다.



**그림 3. Accelerator 인터페이스**

SPI 통신의 데이터 수신 동작은 **그림 4**와 같다. SPI는 Master만 클럭 분주기를 생성하며, Master가 주는 Clock signal이 rising edge 혹은 falling edge일 때, MOSI와 MISO에서 데이터 값을 순차적으로 읽어낸다. Master에서 데이터를 보낼 때 1비트 값인 ~CS(Slave Select) 값이 LOW인 경우 통신을 할 수 있다.



**그림 4. SPI 데이터 수신**

# **본론**

## 설계

* + 1. Controller

본 과제는 bluetooth를 이용한 키보드 입력으로 상태를 전환하기 때문에 Finite State Machine으로 controller를 구현하는 것이 무의미하다. 따라서 5개의 키보드 입력에 따라 상태 변화는 아래와 같다.

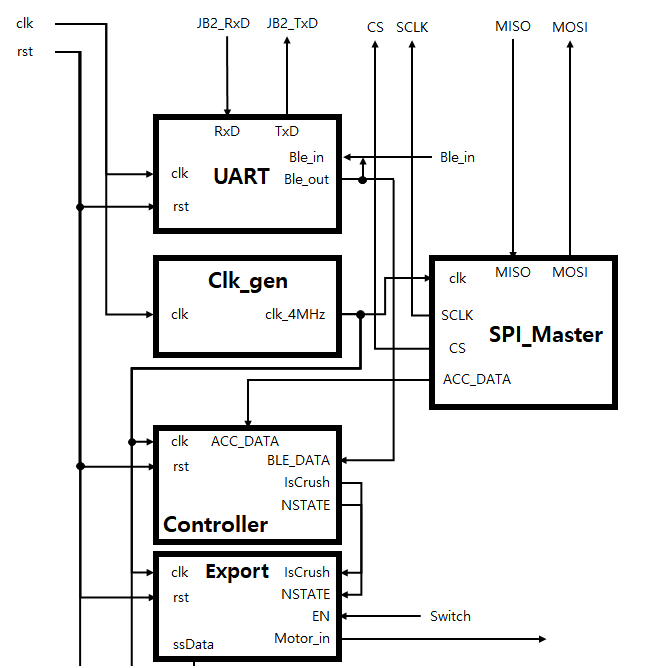
\*EN: Enable 스위치 상태

\*InputKey: 키보드 입력

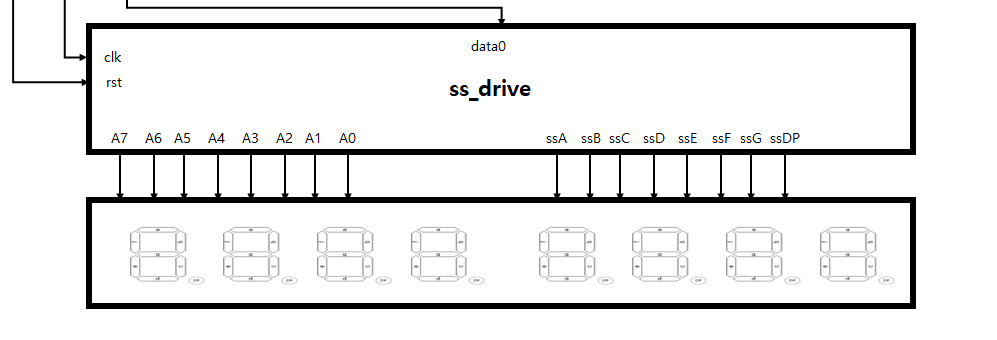
\*IsCrush: 충돌감지 여부 판단 bit

|  |  |
| --- | --- |
| 입력 | 상태 |
| **EN=0** or  **(EN=1 and IsCrush=1)** or  **(EN=1 and InputKey=’t’)** | STOP |
| **EN=1 and InputKey=’q’** | FORWARD |
| **(EN=1 and InputKey=’w’)** or  **(EN=1 and IsCrush=1 and InputKey=’w’)** | BACKWARD |
| **EN=1 and InputKey=’e’** | LEFT |
| **EN=1 and InputKey=’r’** | RIGHT |

* + 1. 알고리즘 블록연결도



**RC car**



## 시뮬레이션

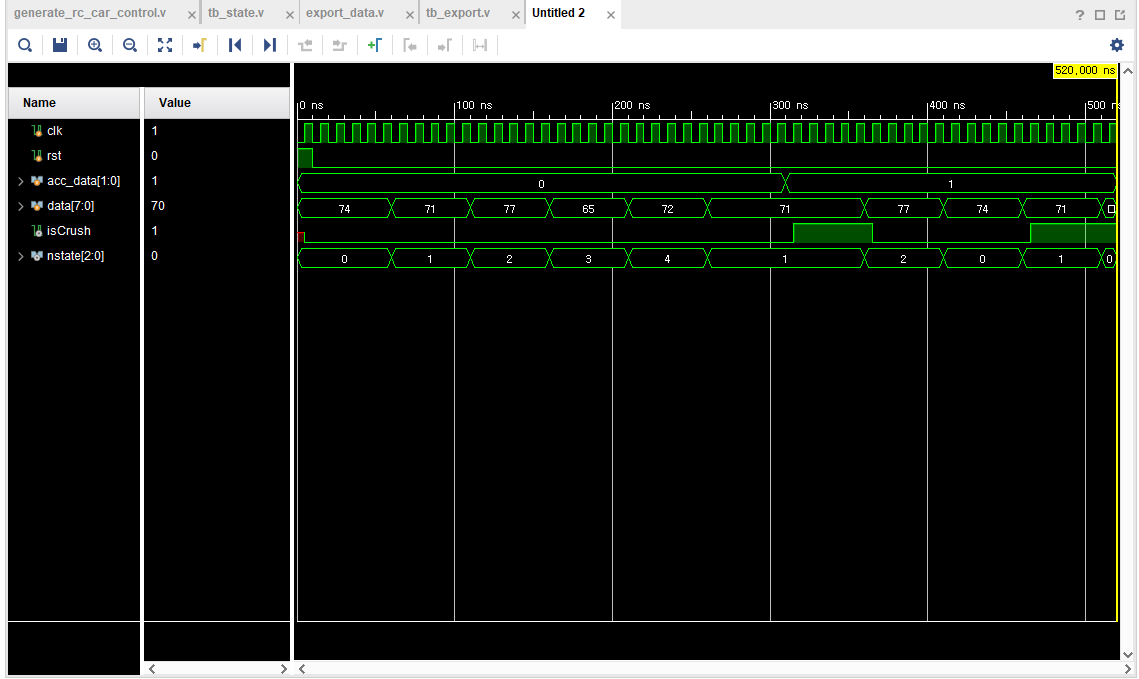
* + 1. 상태

키보드 입력 및 충돌 감지로 인한 상태변화를 시뮬레이션하였다. 시뮬레이션 결과 해석을 위해 필요한 정보를 아래 표로 정리하였다.

|  |  |
| --- | --- |
| 키보드 입력 | data [7:0] |
| ‘q’ | ‘h71 |
| ‘w’ | ‘h77 |
| ‘e’ | ‘h65 |
| ‘r’ | ‘h72 |
| ‘t’ | ‘h74 |

|  |  |
| --- | --- |
| 상태 | nstate [2:0] |
| FORWARD | 3’b001 |
| BACKWARD | 3’b010 |
| LEFT | 3’b011 |
| RIGHT | 3’b100 |
| STOP | 3’b000 |

|  |  |
| --- | --- |
| 신호 | 의미 |
| isCrush | 충돌감지 신호 |
| acc\_data | 충돌감지 인식 데이터 |

****

**그림 5. 시뮬레이션 결과 (1)**

위 그림은 Bluetooth 키보드 입력 값에 대응하는 8 비트 data 값에 따라 3 비트 nstate 값이 변화하는 것에 대한 시뮬레이션 결과이다. ‘t’를 의미하는 ‘h74는 3’b000의 STOP 값이 출력되는 것을 확인할 수 있다. 이와 같이 위 표에 해당되는 ‘q’, ‘w’, ‘e’, ‘r’를 차례대로 입력했을 때, FORWARD, BACKWARD, LEFT, RIGHT가 순서대로 출력되는 것을 확인할 수 있다. 가속도 데이터를 의미하는 acc\_data가 1인 동시에 data가 직진 상태를 의미하는 h’71 일 때, 충돌 상태를 의미하는 isCrush 값이 HIGH(1) 값을 가지는 것을 확인할 수 있다. isCrush 값은 직진 상태인 상태에만 값이 변화하기 때문에 data 값으로 ‘w’, ‘e’를 입력하였을 때, isCrush의 값이 LOW(0)로 유지되는 것을 확인할 수 있다.

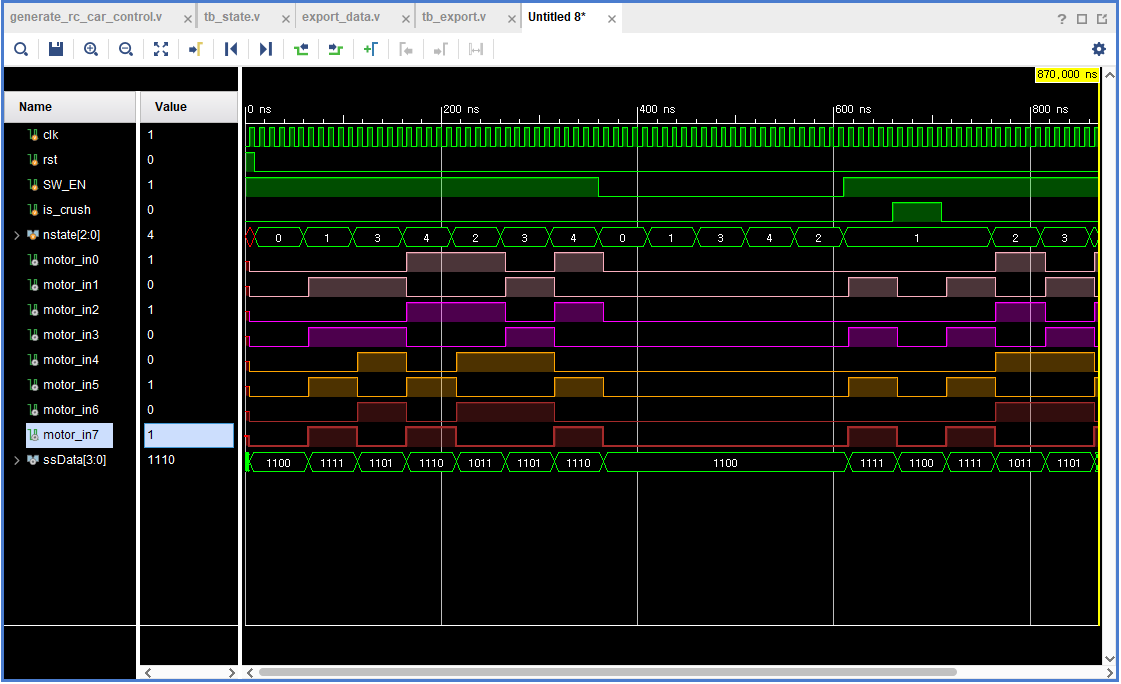
* + 1. 모터 출력

상태값에 대응하는 사륜 구동 제어를 시뮬레이션하였다. 시뮬레이션 결과 해석을 위해 필요한 정보를 아래 표로 정리하였다. nstate 값은 **B. 시뮬레이션 i. 상태**에서 output 값인 nstate와 동일하다.

|  |  |
| --- | --- |
| 신호 | 의미 |
| SW\_EN | Enable 스위치 |
| is\_crush | 충돌감지 신호 |
| accData | 충돌감지 인식 데이터 |
| ssData | 7 Segment 데이터 |

|  |  |
| --- | --- |
| 신호 | 의미 |
| motor\_in0 | 우측 전방 바퀴 후진 신호 |
| motor\_in1 | 우측 전방 바퀴 전진 신호 |
| motor\_in2 | 우측 후방 바퀴 후진 신호 |
| motor\_in3 | 우측 후방 바퀴 전진 신호 |
| motor\_in4 | 좌측 전방 바퀴 후진 신호 |
| motor\_in5 | 좌측 전방 바퀴 전진 신호 |
| motor\_in6 | 좌측 후방 바퀴 후진 신호 |
| motor\_in7 | 좌측 후방 바퀴 전진 신호 |

|  |  |
| --- | --- |
| 신호(이진수) | 의미 |
| ss\_S(4’b1100) | 7 Segment ‘s’ 출력 |
| ss\_F(4’b1111) | 7 Segment ‘f’ 출력 |
| ss\_B(4’b1011) | 7 Segment ‘b’ 출력 |
| ss\_L(4’b1101) | 7 Segment ‘l’ 출력 |
| ss\_R(4’b1110) | 7 Segment ‘r’ 출력 |

****

**그림 6. 시뮬레이션 결과 (2)**

**그림 6**은 3 비트 nstate 값에 대응하는 차량 구동 제어에 대한 시뮬레이션 결과이다. SW\_EN 값이 HIGH(1) 값을 가지고, nstate 값이 직진 상태를 의미하는 3’b001 일 때, 전진 신호를 의미하는 motor\_in1, motor\_in3, motor\_in5, motor\_in7이 HIGH(1) 값을 가지는 것을 확인할 수 있다. 이와 달리, nstate 값이 후진 상태를 의미하는 3’b010일 때, 후진 신호를 의미하는 motor\_in0, motor\_in2, motor\_in4, motor\_in6이 HIGH(1) 값을 가지는 것을 확인할 수 있다.

nstate 값이 좌회전 상태인 3’011 일 때, 우측 바퀴 후진 신호인 motor\_in0, motor\_in2과 좌측 바퀴 전진 신호인 motor\_in5, motor\_in7 LOW(0) 값, 우측 바퀴 전진 신호인 motor\_in1, motor\_in3과 좌측 바퀴 후진 신호인 motor\_in4, motor\_in6은 HIGH(1) 값을 가지는 것을 확인할 수 있다. 결국, 좌측 바퀴의 후진 상태와 우측 바퀴의 전진 상태를 유지하여 좌회전하는 것을 확인할 수 있다. 우회전의 경우, 좌회전과 반대로 좌측 전진 신호가 HIGH(1), 후진 신호가 LOW(0)로 우측 전진 신호가 LOW(0), 후진 신호가 HIGH(1) 값을 가지는 것을 확인할 수 있다.

SW\_EN이 LOW(0)일 때, 어떠한 값을 입력하여도 작동하지 않는 것을 확인할 수 있다. 또한, is\_crush 값이 HIGH(1)인 경우, 직진 상태에서 정지 상태로 넘어가는 것을 확인할 수 있다.

ssData는 nstate 값에 따라 위 표에 해당되는 값으로 변화하여 7 segment 출력으로 이어진다. nstate 값이 차례대로 1(3’b001), 3(3’b011), 4(3’b100), 2(3’b010)일 때, ssData 값이 1111, 1101, 1110, 1011임을 확인할 수 있다. 이는 7 Segment에서 ‘f’, ‘l’, ‘r’, ‘b’로 출력된다.

## 합성 결과

* + 1. Slice Logic

테이블이(가) 표시된 사진

자동 생성된 설명

**그림 7. Slice Logic 결과**

* + 1. I/O와 GT Specific

테이블이(가) 표시된 사진

자동 생성된 설명

**그림 8. I/O와 GT Specific 결과**

# **결론 및 토의**

## 분석 및 완성도

* + - * 1. RC Car 원격 조종 컨트롤러 설계 (100/100)

테스트 벤치 결과 분석 및 실제 구동을 확인한 결과, UART통신을 이용한 키보드 데이터 전달 및 상태 변환에 아무런 문제없이 작동하는 것을 확인할 수 있다.

* + - * 1. 충돌감지 및 긴급 제동 시스템 설계 (50/100)

본래의 목적은 3축 가속도 센서의 각 축을 고려하여, 이동방향 속도 변화량에 대한 문턱값(threshold)으로 긴급제동 명령을 제어하는 것이었다. 하지만, 이를 구현하기 위한 ALU 설계에 개인적 한계가 존재하였으며 불안정성이 높은 간단한 방법을 통해 제어하였다.

현 구현 방법은 차량이 전진하는 상황에서 각 축의 5bit 가속도 데이터(총 15-bit 데이터) 중 전면 충돌 시 민감하게 반응하는 데이터를 관찰 및 선정하였으며, 해당 bit의 변화에 따라 충돌감지 신호를 제어하였다.

이의 문제점으로는 충돌상황이 아닌 노면의 작은 영향에도 민감하게 반응하여 긴급제동하는 경우가 존재하였다.

## 개선 및 추가 연구 내용

* + - * 1. 강인한 충돌감지 및 긴급제동 시스템 구현

위에서 언급한 것과 동일하게 accelerator의 각 축의 데이터를 연산 및 문턱값 설정을 통한 더욱 강인한 시스템 설계가 필요하며, 전진 상황뿐만 아니라, 좌우 및 전후진 시 충돌감지가 가능한 범용적 시스템으로 개선이 가능하다.

* + - * 1. 조이스틱을 이용한 차량 조종

차량 조종을 키보드를 이용한 원격조종을 하는 것보다 조이스틱을 이용한 차량 조종이 더욱 현실성이 있다. 이를 위해 조이스틱을 연결한 보드와 조종 차량 간의 데이터 통신이 필요하다.

* + - * 1. 각종 센서를 이용한 자율주행 시스템 연구

현재는 accelerator를 이용한 충돌 감지 및 긴급제동 시스템을 구현하였으나, IR센서를 이용한 Line tracing이나 초음파 센서를 이용한 충돌 방지 시스템, GPS 혹은 라이다 센서를 이용한 이동경로계획 및 자율주행 시스템 연구가 가능하다.

## 응용 가능성

본 과제의 적용가능분야로는 무선 차량 조종 장난감이 있다. 또한 아이가 차량에 탑승하고 부모가 차량을 조종하는 경우, 아이의 안전을 위한 충돌감지시스템으로 적용가능하다. 추가적으로 위에서 언급한 개선 및 추가 연구가 진행된다면, 더욱 안전하고, 흥미도가 높은 장난감으로 발전가능하며 음식점 내 자율서빙로봇 등에도 적용가능하다.

# **부록**

## Reference

* + - * 1. [Nexys A7 3-Axis Accelerometer SPI 소스코드](https://github.com/FPGADude/Digital-Design/tree/main/FPGA%20Projects/Nexys%20A7%203-Axis%20Accelerometer%20SPI)
        2. [Vivado Flash Programming 소스 영상](https://www.youtube.com/watch?v=eVMBHD2S6v4)
        3. [Flash Memory와 SRAM Programming 개념](https://cms3.koreatech.ac.kr/sites/yjjang/down/dsys08/M04_fpga.pdf)
        4. [Nexyz A7 Reference Manual](https://digilent.com/reference/programmable-logic/nexys-a7/reference-manual)

## Demo ([링크](https://www.youtube.com/shorts/jeqbFufcGZM))

## 조원별 역할

|  |  |
| --- | --- |
| 조원 | 역할 |
| 송태웅 | * + - * 1. 하드웨어 설계         2. 알고리즘 설계         3. 보고서 작성(본론B. 결론) |
| 홍준형 | 테스트 벤치 코드 작성 및 결과 분석  보고서 작성(서론, 본론A, 부록) |

## 개인 소감문

|  |  |
| --- | --- |
| 조원 | 소감문 |
| 송태웅 | FPGA 보드 및 Verilog 코딩에 대한 큰 흥미가 생겼으며 이를 이용하여 다양한 어플리케이션을 하고 싶은 마음이 생겼습니다. FPGA 보드에 적합한 효율적인 algorithm structure와 논리 회로에 대한 높은 이해도를 함양한 엔지니어가 되고 싶은 욕심이 생겼습니다. |
| 홍준형 | Verilog의 이해도가 낮았는데 조별과제와 기말과제를 같이 진행하면서 팀원의 도움으로 Verilog에 대해 조금 알게 되었고, 이를 통해 RC Car를 원격으로 조종하면서 재미를 느꼈습니다. 뒤늦게 재미를 찾게 되어 아쉽지만 앞으로 학업에 정진할 수 있도록 동기 부여가 된 거 같아 뜻깊은 시간이었습니다. |

## 소스코드 깃허브 ([링크](https://github.com/xodnde3123/DigitalSystemDesign/tree/main/FinalProject))