**Lab1.**

# **Homework1. 5-tap fir symmetric filter**

**Verilog Code / 주석**

텍스트, 스크린샷, 문서, 폰트이(가) 표시된 사진

자동 생성된 설명

<5-tap FIR symmetric filter>

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

<1~5번 4-bit D F/F>

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

<6번 10-bit D F/F>

**Synthesis Report**

**텍스트, 폰트, 라인, 번호이(가) 표시된 사진

자동 생성된 설명**

**<Timing>**

**텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명**

**텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명**

**<Utilization>**

**텍스트, 스크린샷, 소프트웨어, 번호이(가) 표시된 사진

자동 생성된 설명**

**<Power>**

**FPGA의 전력 소모와 온도 추정치가 높게 도출됨을 알 수 있다. 칩 내 총 전력은 10.384W이고, 이로 인해 Junction temperature가 125°C로 상승한 것으로 볼 수 있다. Thermal margin이 -59.8°C이므로, 약 60°C정도 더 낮아야 안정적이다. 특히, I/O에서 발생하는 전력 소모가 약 95%로 매우 많음을 알 수 있다. 이는 주된 발열 원인이고, 이를 해결하기 위한 방법을 모색해야한다.**

**먼저, 사용하지 않는 I/O pin을 비활성화하고, I/O pin에서의 pull-up, pull-down 저항을 비활성화하면 전력 소모를 줄일 수 있다. 또한, 필요하지 않은 logic block에 clk을 차단하여 전력 소모를 줄이는 최소 Clock gating 방법이 있거나, FPGA에 방열판을 추가하여 열 반산을 통해 junction 온도를 낮추는 등의 방법이 있다.**

**도표, 평면도, 기술 도면, 라인이(가) 표시된 사진

자동 생성된 설명**

**<Synthesis schematic>**

**도표, 평면도, 라인, 개략도이(가) 표시된 사진

자동 생성된 설명**

**<RTL(Register Transfer Logic) Level Schematic>**

**Test Bench Code**

텍스트, 스크린샷, 폰트, 문서이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

**Simulation Result**

스크린샷, 라인, 그래프이(가) 표시된 사진

자동 생성된 설명

<Simulation>

스크린샷, 멀티미디어 소프트웨어, 소프트웨어, 그래픽 소프트웨어이(가) 표시된 사진

자동 생성된 설명

<Post-synthesis functional simulation>

Post-synthesis functional simulation은 FIR filter의 din 입력 신호가 다양한 패턴으로 변화할 때, dout 출력 신호가 어떻게 반응하는지 합성 후 회로의 기능적 동작을 검증하는 과정이다. 이때, 경로 지연이나 게이트 지연 등은 고려하지않고, 논리적 기능만을 확인한다는 점에서 behavioral simulation과 차이가 있다. 이때, 정상적인 기능을 수행함을 파형을 통해 확인할 수 있다. 자세한 설명은 고찰에서 다룰 것이다.

스크린샷, 텍스트, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

<Post-synthesis timing simulation>

스크린샷, 도표, 사각형이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명  
din의 변화시점은 약 110ns이고, dout의 반응 시점은 약 114ns로 약 4ns 정도의 delay가 발생한 것으로 판단하였다. 이는 FIR filter의 각각의 경로 지연이 누적된 결과이다. 합성된 회로에서는 각각의 논리 게이트와 배선에서의 지연이 더해지기에 순수 RTL simulation에 비해 timing delay가 증가한다.

Gate delay에 대해 자세히 설명하자면, 합성된 FIR filter는 여러 논리 게이트와 Mux, F/F으로 구성되어 있다. 각 논리 소자는 신호를 전달할 때 약간의 delay가 발생하게 된다. 특히, FIR filter는 여러 tap을 거쳐야 하기에 경로 delay가 발생하게 되는 것이다.

경로 지연은 신호가 각 논리 소자 사이를 통과하거나, 배선을 거쳐야하는 경우에 발생할 수 있다. 각각의 dealy 요소(DFF)이나 곱셈, 합산 연산들이 연결되는 경로 길이가 길어질수록 경로 지연이 증가한다.

|  |
| --- |
| **Discussion** |
| . - **Verilog Coding을 시작하기 전 작성한 Block Diagram**    <초반에 작성한 5-tap FIR symmetric filter>    <연산 법칙을 통해 delay를 줄인 실제 설계에서의 block diagram>  Finite Impulse Response Filter는 디지털 신호 처리를 위한 필터의 한 종류이다. 5-tap FIR Filter는 5개의 delay 요소(tap)를 가지고 있으며, 각 tap은 입력 신호의 delayed 버전을 저장한다. Symmetric filter라는 것은 filter의 계수가 대칭적인 패턴을 가진다는 것이고, 이를 통해 계산 효율을 높일 수 있고, 특정 주파수 응답을 요구하는 동작에서 유리하다. 특히, Low Pass Filtering 에서 유리하다. 문제에서 사용한 filter coefficient는 coeff0=1, coeff1=3, coeff2=6이다. Filter가 대칭적이므로 계수가 대칭적이다. Tap1과 tap5는 coeff0을 사용하고, tap2와 tap4는 coeff1을 사용하고, tap3는 coeff2를 사용하였다. 출력 dout은 각 delayed된 입력값과 해당coeff를 곱한 결과들의 합이다. 이러한 구조를 통해 noise 제거와 signal smoothing을 수행할 수 있다.  **- 작성한 Verilog Module에 대한 설명**  1. DFF  D Flip-Flop을 구현한 모듈이다. 이는 clk 신호와 rst 신호에 따라 입력값인 d를 출력 q로 전달하거나 초기화하는 역할을 수행한다. 이 모듈은 FIR filter 모듈에서 tap으로 사용되고, 입력 신호 d를 1clk 주기씩 지연시키는 기능을 한다. Clk이 positive edge일 때 결정되고, rst가 1로 활성화된 경우에 입력 d를 q에 저장한다. Rst가 0으로 비활성화된 경우에는 clk의 posedge에서 출력 q는 0으로 초기화된다.  2. DFF\_10bit  10bit D Flip Flop을 구현한 모듈이다. FIR Filter의 마지막 단계에서 10bit 크기의 delay 요소로 사용되며, filter의 최종 출력값인 dout을 clk 신호에 맞춰 전달하는 역할을 수행한다. Clk의 posedge에서 동작하며, rst에 따라 입력값 d를 출력q로 전달하거나, 출력을 0으로 초기화한다. Rst가 1로 활성화된 경우, 입력 d값이 출력 q에 그대로 전달된다. Rst가 0으로 비활성화된 경우에는 clk의 posedge에서 출력 q가 0으로 초기화된다. 즉, rst가 0인 동안에는 d값을 유지하지않고, q는 항상 0으로 설정된다. 이는 FIR filter의 최종 출력을 저장하거나 delay 시키기위한 목적으로 설계한 것이고, 필터링된 최종 합산 값인 sum을 clk에 맞춰 출력 dout에 전달하는 데 사용된다.  3. FIR\_filter  작성한 FIR\_Filter module은 5-tap FIR symmetric filter를 verilog로 구현한 module이다. Filter coefficient는 coeff0, coeff1, coeff2의 파라미터로 정의하였고, 이는 각각 signed 4bit이다. 입력 신호 din이 DFF 모듈을 통해 연속적으로 지연된다. Delay1부터 delay5까지 5개의 D Flip Flop이 사용되어 각 tap의 delay 요소를 구성한다. 각 tap의 지연 신호(delay1\_delay5)는 대응되는 계수 coeff0, coeff1, coeff2와 곱셈 연산을 수행한다. 곱셈 결과는 각각 signed 7bit 형식으로 저장된다. 즉, mul0~mul4에 저장된다. Filter의 대칭성을 이용하여 동일한 계수를 공유하는 요소를 이용해 코딩하였다. 각 곱셈 결과를 마지막에 더하여 최종 합인 sum을 계산하고, 이 합은 마지막 DFF\_10bit 모듈을 통해 dout으로 출력된다. 이는 signed 4bit와 4bit를 덧셈 연산하면 sign extension에 의해 5bit 출력이 나오고, signed 4bit와 4bit를 곱셈 연산하면 부호 1bit, 크기 6bit의 총 7bit의 결과가 도출됨을 이용하여 7bit+7bit+7bit+7bit+7bit의 형식으로 곱셈 연산을 먼저 진행하고, 그 결과들을 덧셈 연산하여 최종 10bit의 sum 입력을 받을 수 있는 10bit의 DFF module을 설계하였다.  7bit+7bit+7bit+7bit+7bit  =(8bit)+(8bit)+7bit  =(9bit)+7bit  =10bit  **- 작성한 Code가 어떤 동작을 하는지? 왜 그런 동작을 하는지?**  FIR filter는 입력 신호를 받아서 여러 단계의 delay 요소를 거치고, 각 지연된 신호에 특정 필터링 계수를 곱하고 합산하는 방식으로 동작한다. 이는 원래 주어진 FIR filter가 덧셈 연산 후에 곱셈 연산을 수행해야하는데, 이는 분배법칙과 교환법칙 등의 연산법칙에 의해 곱셈을 먼저 수행하고, 덧셈을 수행해도 된다는 점에 기인하여 설계 시에 coeff와 delay tap을 먼저 각각을 곱하고, 이를 모두 더하는 방식으로 설계하였다. 이를 통해 원래 방식대로 설계할 때 보다 delay 시간을 더 줄일 수 있고, 더 빠른 연산 속도를 얻을 수 있다.  먼저, 각각의 delay 요소들은 tap으로 사용되는데, delay1은 din이 첫번째 clk 동안 지연된 값, delay2는 두번째 clk 동안 지연된 값이고, 최종적으로 delay5는 5번째 clk 동안 지연된 값을 가진다. 이러한 delay 요소는 FIR Filter의 기본적인 구조로, 과거 입력 값을 현재 필터 연산에 사용하기 위해 필요하다. 각각의 delay 요소는 filter coefficient와 곱해진다. FIR Filter가 symmetric 하기 때문에 delay1과 delay5는 동일한 계수인 coeff0을 사용하고, delay2와 delay4는 동일한 계수 coeff1을, 중간 tap인 delay3는 coeff2를 사용한다. 이 곱셈은 filter의 주파수 응답을 정의하는 부분으로, 각 계수에 따라 특정 주파수 성분이 증폭되거나, 제거되는 필터링 동작을 수행한다.  각 곱셈 결과인 mul0~mul4를 모두 합하여 최종적인 필터링 결과인 sum을 도출한다. Sum은 FIR Filter의 출력으로서, 특정 주파수 성분이 필터링된 신호를 나타낸다. Sum은 DFF\_10bit 모듈의 입력으로 들어가고, clk 신호에 맞춰 최종 출력인 dout으로 전달되어, 결론적으로 최종 필터링을 거쳐 결과를 안정적으로 출력할 수 있도록한다.  **- 작성한 Test Bench Code가 어떤 동작을 하는지?**  Clk은 clock 신호이고, rst는 rest 신호로 FIR filter의 동작과 초기화를 제어한다. Rst가 활성화되면 모듈 내부의 FF 상태는 초기화된다. Din은 signed 4bit 신호로, filter에 특정한 입력값을 인가하기 위해 사용된다. Dout은 FIR\_filter 모듈의 signed 10bit 출력으로, 최종 필터링된 신호가 출력된다. Test bench에서는 dout의 값은 직접 제어하지않고, FIR\_filter 모듈의 출력으로 자동 연결된다.  FIR\_filter uut (  .clk(clk),  .rst(rst),  .din(din),  .dout(dout)  );  를 통해 FIR\_filter 모듈을 uut 이름으로 인스턴스화했다. Clk, rst, din, dout 신호를 이름에 의한 입력으로 입력하여 test bench에서 신호를 제어할 수 있게한다.  initial begin  clk = 1'b1; // 클럭 초기 설정 (1로 설정)  rst = 1'b0; // 리셋 비활성화로 초기 설정  #7 rst = 1'b1; // 7ns 후에 리셋 활성화  end  clk의 초기값을 1bit binary 1로 설정하고, rst의 초기값을 0으로 설정한다. Simulation 시작 후 7ns가 경과하면 reset 신호를 1로 설정하여 활성화한다. 이를 통해 FIR filter 모듈을 초기화하였다. 그 후에 10ns 주기로 clock을 생성하였다.  initial begin  din = 4'b0; // din 초기값을 0으로 설정  wait(rst == 1'b1); // 리셋이 활성화될 때까지 대기  입력 신호 din을 4bit로 0으로 설정하고, rst 신호가 1로 활성화될 때까지 wait 동작을 한다. Rst가 1이 되면 FIR filter가 초기화되고, 이후부터 입력 신호 패턴을 필터링하게 된다. 이후 세 단계로 din에 값을 넣어준다.  Step1. -7부터 7까지 증가하는 입력  for (i = -7; i <= 7; i = i + 1)  repeat (4) @(posedge clk) din = i;  i값을 -7부터 7까지 증가시키며 din에 적용한다. Clock의 posedge에서 din에 현재의 i값을 4번 반복하여 입력한다는 의미이다. 즉, din은 4 clk 주기동안 같은 값을 유지하게 된다. 이 단계에서는 filter가 -7부터 7까지의 값들이 증가하면서 인가되는 상황을 simulation하는 것이다.  Step2. 7부터 -7까지 감소하는 입력  for (i = 7; i > -7; i = i - 1)  repeat (4) @(posedge clk) din = i;  for 문을 통해 i를 7부터 -7까지 감소시키며 din에 적용한다. 마찬가지로 4 clk 주기 동안 같은 i값을 din에 적용하는데, 이때 filter에 7부터 -7로 감소하는 입력 신호를 인가하여, filter의 대칭 응답을 test하는 부분이다.  Step3. -7부터 7까지 반복하여 양수, 음수를 번갈아 입력  for (i = -7; i <= 7; i = i + 1) begin  repeat (4) @(posedge clk) din = i; // 양수 입력  repeat (4) @(posedge clk) din = -i; // 음수 입력  end  첫번째 repeat 문에서는 i값을 4 clk 주기동안 입력하고, 두번째 repeat 문에서는 -i값을 4 clk 주기동안 입력하여 양수와 음수를 번갈아가며 din에 입력한다. 이때는 filter에 양수와 음수를 교대로 입력하기에, filter가 양수, 음수 변화에 따라 어떻게 작동하는지 test하는 것이다.  3 종류의 simulation을 마치고 40ns 후에 simulation을 종료하였다.  **- Simulation 파형이 왜 그렇게 나온것인지?**  스크린샷, 라인, 그래프이(가) 표시된 사진  자동 생성된 설명  Simulation 결과는 tb\_FIR\_filter에서 실행된 FIR filter의 출력 파형을 보여준다. Din에 다양한 입력값을 입력했을 때, FIR filter가 어떻게 반응하는지 확인할 수 있다.  **1. 0ns~500ns구간**  해당 구간에서는 din이 -7에서 7로 증가한다. FIR filter는 이전 값들을 tap에 저장하고, 각 tap에 대칭적인 coeff를 곱한 후에 이들을 합하여 dout을 생성한다. Din이 점자적으로 증가하면서 FIR filter는 과거 입력 값들인 delay1~delay5를 포함한 결과를 출력하고, dout이 linear하게 증가하는 모습을 볼 수 있따. 해당 구간의 dout은 din이 증가함에 따라 점진적으로 증가하며, FIR coeff와 연산 결과에 의해 출력 신호는 완만한 상승 곡선을 보임을 확인할 수 있다.  **2. 500ns~1000ns 구간**  해당 구간에서는 din이 7에서 -7까지 감소한다. FIR filter는 이전 tap에 저장된 값을 사용해 입력 신호가 감소하는 상황에 맞춰 dout을 생성한다. Din이 점진적으로 감소하므로 dout도 감소하고, 이때 filter의 구조로 인해 linear한 형태의 파형이 나타난다. Dout이 linear하게 감소하면서도 약간의 delay를 가지고 있기에 filter 특유의 delay된 응답이 나타난다.  **3. 1000ns 이후 구간**  해당 구간에서는 양수와 음수가 번갈아가며 din으로 입력된다. 예를 들어, 7,-7,6,-6,5,-5..의 패턴으로 변한다. FIR filter는 과거 입력 신호들을 저장하여 연산을 수행하는데, 양수와 음수가 번갈아가며 인가되기에 출력 신호는 진동하는 파형을 보인다. Dout의 파형은 이때, Damped Oscillation 형태를 띠게 되고, din이 양수와 음수로 반복될 때 filter는 이전 입력들을 유지하고 평균화하는 특성이 있기에 이러한 형태의 파형이 나타난다. 이때 발생하는 진동은 FIR filter가 대칭 구조를 가지고 있기에 특정 주파수 성분을 강조하거나 제거하는 필터링 효과로 인해 발생한 결과라고 해석할 수도 있다.  또한, test한 두가지 경우에 대해서는 오버플로우가 발생하지 않았다. 10bit signed 형식에서 표현할 수 있는 값의 범위는 -512에서 511로 충분히 큰 범위를 가지기에, 테스트한 값들은 오버플로우가 발생하지 않았다.  **- 작성한 Code가 잘 동작하지 않는다면 어디서 문제가 발생한 것인지?**    위 사진은 post-synthesis simulation에서 파형이 깨지는 모습이다. FIR filter에서는 signed 숫자를 다루기에 signed 숫자 4bit+4bit를 하면 출력을 선언할 때, 5bit를 선언해야한다. 하지만, 이러한 부분을 잘못 계산하였고, signed 곱셈 역시, 4bitx4bit에서 결과값은 부호 1bit, 크기 6bit의 총 7bit가 선언되어야 하는데, 이를 잘못계산하여 6bit로 선언하여 파형이 깨진 것이라 판단하였다.  **- 오류를 해결했다면, 어떤 부분에서 해결을 했고, 왜 오류가 발생하였는지?** |

스크린샷, 텍스트, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

앞서 설명한 오류 원인에 맞게 signed 4bit들의 곱셈에 대해 결과값 7bit를 정상적으로 선언하였고, 곱셈 결과(mul0~mul4)의 합을 10bit의 sum으로 선언하여 다시 simulation을 진행하여 정상 작동함을 확인하였다.