COTS SW Dedication

Introduction

정세진
Dependable Software Laboratory
Konkuk Univ.
2015.10.07

IP CORE LIBRARY 관련 실험

IP Core Library

- IP (Intellectual Property) Core in FPGA
  - 복잡한 시스템의 설계를 간단히 하기 위해(편의성 및 효율성) 미리 정의한 기능과 회로의 라이브러리
    - Vendor, 3rd party 등에서 제공
    - Design, chip, cell, logic, etc
  - Microsemi 에서는 Libero SoC 안의 Smart Design tool 에서 IP Core 사용을 제공

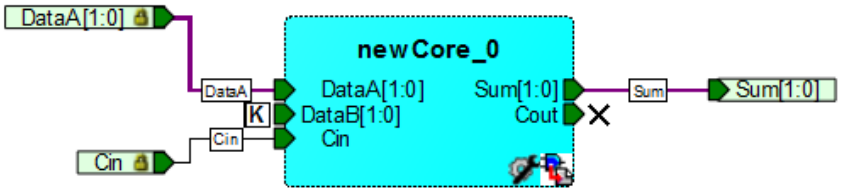
추가 고려사항

RTL design 지원 도구 확인

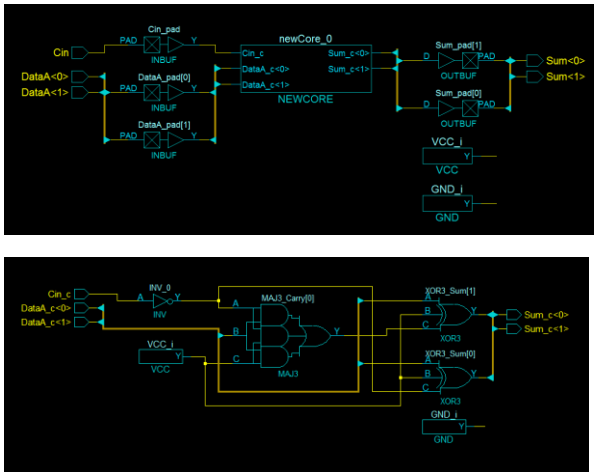
• 각 벤더 별로 RTL design 생성 도구 존재

• Microsemi Libero 11.5 에서는
  - Smart Design을 지원 (블록 다이어그램 형태로 작성), 여러 basic block(ex adder, multiplexer) 존재
    → HDL design을 직접 작성하는 경우 → 같은 기능을 하는 코드를 basic 블록의 코드로 대체여부 확인 필요
      (Smart Design에서 디자인 블록을 사용하는 경우 → 해당 블록의 코드를 프로젝트에 import)

  - Smart Design을 이용해 작성된 블록 디자인 → FBD 와 유사
    (Smart Design은 작성된 디자인을 Verilog로 변환)
    → Smart Design을 이용하여 디자인 후 Verilog 로 변환하는 것과,
    → FBD를 이용하여 디자인 후 RTL design (Verilog, VHDL) 으로 변환하는 것의 차이점 및 장단점 고려 필요



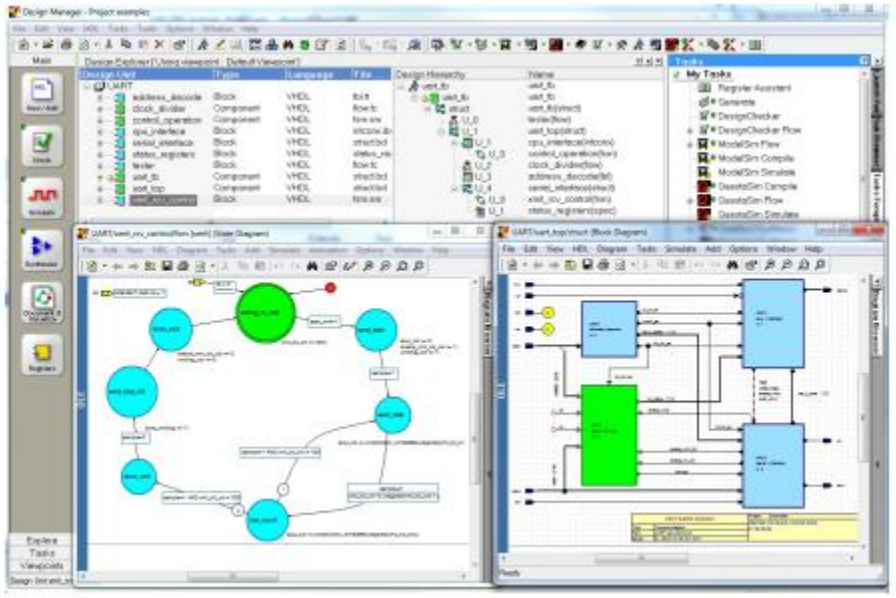
Actel Smart Design을 이용하여 작성한 adder (newCore_0) 디자인



Synplify Pro를 사용해 합성한 결과(위) 와 newCore 내부 (아래)

RTL Design Tool

- Microsemi (Actel)
  - Smart Design
- Mentor Graphics
  - HDL Designer
  - Design rule checking 제공
- Xilinx
  - ISE
  - ISE WebPACK
- Altera
  - Quartus 2
    - Quartus 2를 통해서
    - 직접 코드 작성
    - Design, schematic 작성 모두 가능



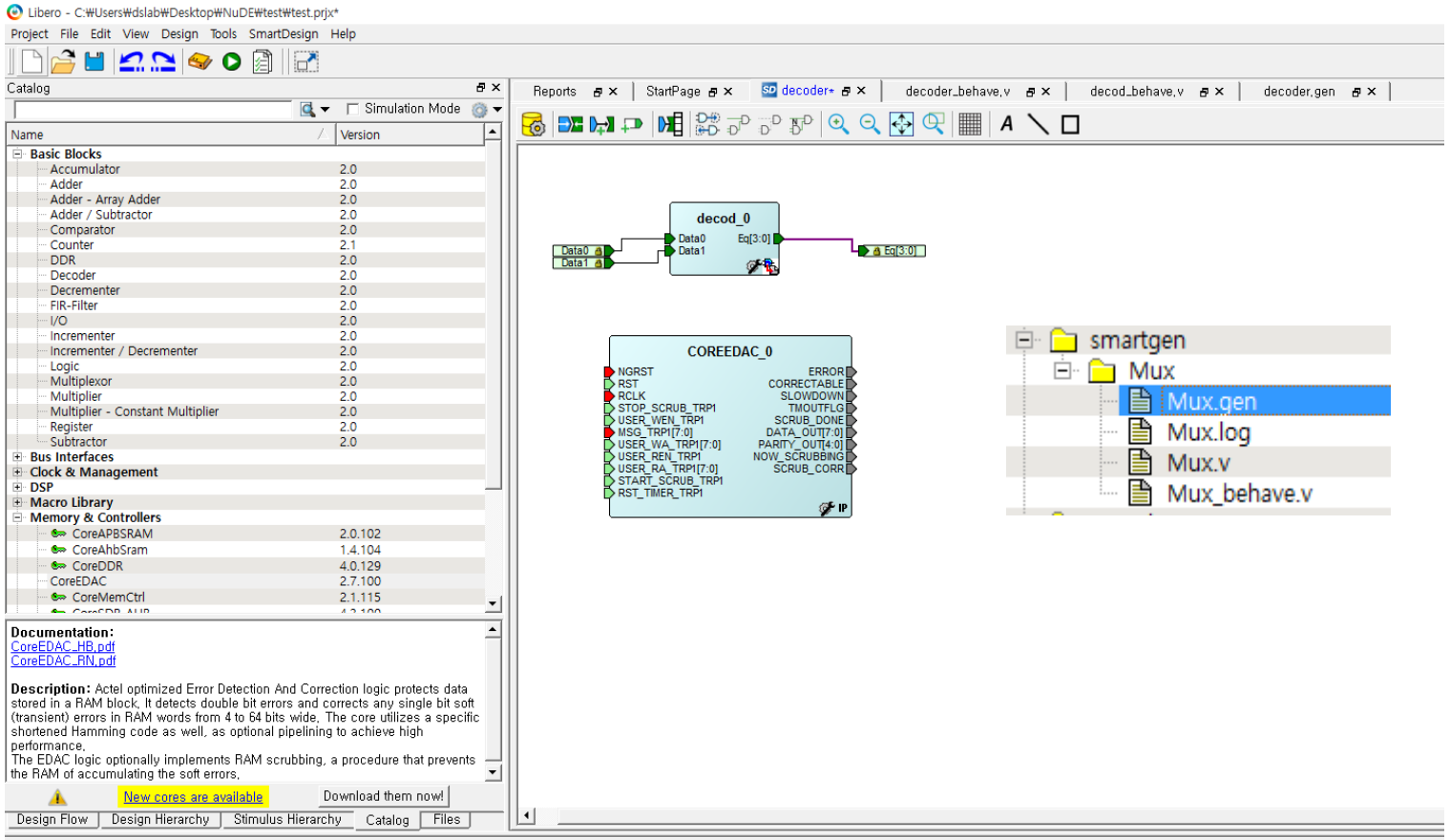
Mentor Graphics HDL Designer

	Xilinx's **ISE** or the free **ISE WebPACK**	Altera's **Quartus II** or the free **Web Edition**
Design-entry	VHDL, Verilog, ABEL, Schematic, EDIF	VHDL, Verilog, SystemVerilog, AHDL, Schematic, EDIF
Core generator	Yes (**CORE Generator**)	Yes (**MegaWizard Plug-Ins**)
Functional simulation	No	No (last version with simulation was 9.1SP2)
Testbench simulation	Use **ISim**	Use **ModelSim-Altera Starter Edition**
Synthesis/P&R	Free version limited to small & medium devices	Free version limited to small & medium devices
Programming	Yes	Yes
FPGA editor	Yes (FPGA editor)	Yes (Chip Editor)
Embedded logic analyzer	**ChipScope PRO** (a separate product - not free)	**SignalTap II** (included in Quartus II/Web edition)
Older versions	Available from **ISE Classics**	Available from the **Quartus II Software Archive**
OS support	Windows + Linux	Windows + Linux
Price	Free version: $0 Full version: starting at $2995 for a 12 month license	Free version: $0 Full version: $2995 for a 12 month license
Software matrix	Check **here**	Check **here**

ISE and Quartus 2 spec

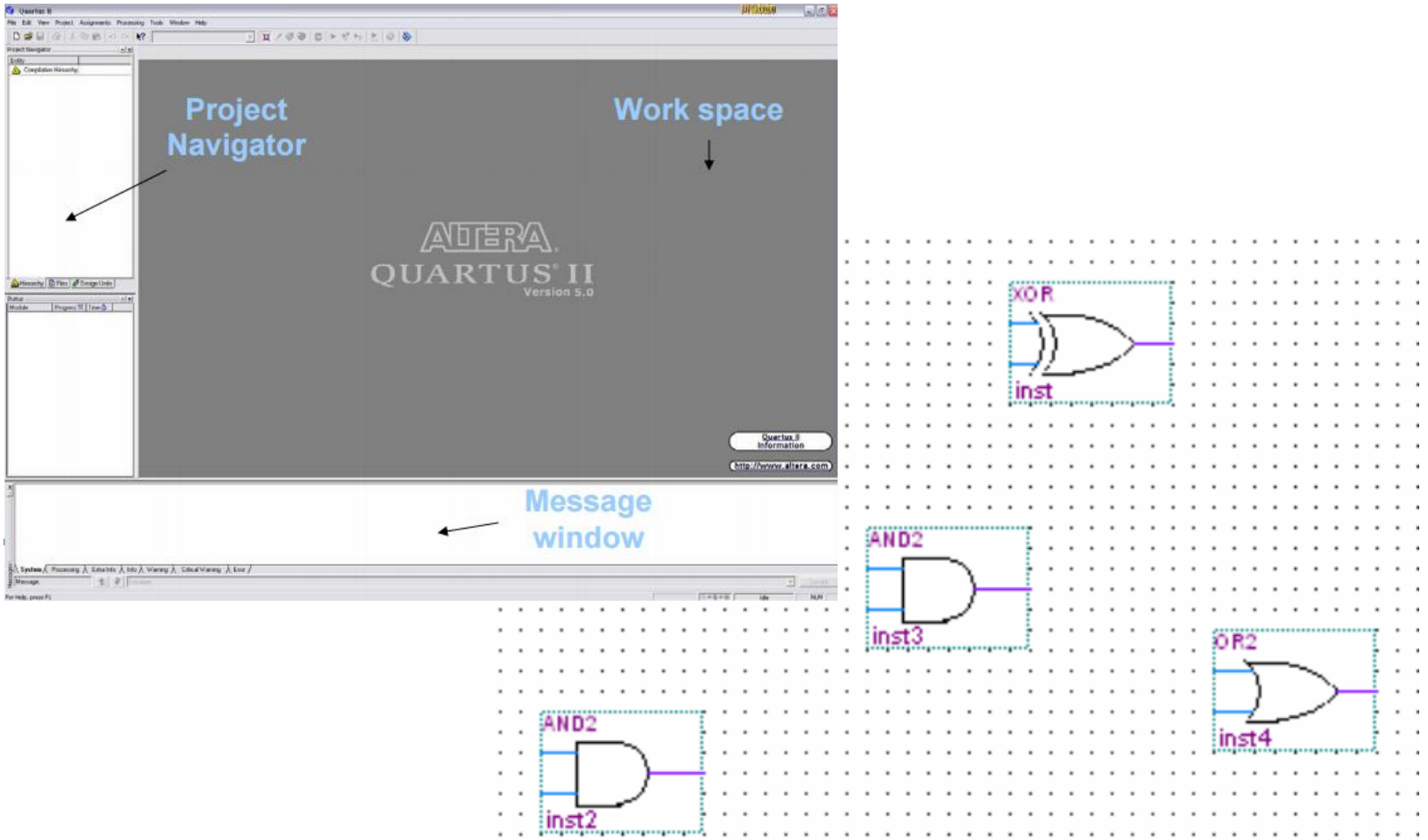
Smart Design Example

- **Smart Design을 이용한 디자인 작성 화면**



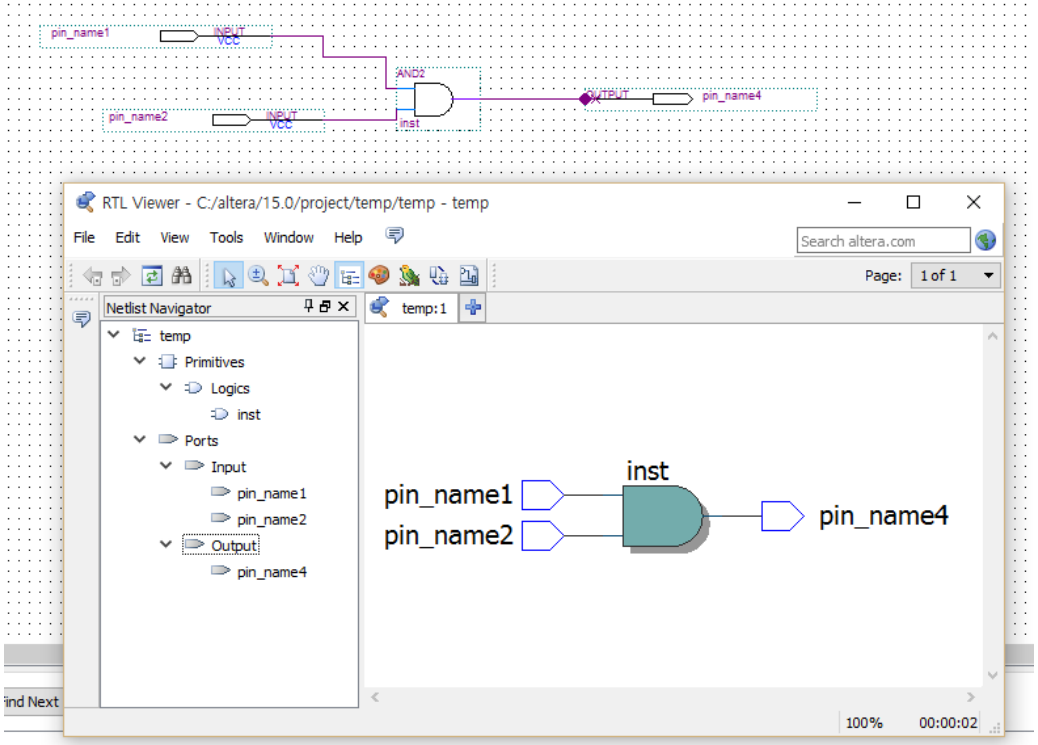
Quartus 2 Example

- **Design 예제**



Quartus 2 Example

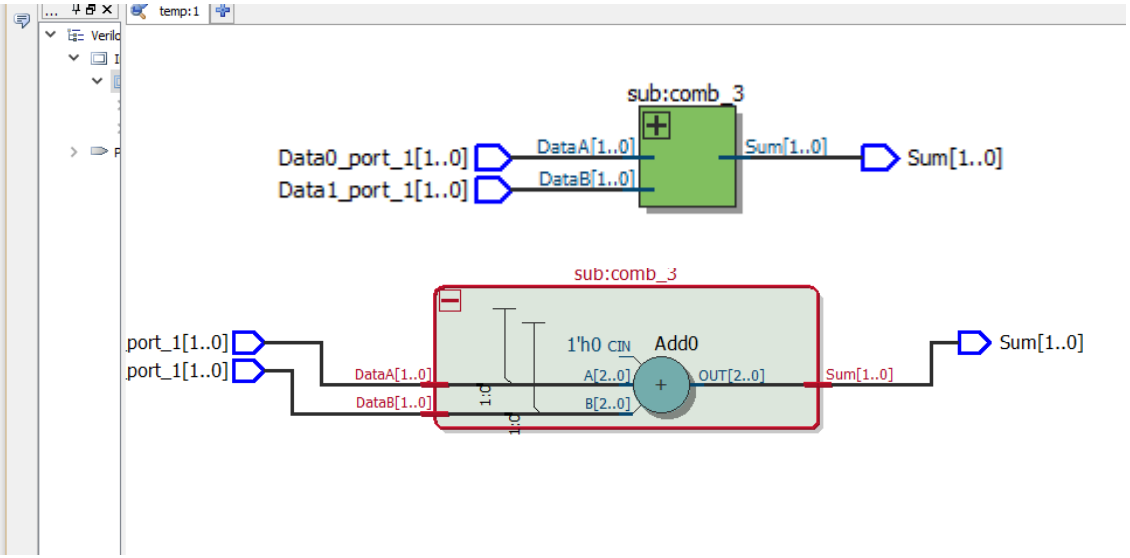
- 다음 예제에 대해 합성한 결과 (Netlist viewer)
  - 합성 도구는 Synplify Pro 선택 (설정)



Quartus 2 Example

- Verilog 코드를 넣고 합성 해본 결과
  - 합성 도구는 Synplify Pro 선택 (설정)

```
`timescale 1 ns/100 ps

module Verilog1(Data0_port_1, Data1_port_1, Sum);

input [1:0]  Data0_port_1;
input [1:0]  Data1_port_1;
output [1:0] Sum;

wire [1:0]   Data0_port_1;
wire [1:0]   Data1_port_1;
wire [1:0]  Sum;

sub( Data0_port_1, Data1_port_1, Sum);

endmodule

module sub(DataA, DataB, Sum);

  input [1:0] DataA, DataB;
  output [1:0] Sum;
  reg [1:0] Sum;

  always @ (DataA or DataB)
  begin

    Sum = DataA - DataB;
  end

endmodule
```



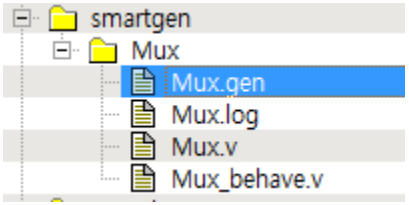
RTL Design Tool 실험

- Microsemi Libero SoC 의 Smart Design + Synplify Pro 조합

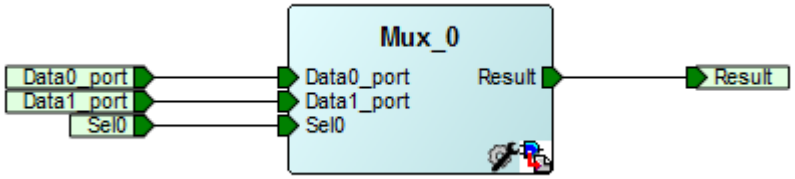
- Multiplexer 에 대해 4 가지 실험 수행

실험	Design	Verilog
실험 1	2 to 1 MUX	No interface
실험 2	2 to 1 MUX	With Interface
실험 3	2 to 1 MUX	IF 문 사용
실험 4	3 to 1 MUX	With Interface

- Smart Design에서 블록을 삽입할 경우 → 그에 맞는 structural Verilog code import
  - Verilog 코드의 기본은 smart design 에서 import한 코드 사용 + 수정



2 to 1 MUX : Smart Design



Smart Design을 이용한 Multiplexer 블록

```
`timescale 1 ns/100 ps
// Version: v11.4 11.4.0.112


module Mux(
        Data0_port,
        Data1_port,
        Sel0,
        Result
    );
input  Data0_port;
input  Data1_port;
input  Sel0;
output Result;


    MX2 MX2_Result (.A(Data0_port), .B(Data1_port), .S(Sel0), .Y(
        Result));

endmodule
```

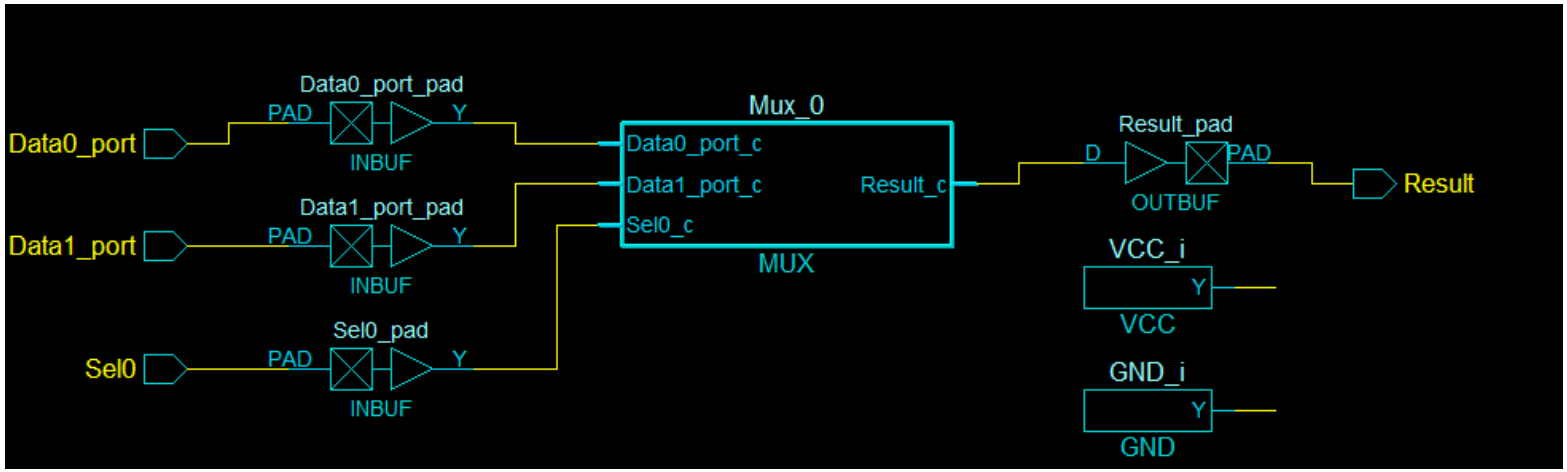
Smart Design을 통해 import 한 MUX 블록 코드 (Structural Verilog?)

Smart Design에서 사용하는 라이브러리 ->
macro library를 이용하여 vendor 내부적으로 작성한 코드를 사용

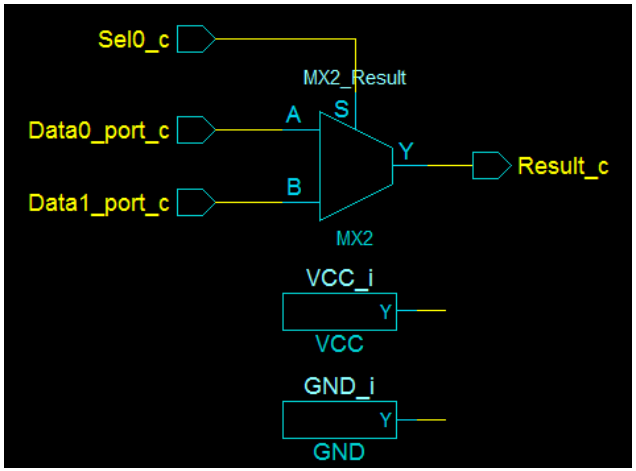
```
module Mux(Data0_port, Data1_port, Sel0,Result);

  input Data0_port;
  input Data1_port;
  input Sel0;
  output Result;
  reg Result;

  always @(Data0_port or Data1_port or Sel0)
  begin

    case ({Sel0})
      0 : Result = Data0_port;
      1 : Result = Data1_port;
      default : Result = 1'bx;
    endcase

  end

endmodule
```

Smart Design을 통해 import 한
MUX 블록 코드 (Behavioral Verilog)

Smart Design을 이용한 2 to 1 MUX의 합성 결과



Smart Design을 이용한 Mux의 Synthesis 결과

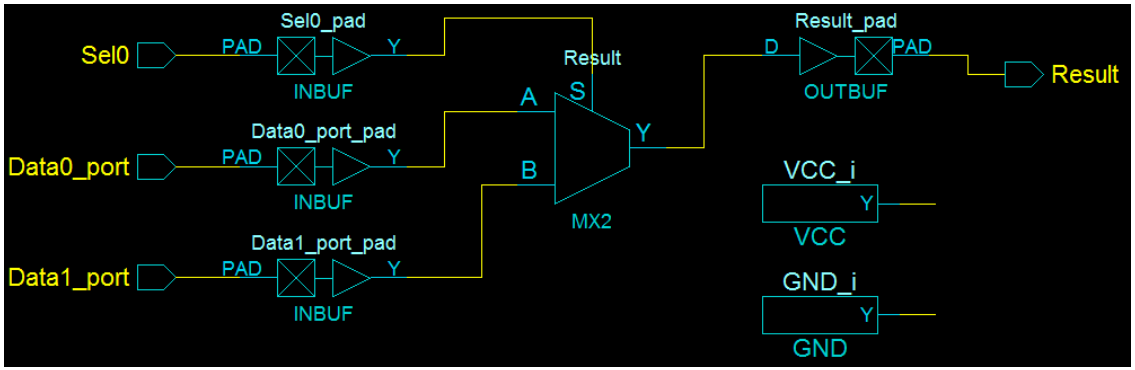


Smart Design을 이용한 Mux의 Synthesis 결과
Mux 내부

Verilog 를 이용한 2 to 1 MUX 의 합성 결과

- **Smart Design MUX의 behavioral Verilog 와 같은 코드**

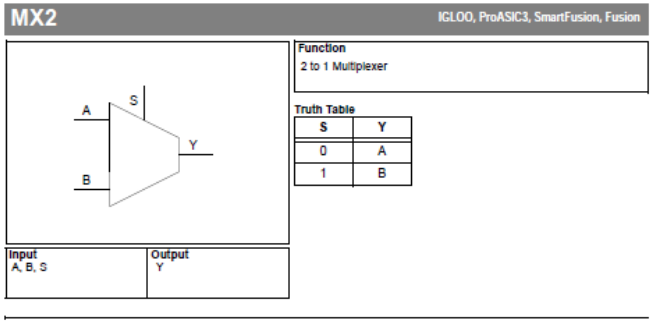
```
module mux_test(Data0_port, Data1_port, Sel0, Result);

  input Data0_port;
  input Data1_port;
  input Sel0;
  output Result;
  reg Result;

  always @(Data0_port or Data1_port or Sel0)
  begin

    case ({Sel0})
      0 : Result = Data0_port;
      1 : Result = Data1_port;
      default : Result = 1'bx;
    endcase

  end

endmodule
```



합성 결과

실험 1 결과

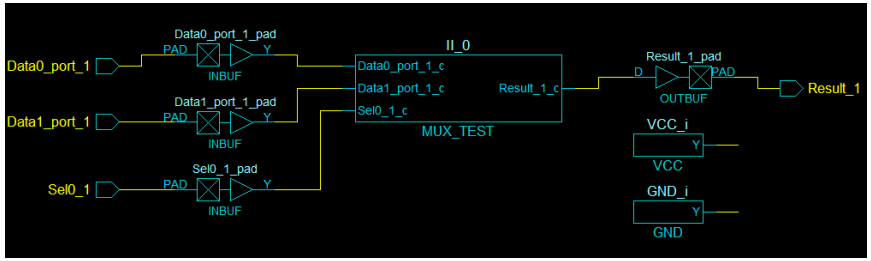
- INBUF, OUTBUF, MX2는 macro library
- Smart Design이용한 합성 결과에서는 MUX 블록 이용 확인
- 코드만을 사용한 합성 결과에서는 사용하지 않음을 확인
- 두 디자인 모두 MX2 (2:1 MUX macro library) 이용 확인
- 실험 1 결과
  - 같은 기능을 하는 MUX 블록으로 대체 사용하지 않음을 확인
  - 차이점이 INBUF, OUTBUF 로 인한 발생 여부에 대해 확인 필요



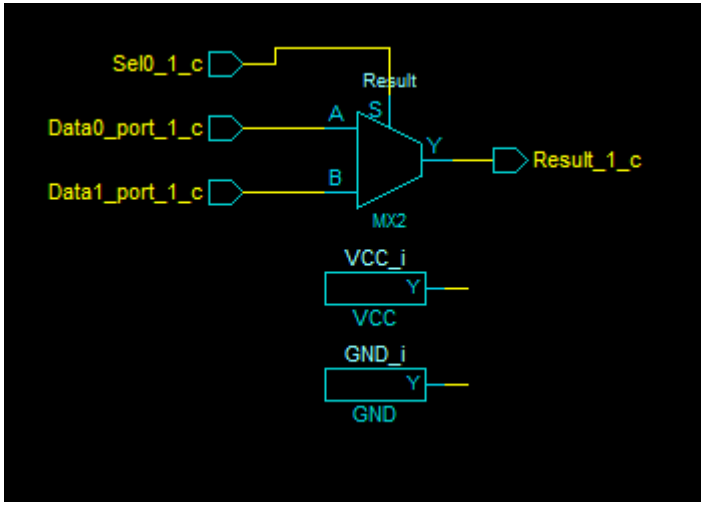
MX2 : Multiplexer macro library

실험 2 결과

- 실험 2 : 실험 1 의 Verilog 코드에 interface 추가
- Smart Design에서 만든 것과 같은 MUX 코드를 사용 하였기에 유사한 결과 확인
- MUX 부분에 대해 같은 코드를 사용 -> 같은 기능을 하는 다른 형태 코드로 변경 해서 확인 필요



Interface 만든 Verilog 합성 결과



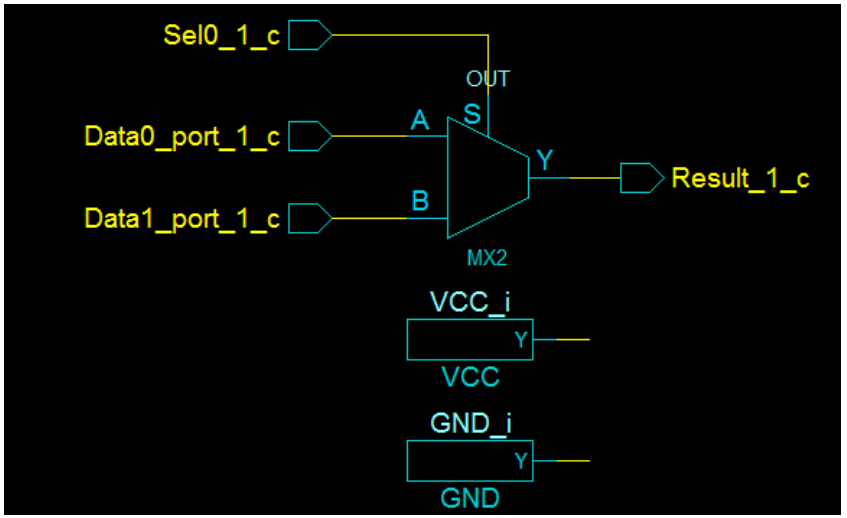
Interface 만든 Verilog MUX_TEST 합성 결과

실험 3 결과

- 실험 2 에서 MUX 코드를 같은 기능을 하는 다른 코드로 변경
  - CASE -> IF/ELSE 로 변경
- If 문을 사용한 mux 에서도 MX2 사용
  - 유사함을 확인
  - Macro library에서 MX2 를 사용하기 때문으로 추정

```
module mux_test(K, IN0, IN1, OUT);

    input K;
    input  IN0;
    input  IN1;
    output  OUT;

    reg OUT;

    always @(IN0 or IN1 or K)
        if(K) begin
            OUT = IN1;
        end else begin
                OUT = IN0;
        end
endmodule
```

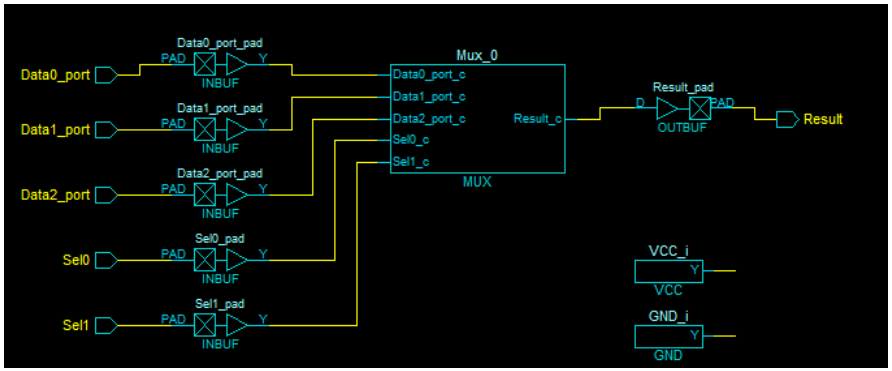
IF 문을 이용하여 작성한 MUX Verilog code

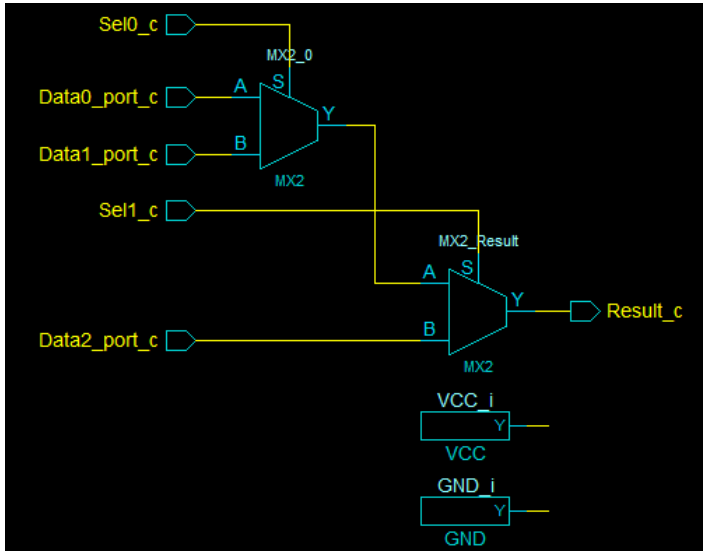


IF 문을 이용하여 작성한 MUX Verilog code 합성 결과

실험 4 : 3 to 1 MUX 를 이용한 실험

- **Smart Design의 3 to 1 MUX 합성결과**





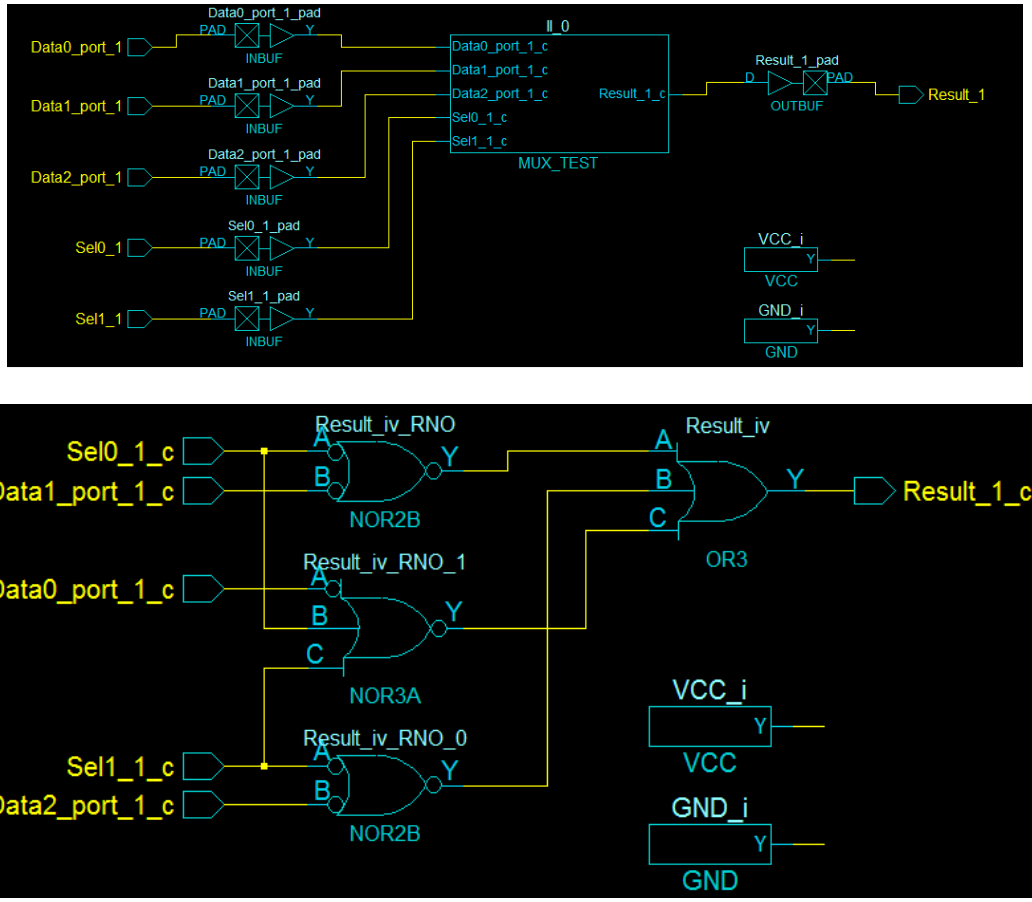
실험 4 : 3 to 1 MUX 를 이용한 실험

- 3 to 1 MUX 의 예제를 interface 와 module 사용의 형태로 Verilog 로 작성하여 실험

```
mux_test( Data0_port_1, Data1_port_1, Data2_port_1, Sel0_1, Sel1_1, Result_1);

endmodule

module mux_test(Data0_port, Data1_port, Data2_port, Sel0, Sel1, Result);

  input Data0_port;
  input Data1_port;
  input Data2_port;
  input Sel0;
  input Sel1;
  output Result;
  reg Result;

  always @(Data0_port or Data1_port or Data2_port or Sel1 or Sel0)
  begin

    case ({Sel1,Sel0})
      0 : Result = Data0_port;
      1 : Result = Data1_port;
      2 : Result = Data2_port;
      default : Result = 1'bx;
    endcase

  end

endmodule
```

실험 4 : 3 to 1 MUX 를 이용한 실험

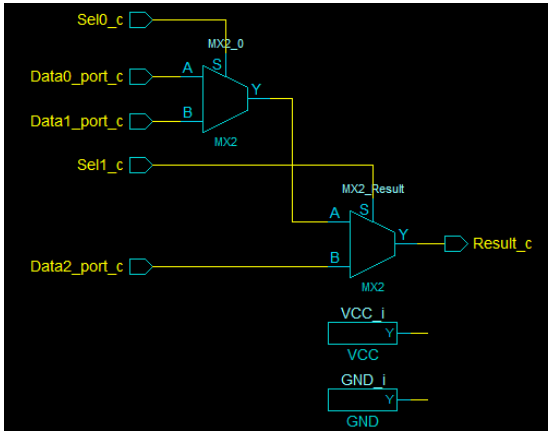
- Verilog 로 작성한 3 to 1 MUX 합성결과



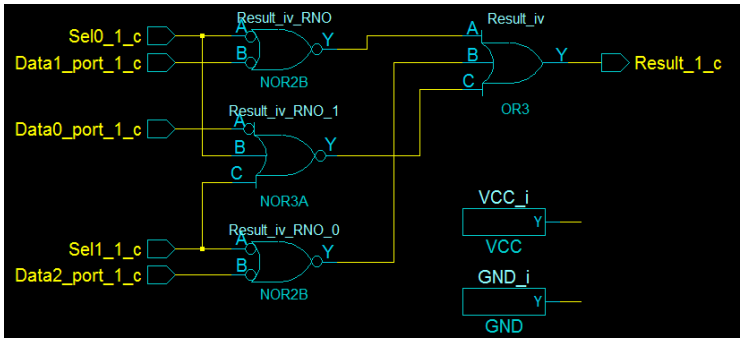
Verilog 로 작성한 3 to 1 MUX 합성 결과

실험 4 결과

- 3 to 1 MUX 기능을 하는 부분에서 다른 합성결과 도출
  - 같은 기능을 하는 코드를 사용 하지만 다른 합성결과
  - 결론적으로, Multiplexer 에 대해서는 ‘Smart Design의 블록들의 코드를 사용하지 않는다’ 라고 할 수 있을 것 으로 보임



Smart Design으로 작성한 3to1 Multiplexer 합성



Verilog로 작성한 3to1 Multiplexer 합성

Smart Design & Verilog 실험 추가

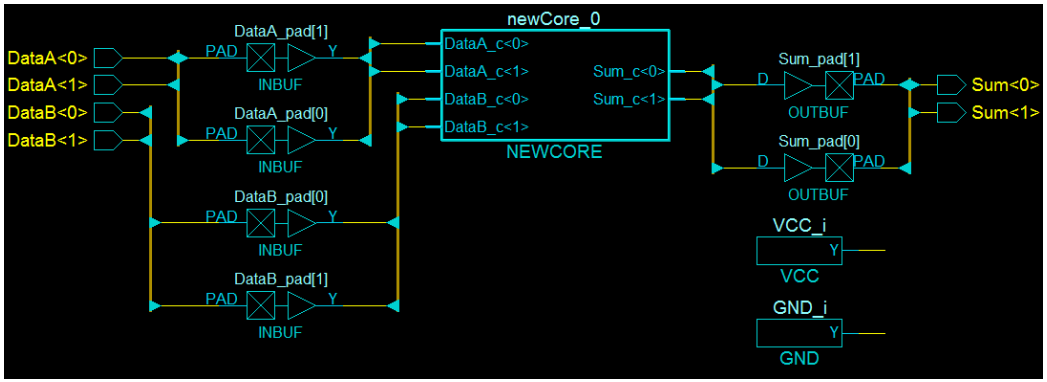
- 코어 부분의 코드를 그대로 이용
  - Subtractor
  - Adder (2 bit, 4 bit, 8 bit)
  - Multiplier
  - Comparator
  - Counter
  - Incrementer (Decrementer)

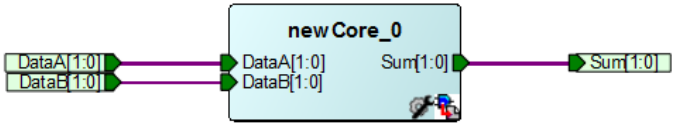
- FIX_RISING 에서 사용한 basic logic 종류
  - Adder
  - Comparator
  - Mux

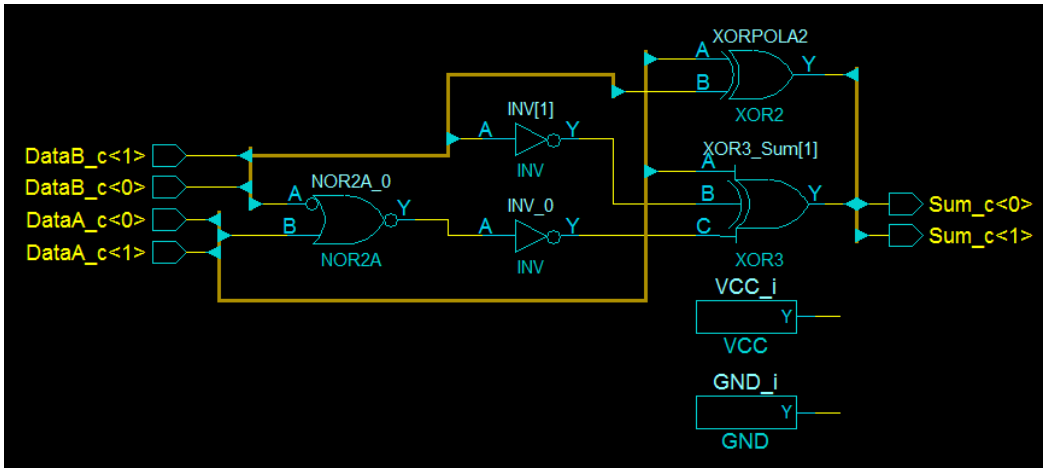
- FIX_RISING 에서 사용한 macro library 종류
  - MX2
  - AND2
  - OR2

Subtractor : Smart Design

- Smart Design을 이용한 subtractor 블록 및 합성결과

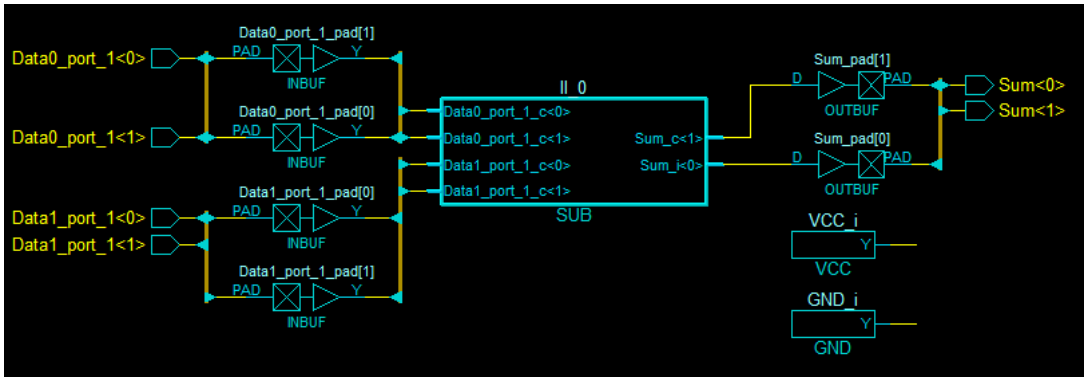


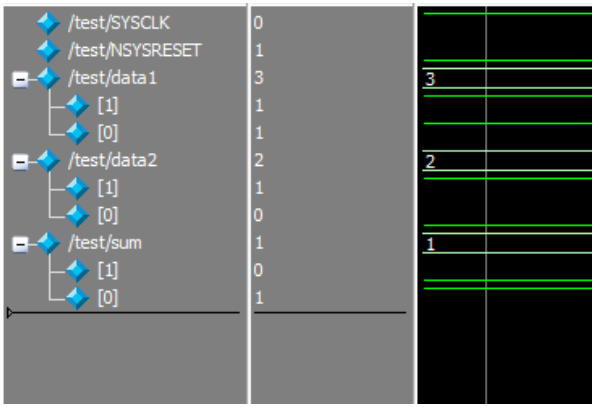


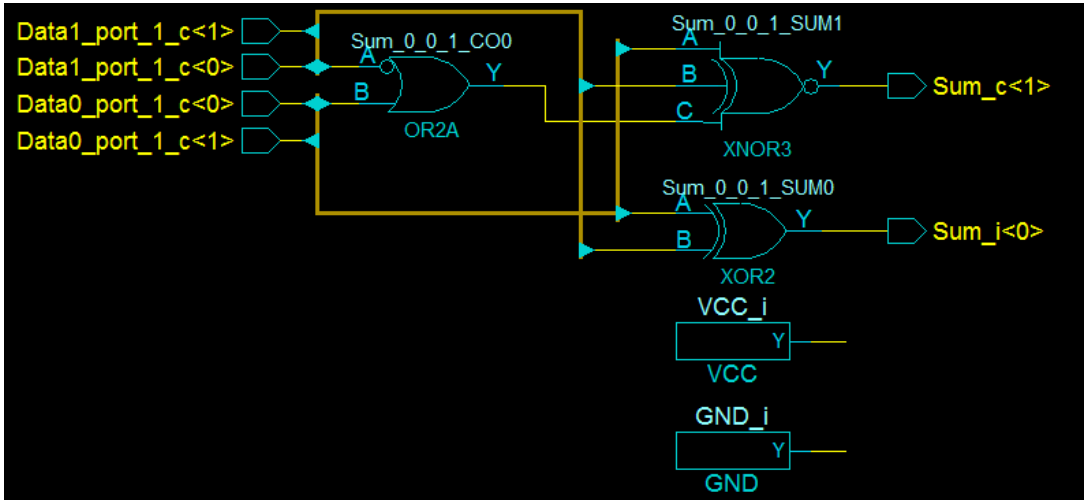


Subtractor : Verilog

- Verilog 로 작성한 subtractor 합성결과 및 simulation 화면

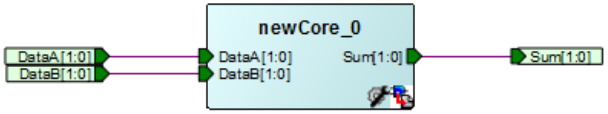


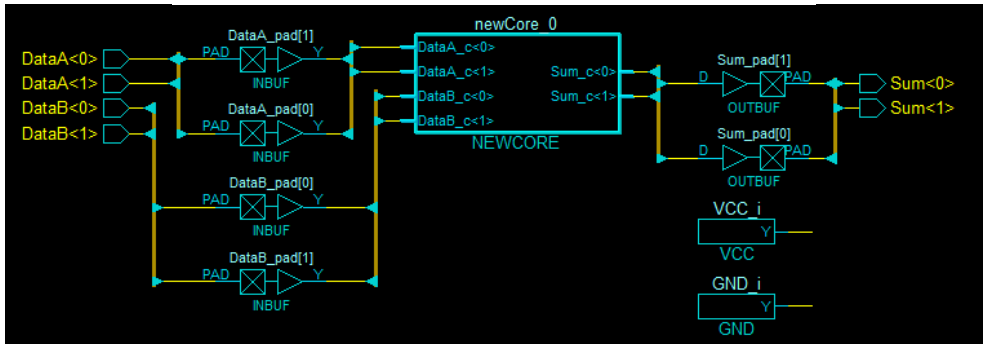


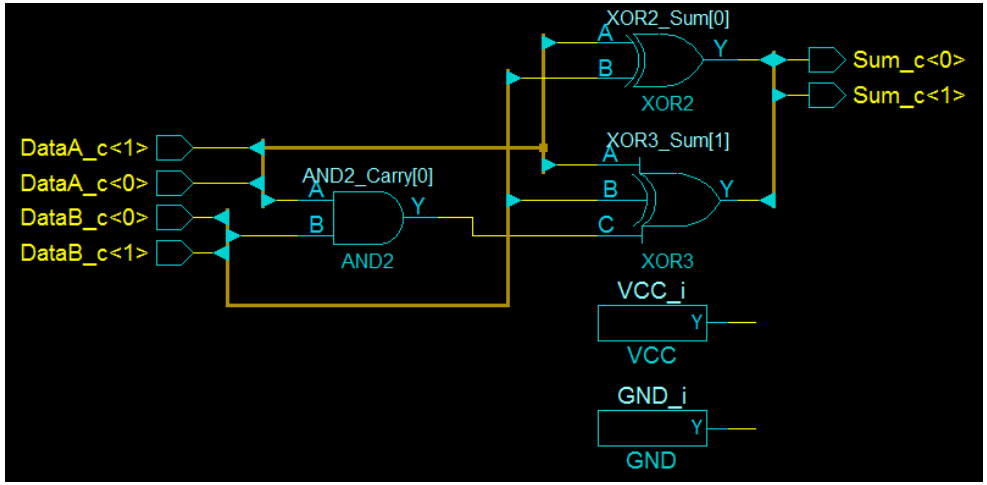


Adder : Smart Design

- Smart Design을 이용한 adder 블록 및 합성결과

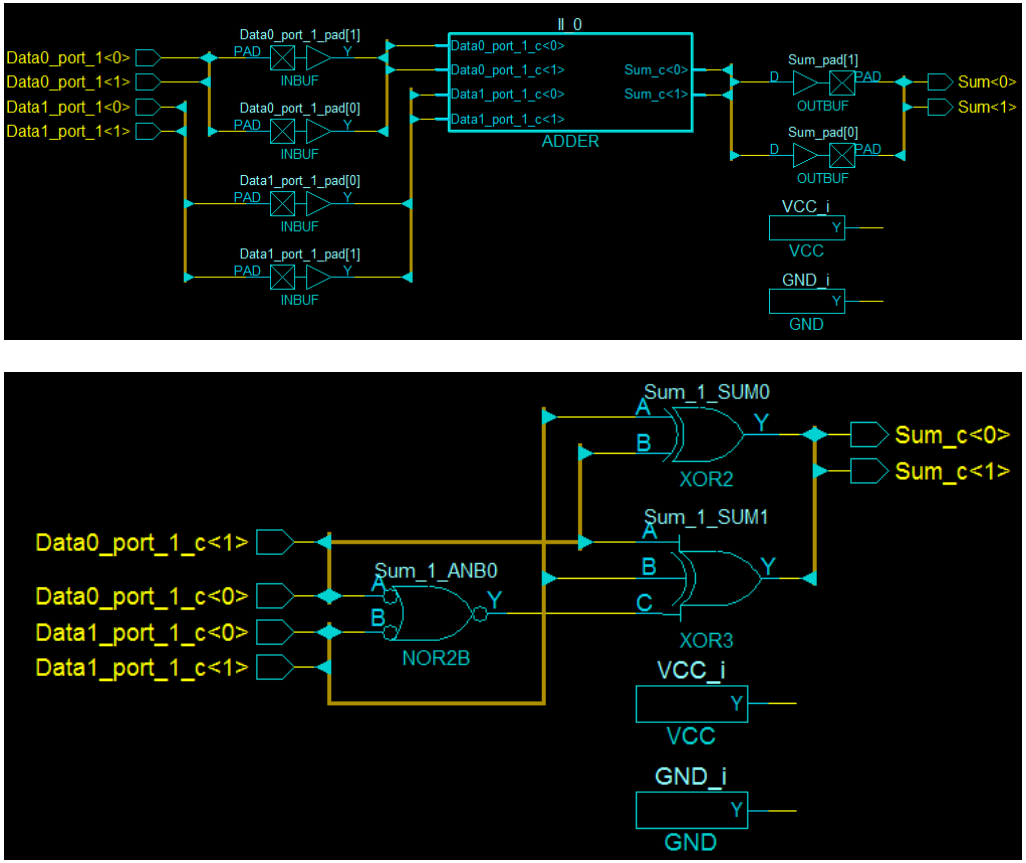






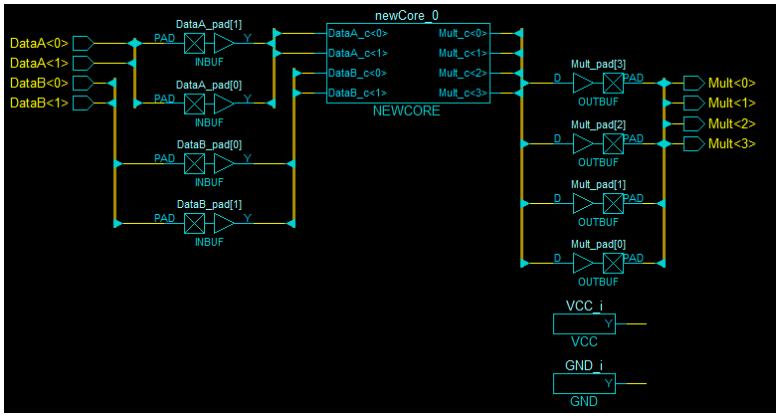
Adder : Verilog

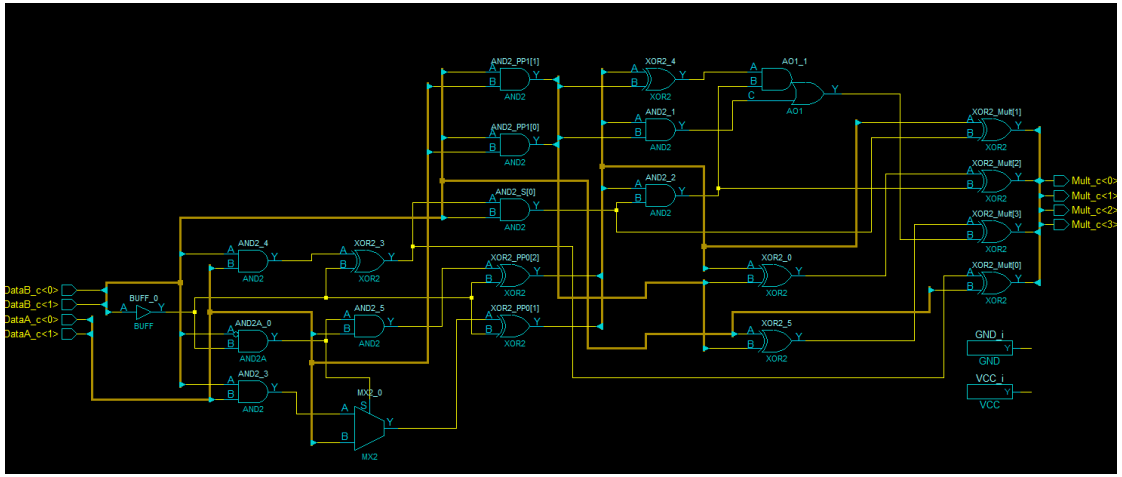
- Verilog 로 작성한 adder 합성결과



Multiplier : Smart Design

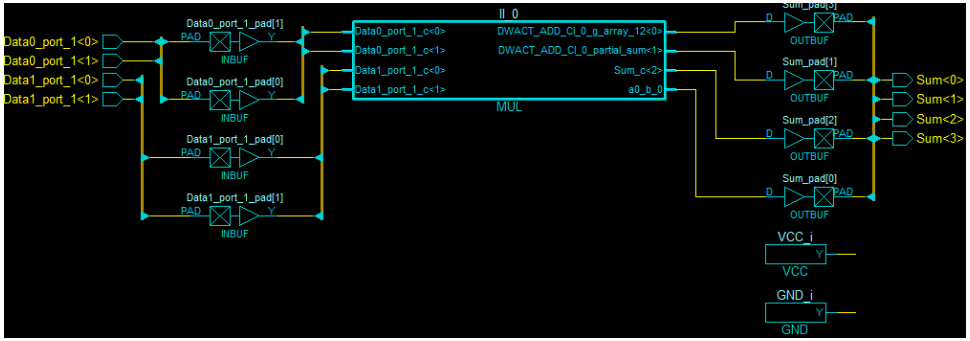
- **Smart Design을 이용한 multiplier 블록 합성결과**

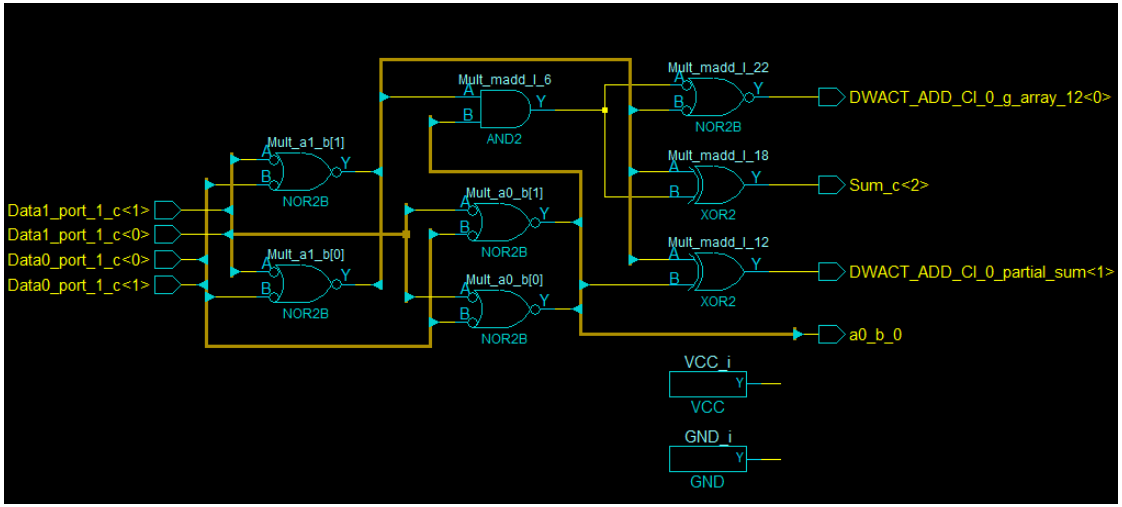




Multiplier : Verilog

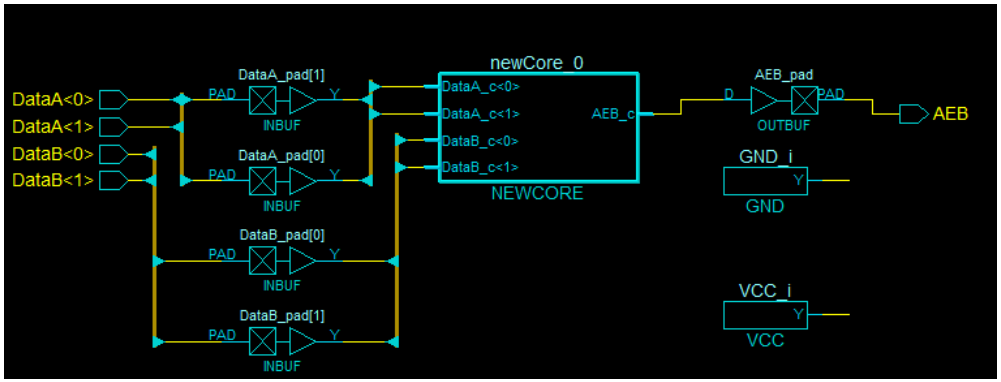
- Verilog로 작성한 multiplier 블록 합성결과

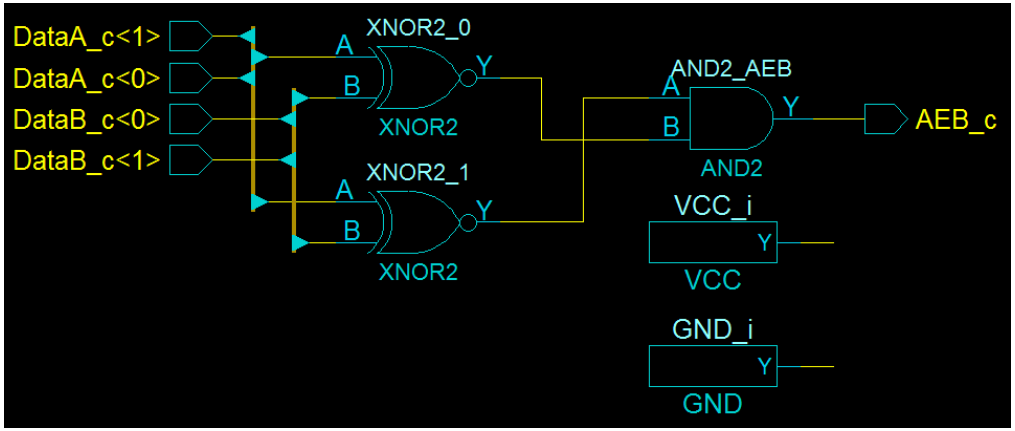




Comparator : Smart Design

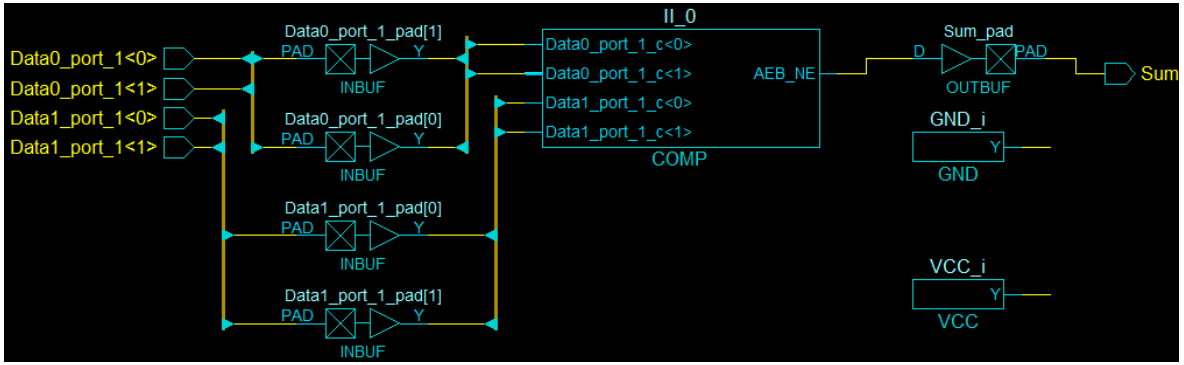
- **Smart Design을 이용한 Comparator 블록 합성결과**

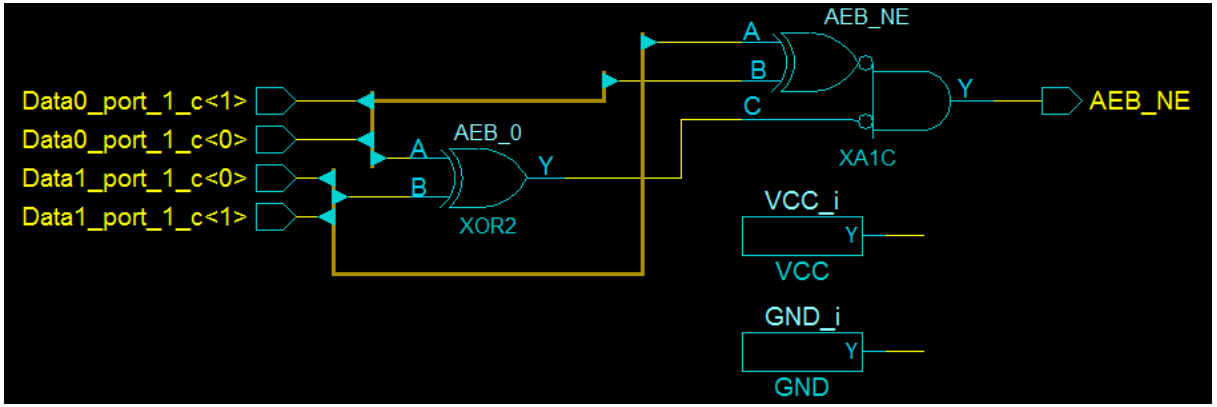




Comparator : Verilog

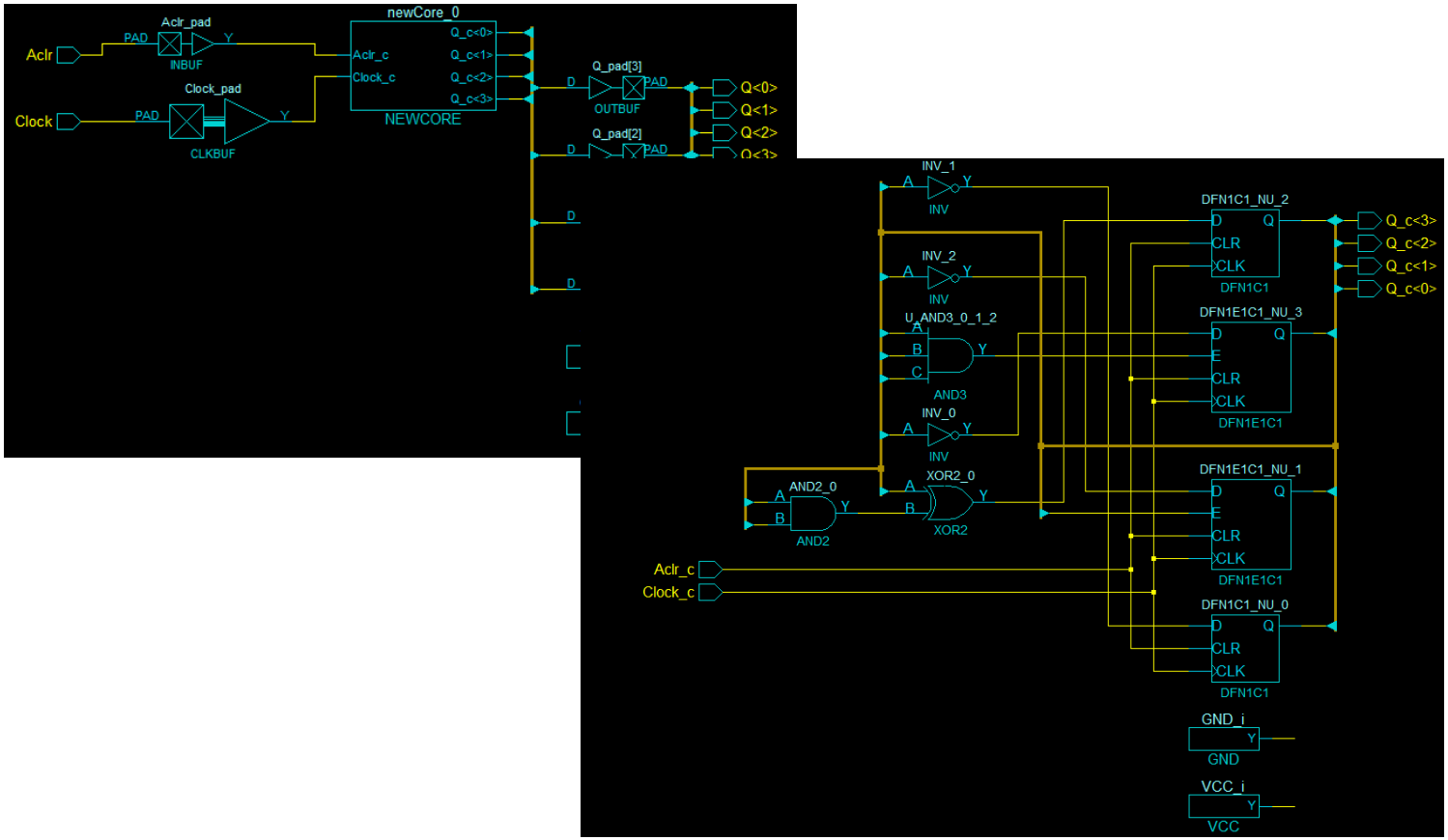
- Verilog로 작성한 Comparator 블록 합성결과





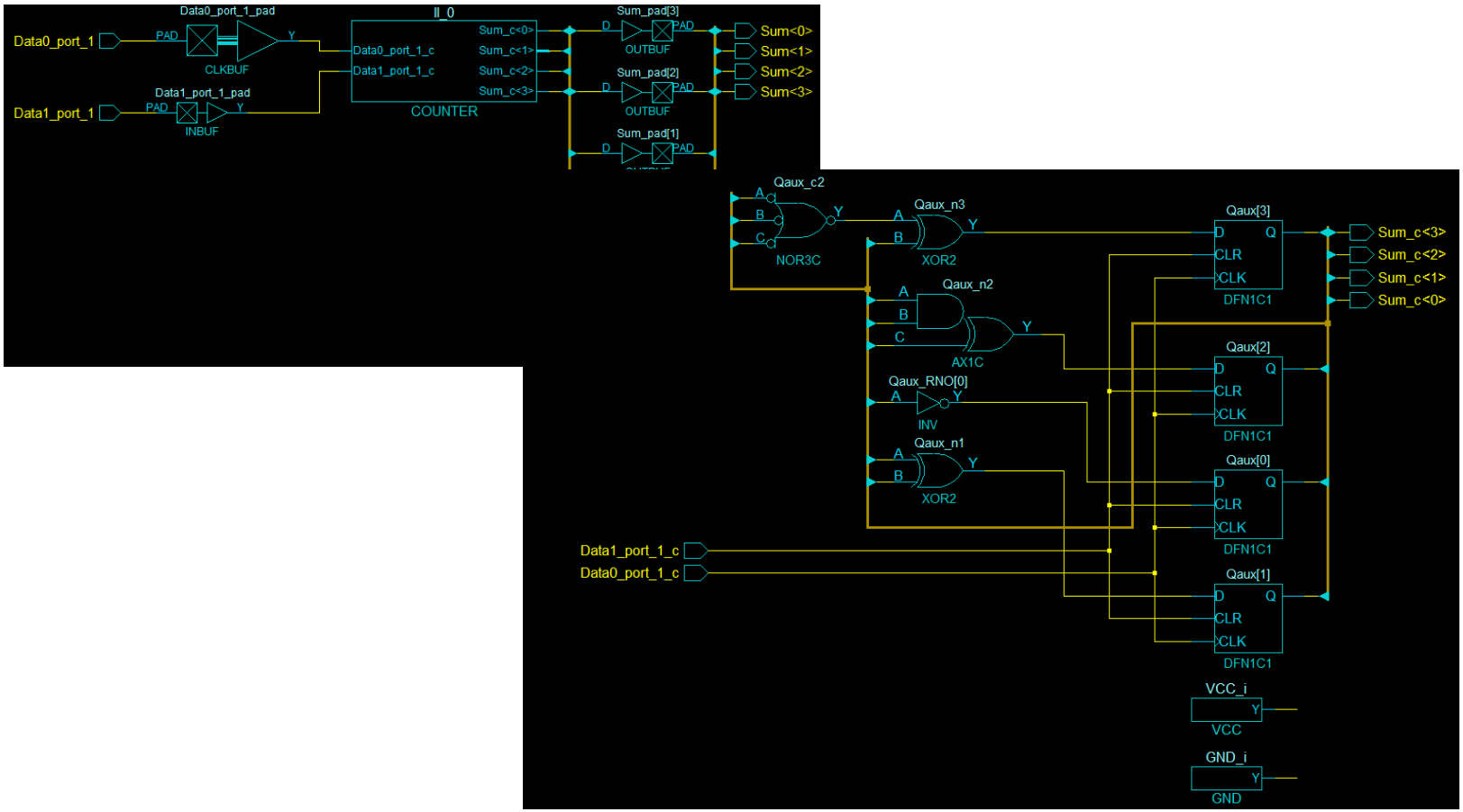
Counter : Smart Design

- Clock에 따라 count 증가시키는 로직
- Smart Design을 이용한 counter 블록 합성결과



Counter : Verilog

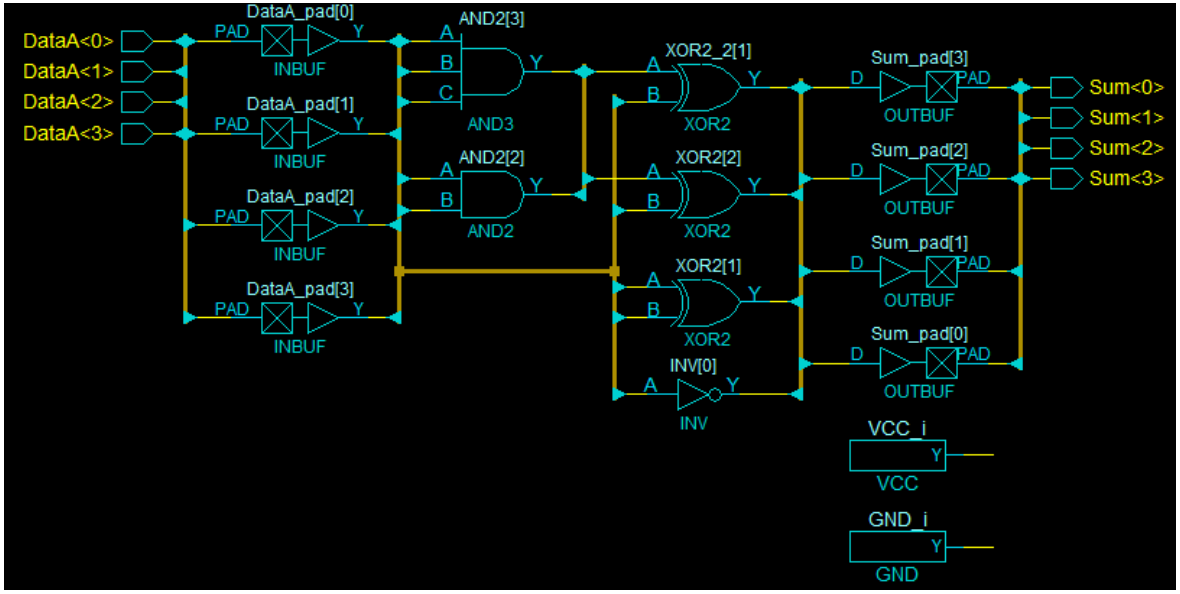
- Verilog로 작성한 counter 블록 합성결과



Incrementer : Smart Design

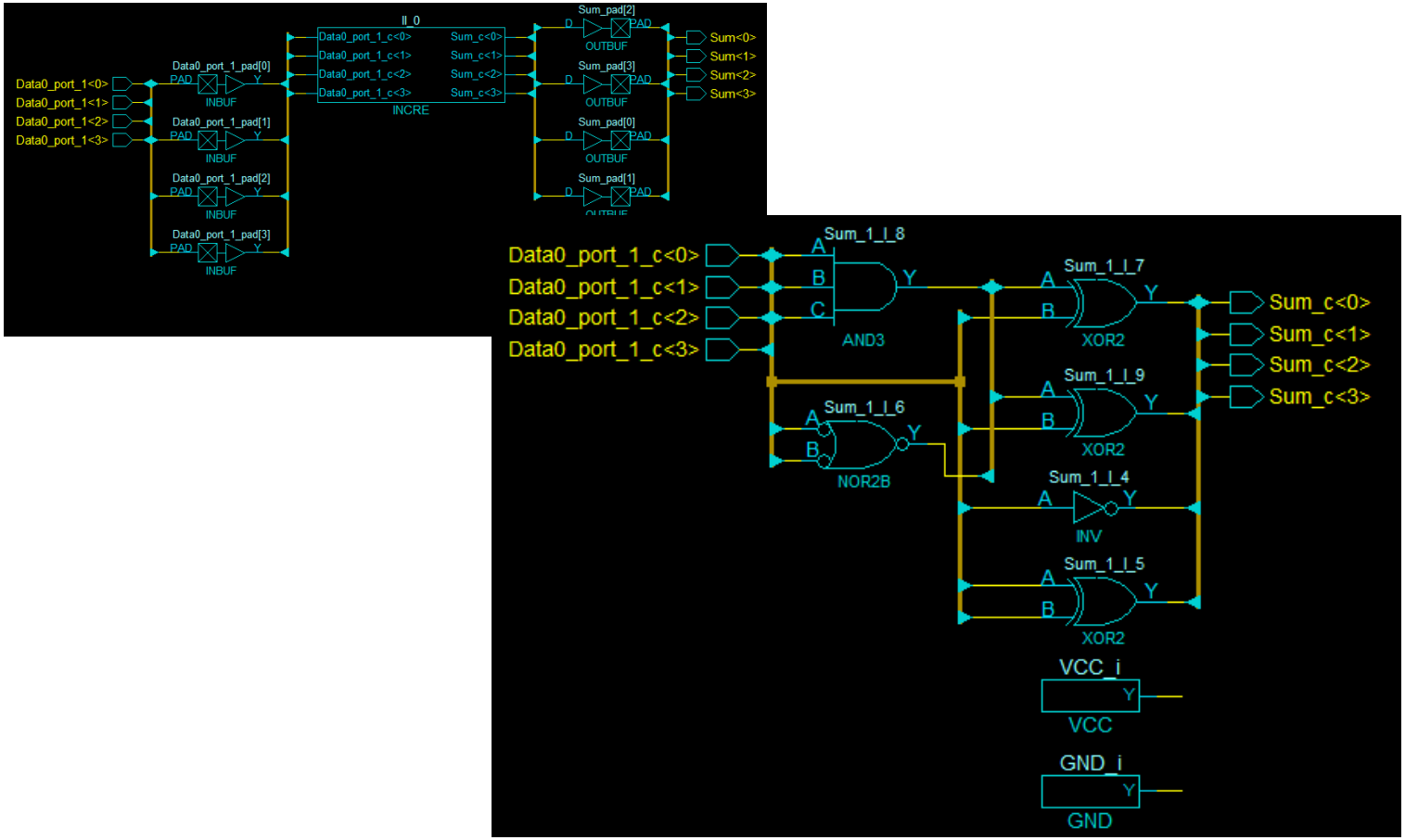
- 입력 받은 data에 +1 하는 logic

- Smart Design을 이용한 incrementer 블록 합성결과



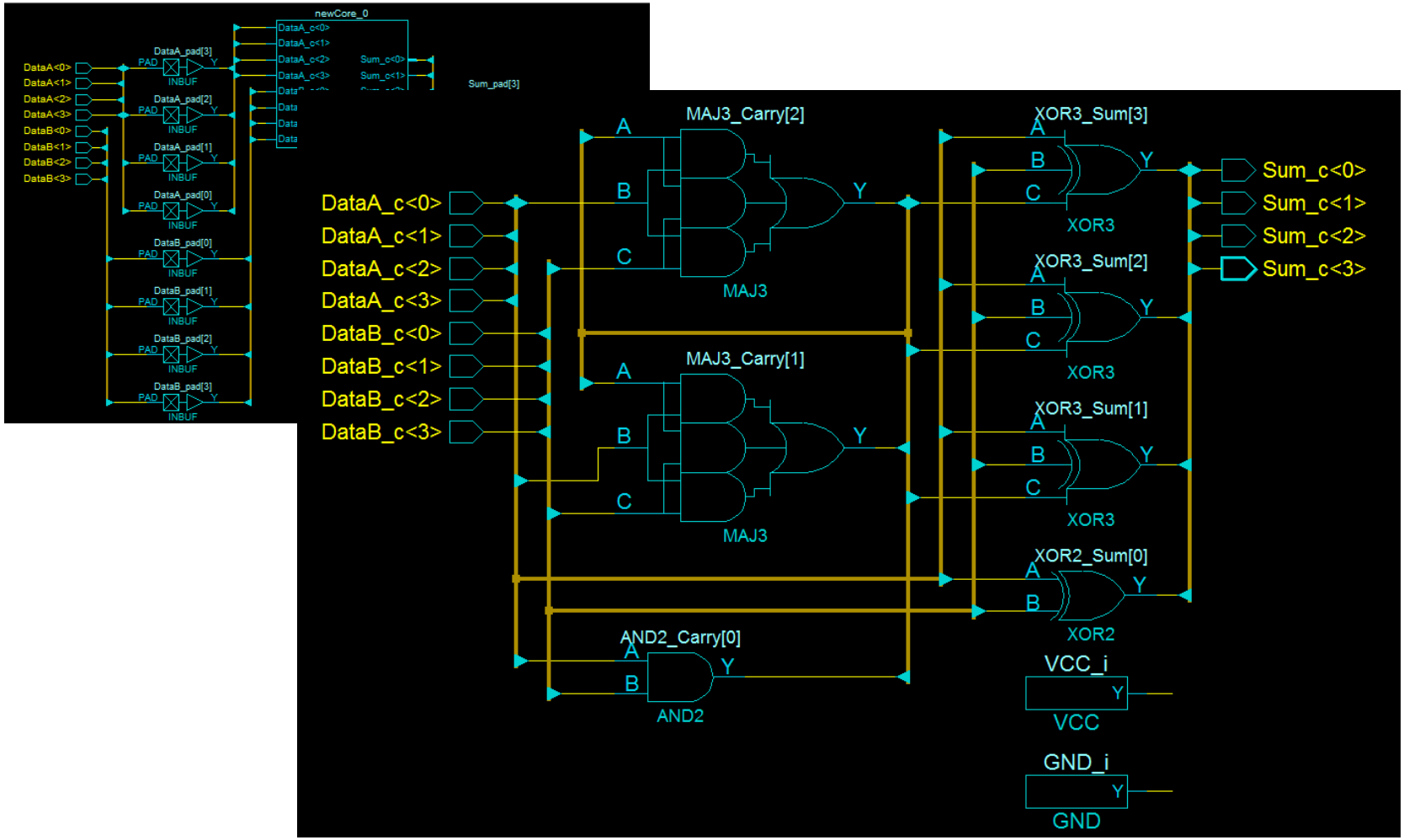
Incrementer : Verilog

- Verilog로 작성한 incrementer 블록 합성결과



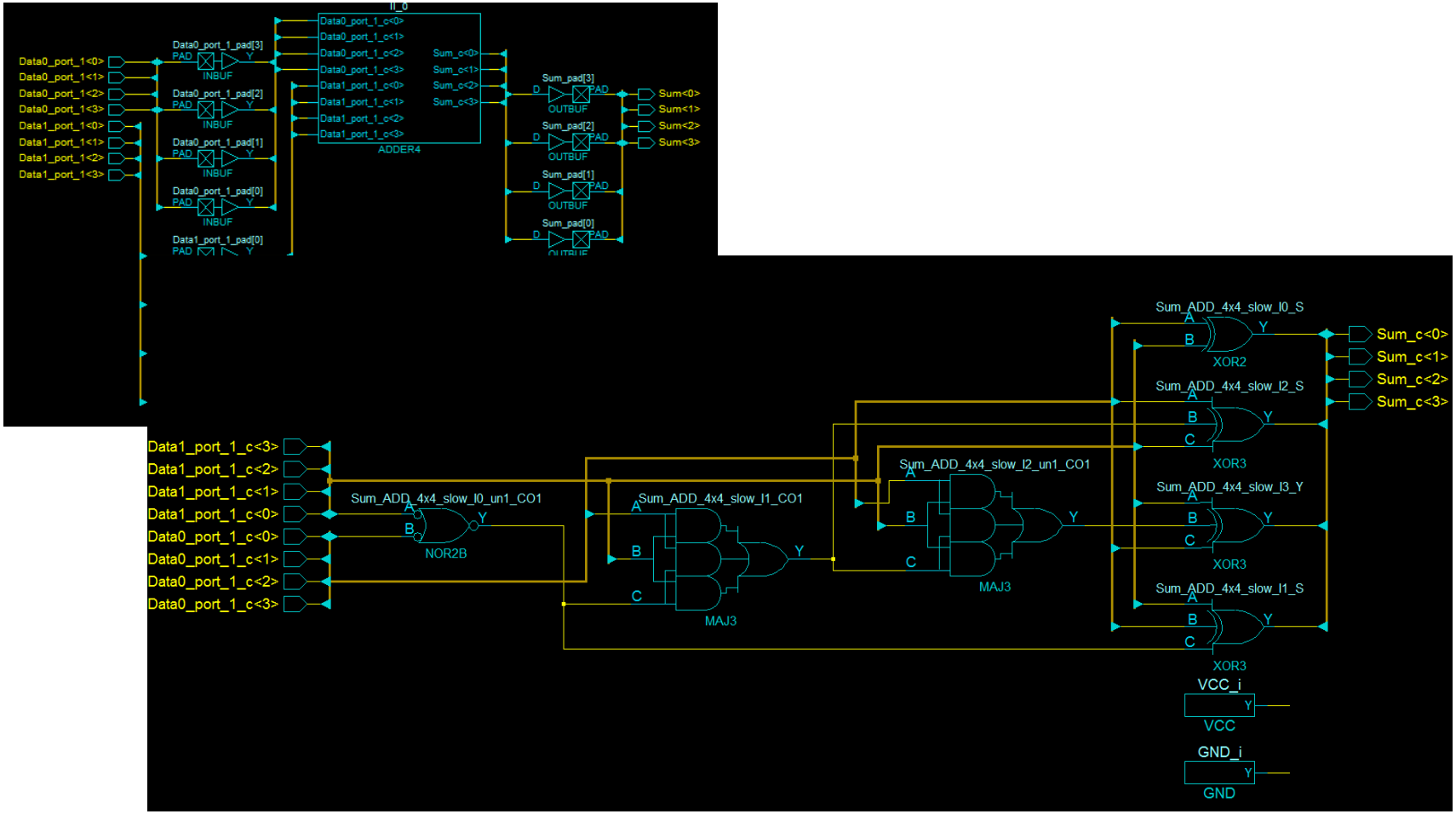
Adder 4 bit : Smart Design

- Smart Design을 이용한 adder 블록 합성결과



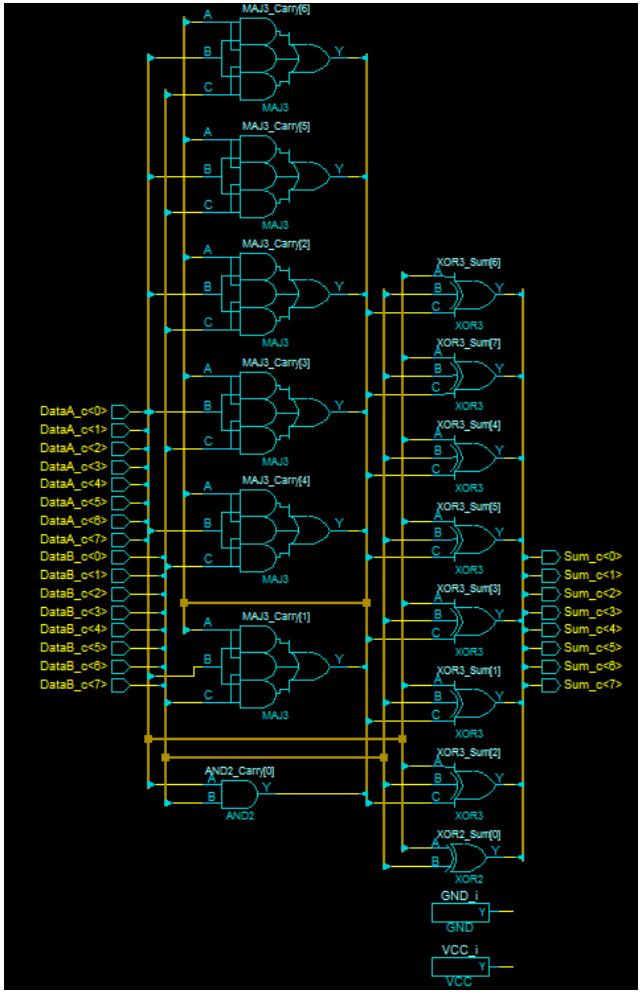
Adder 4 bit : Verilog

- Verilog로 작성한 adder 블록 합성결과



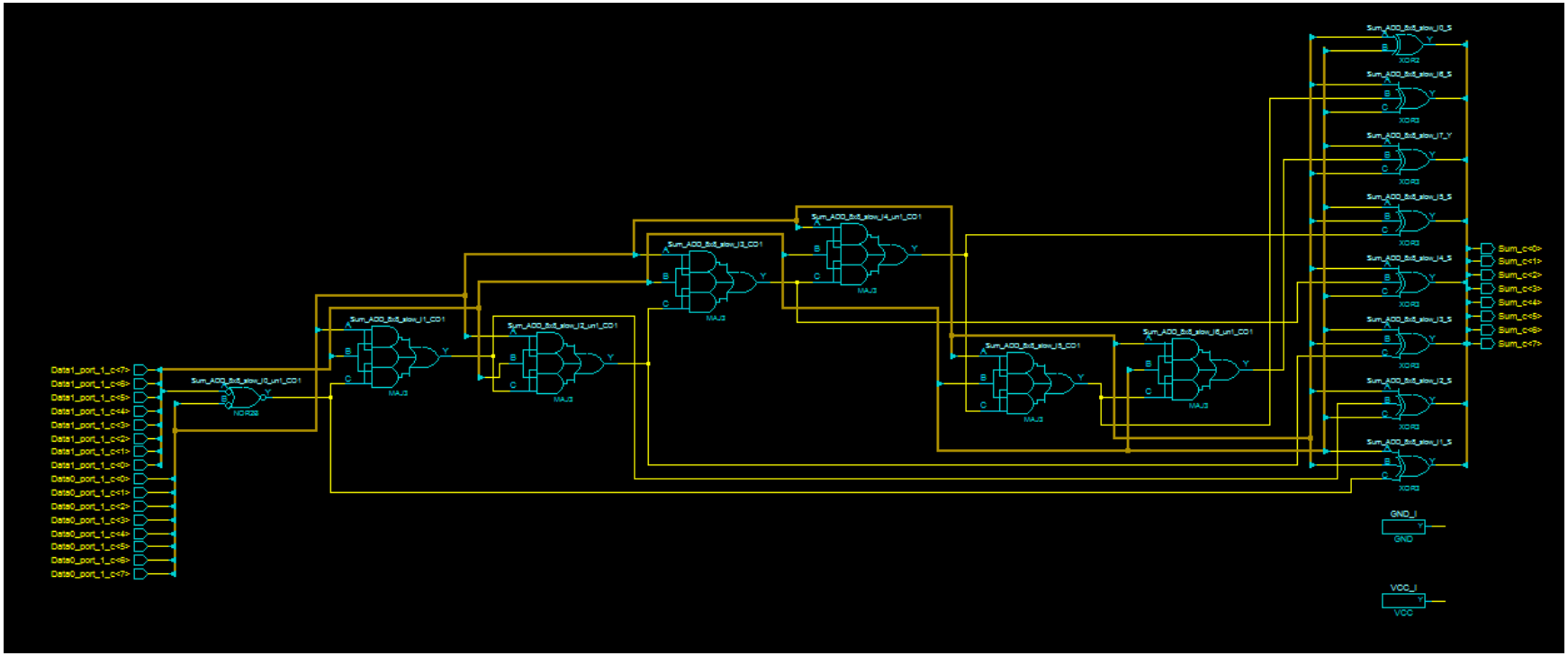
Adder 8 bit : Smart Design

- Smart Design을 이용한 adder 블록 합성결과



Adder 8 bit : Verilog

- Verilog로 작성한 adder 블록 합성결과



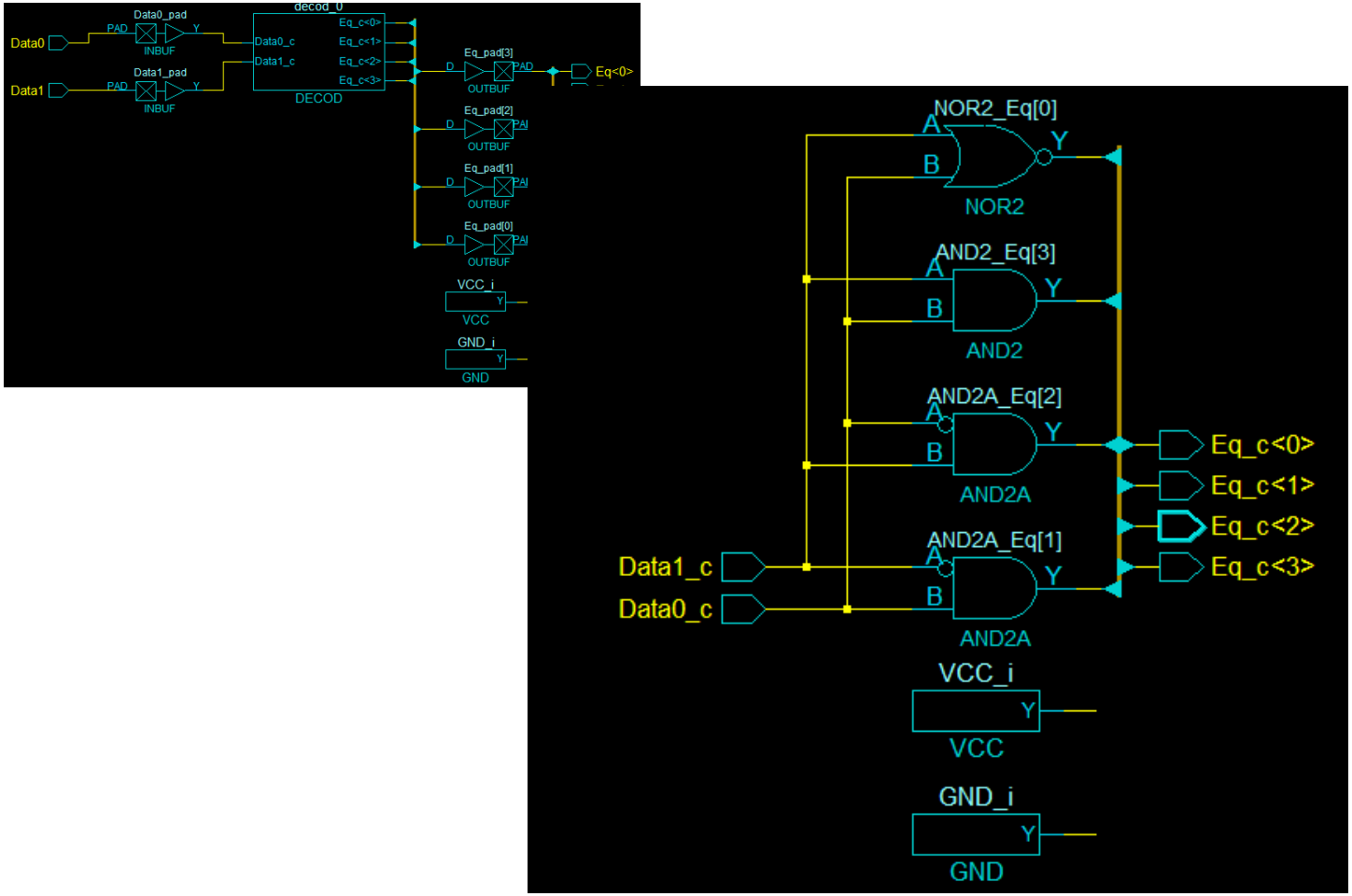
Decoder 4 bit : Smart Design

- 입력 조합을 신호로 발생

C	B	A	O_0	O_1	O_2	O_3	O_4	O_5	O_6	O_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

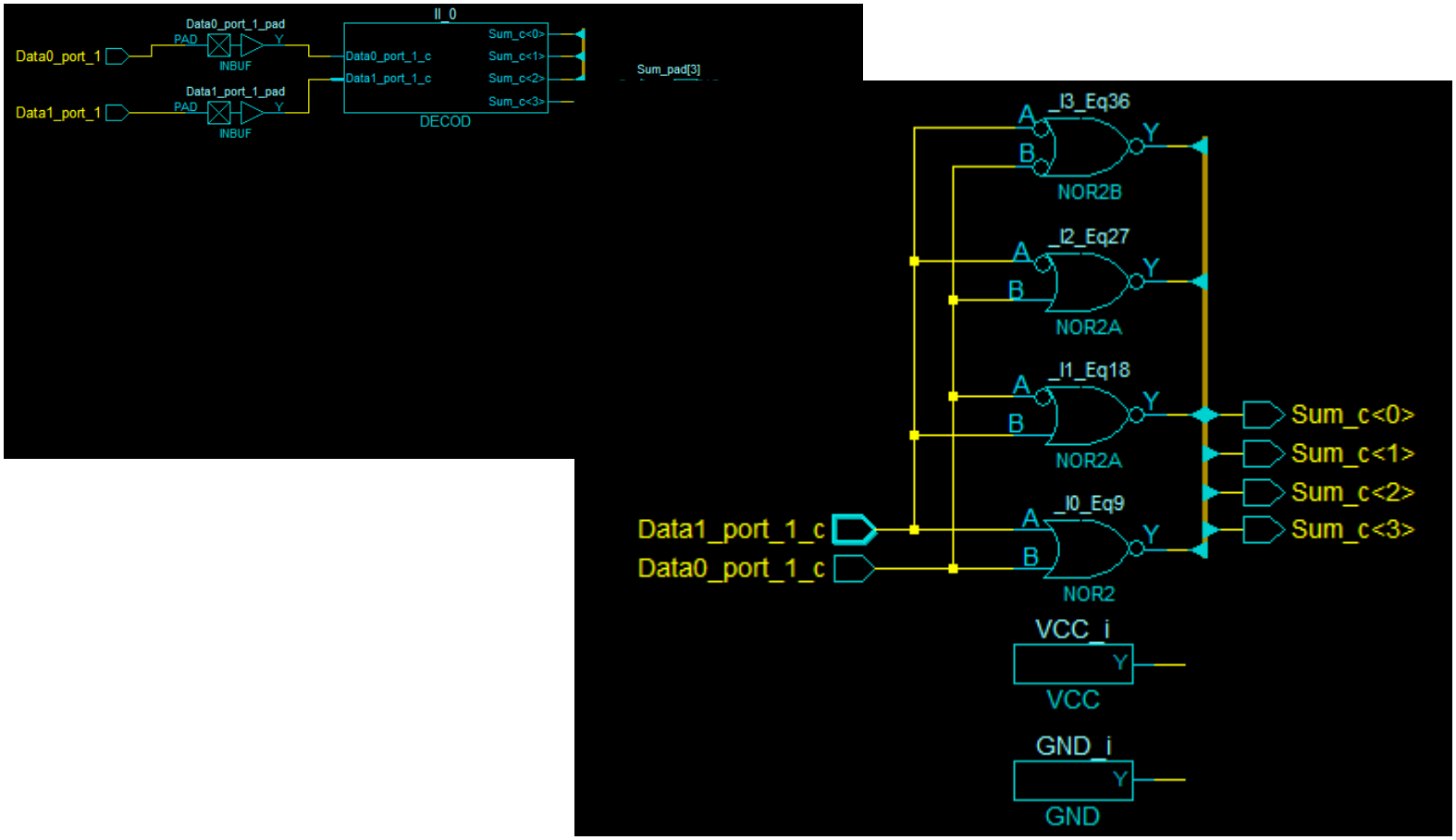
Decoder 4 bit : Smart Design

- Smart Design을 이용한 decoder 블록 합성결과



Decoder 4 bit : Verilog

- Verilog로 작성한 decoder 블록 합성결과



Conclusion

- 실험 결과로 보았을 때
  - Smart Design의 블록들은 structural verilog로 추정되는 코드를 이용하여 블록들을 미리배치시켜 가지고 있음
  - Verilog의 경우에는 합성 단계에서 맞춰서 게이트로 변환 시키는 것으로 보임

- Basic block 몇몇에 대해 실험해 본 결과 사용하지 않는다 라고 생각 할 수 있음
  - 하지만 Smart Design을 이용하여 디자인을 하기 위해선 -> basic block을 사용 할 수 밖에 없음 (adder, subtractor 와 같은 기본 사칙연산 기능 제공)

Logic	Smart Design	Verilog
Decoder	6 개 (NOT 2, AND 3, NOR 1)	8 개 (NOT 4, NOR 4)
4 bit Adder	11 개 (AND 7, XOR3 3, XOR2 1)	15 개 (AND 6, OR 2, XOR3 3, XOR2 1, NOT 2, NOR 1)
Incrementer	6 개 (AND3 1, AND 1, XOR 3, NOT 1)	9 개 (AND3 1, XOR 3, NOT 3, NOR 2)

현재 실험한 로직들의 Gate 사용 개수 예제

Smart Design & FBD

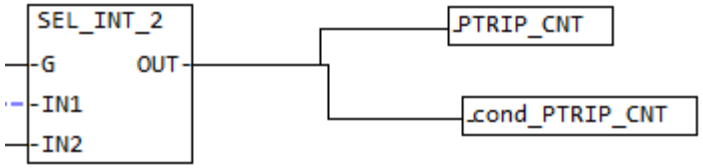
- Smart Design을 이용한 RTL 생성
  - Smart Design에서 제공하는 block 을 이용하여 디자인 작성
    - FBD와 유사하게 블록의 연결들을 통해 디자인
    - 작성 시에 블록으로 구성된 기능의 경우 코드 import
  - 'Generate Component' 기능을 통해 RTL (Verilog) 로 변환
    - 작성한 디자인과 동일한 Verilog 생성

- FBD를 이용한 RTL 생성
  - FBD를 이용하여 디자인
  - FBDtoVerilog 를 이용하여 RTL 로 변환 (in NuDE)
    - FBD의 각 block에 해당하는 Verilog 코드의 정확성 검증 여부 확인 (LIB.v)

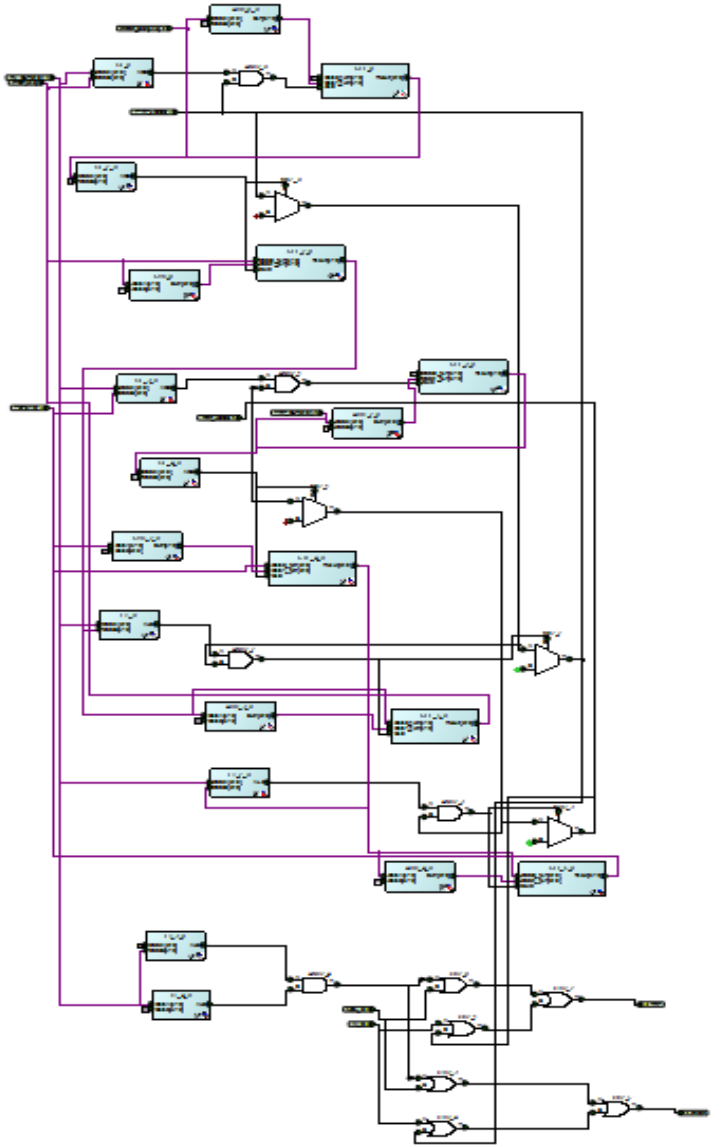
- 구조상의 동작 흐름은 동일
  - 예제로 FIX_RISING 로직을 Smart Design으로 작성

Smart Design & FBD

- 1차 작성 – 실패?
  - Bit 수로만 자료형 표현
  - 빠른 디자인을 위해서는 Basic block 사용이 거의 필수적임
  - 반복되는 상수 사용 문제 → FBD의 local variable?
  - Feedback variable 표현 문제 → bi-directional port?
    - bi-directional port를 이용한 feedback variable 표현 시 이후의 게이트 합성 X
    - (현재 FBD 의 connector/continuation 부분)



  - 클럭 표시의 문제



TR-1025243

TR-1025243

- **Independently verified**
  - **During the design process**
  - **Evaluated and approved 된 other computer program applications 의 결과와 비교하거나, hand calculation 의 결과와 비교해서 검증**
  - **Using alternative methods 또한 가능**
    - **Like simulation, testing**
- **Adversely impact the ability of an SSC to perform its safety function**
  - **도구의 failure (incorrect result, incorrect programmatic calculations 등) 가 SSCs 의 safety function 수행에 영향을 미치는가**
    - **Design 도구 : 잘못된 output은 의도하지 않는 기능 수행 가능**
    - **Analysis 도구 : 잘못된 output에 따라 유저가 다른 수정 가능**
    - **But 예제를 보니 analysis 도구의 경우 prototype 에 대해 functional testing을 수행함으로 영향을 미치지 않는다 라고 생각 하기도 함**

TR-1025243 Examples

- **Computer program used to perform pipe stress calculation and analysis**
  - **Safety Classification**
    - **Is the computer program integral to a safety-related SSC?**
      - **NO : pipe stress 계산 및 분석에 사용**
    - **Can the computer program impact safety-related SSCs?**
      - **YES : pipe stress 계산 데이터가 reactor coolant system에 영향을 미침**
    - **Will the computer program be used to design or analyze SSCs in a way that could impact SSC safety functions?**
      - **YES : design에 사용**
    - **Will the result derived using the computer program be independently verified for every use?**
      - **NO**
    - **Could failure of the computer program adversely impact the ability of the SSC (that is, the pump) to perform safety functions?**
      - **YES : software 의 fault 가 design, selection, layout of pipe supports 에 직접 영향을 미치는 결과를 도출함**
  - **Safety-related 로 분류**

TR-1025243 Examples

- Computer program used in the design of a safety-related pump
  - Safety-related pump performs safety functions during and after an earthquake
  - Safety Classification
    - Is the computer program integral to a safety-related SSC?
      - NO : 디자인 용도의 computer program
    - Can the computer program impact safety-related SSCs?
      - YES : 디자인 결과가 safety-related pump에 영향을 미칠 수 있음
    - Will the computer program be used to design or analyze SSCs in a way that could impact SSC safety functions?
      - YES : design에 사용
    - Will the result derived using the computer program be independently verified for every use?
      - NO : 결과는 바로 verification 할 수 없음
    - Could failure of the computer program adversely impact the ability of the SSC (that is, the pump) to perform safety functions?
      - NO : 일반적으로 prototype을 만들어 design verification 을 하기 때문에 영향을 미치지 않음
    - Will the computer program be used to assess the ability of SSCs to perform their safety-related function(s)?
      - NO : 디자인 도구
    - Will the computer program be used to monitor operation and control functions of SSCs?
      - NO : 디자인 도구
  - Non-safety-related 로 분류

TR-1025243 Examples

- **Computer program used to perform seismic analysis of components in safety-related systems**
  - **Safety Classification**
    - **Is the computer program integral to a safety-related SSC?**
      - **No**
    - **Can the computer program impact safety-related SSCs?**
      - **YES : component들의 지진을 견디는 능력 등을 확인 하는 용도**
    - **Will the computer program be used to design or analyze SSCs in a way that could impact SSC safety functions?**
      - **YES : analysis 에 사용**
    - **Will the result derived using the computer program be independently verified for every use?**
      - **YES : alternative calculation이 가능함**
  - **Non-safety-related 로 분류**

TR-1025243 적용 예제

- **Smart Design**
  - **Safety Classification**
    - **Is the computer program integral to a safety-related SSC?**
      - **NO : design에 사용**
    - **Can the computer program impact safety-related SSCs?**
      - **YES : safety-related SSCs 디자인하는데 사용 가능**
    - **Will the computer program be used to design or analyze SSCs in a way that could impact SSC safety functions?**
      - **YES : design에 사용하고 SSC 의 safety function 에 영향 가능**
    - **Will the result derived using the computer program be independently verified for every use?**
      - **YES : Simulation, output 인 Verilog 와 design 비교 가능**

TR-1025243 적용 예제

- **Simulator**
  - **Safety Classification**
    - **Is the computer program integral to a safety-related SSC?**
      - **NO : Analysis 용도**
    - **Can the computer program impact safety-related SSCs?**
      - **YES : 분석**
    - **Will the computer program be used to design or analyze SSCs in a way that could impact SSC safety functions?**
      - **YES : 분석 결과에 따라 design에 영향을 줄 수 있음**
    - **Will the result derived using the computer program be independently verified for every use?**
      - **NO : 비교할 approved 된 다른 도구 없음, 검증할 다른 방법 없음**
    - **Could failure of the computer program adversely impact the ability of the SSC (that is, the pump) to perform safety functions?**
      - **NO : 최종적으로 FPGA hardware functional testing 수행 및 SW/HW Co-simulation수행**
    - **Will the computer program be used to assess the ability of SSCs to perform their safety-related function(s)?**
      - **YES : 능력을 평가하는데 사용**

ISL-ESRD-TR-14-04 TASK 3

Task 3 Report : Technical Basis for RG

- Task 3 : RG 개정 및 새로운 RG 개발 시 Software tool의 review및 approval (acceptance) process 에 대해 고려할 점들

- Software Tool : Supporting or used in design, development, testing, review, analysis, or maintenance

- Future regulatory guidance 에서 고려할 내용을 11개 요소에 대해서 설명

- 필요에 따라 software tool 의 개발 및 인증 과정을 3 가지로 구분
  - Developed under 10CFR50 Appendix B QA requirements
  - Does not developed under 10CFR50 Appendix B QA requirements
    - Commercial grade item, SW dedication
  - Developed using high-quality life cycle approach of RTCA DO-330
    - Software Tool Qualification Considerations : 항공 분야의 software tool 인증 표준

Task 3 Report : Technical Basis for RG

- 2.1 Software Lifecycle Processes
- 2.2 Software Tool Categories
  - modeling, analysis, simulation 도구를 포함해서 software tool 의 category를 분류할 필요가 있다
- 2.3 Software and Software Tool Integrity Levels
  - Software tool의 SIL분류는 이렇게 하는 걸 고려해야 한다 인데...
- 2.4 Software Tool Planning
  - 수정에 대한 고려 사항 : software 개발 life cycle의 여러 plan (V&V plan, QA plan 등)시 software tool 에 대한 내용이 포함 되어야 함
  - Software life cycle planning 등에 software tool의 변화나, 버전업 등에 대해 planning 이 세워져야 함
- 2.5 Software Tool Selection and Use
  - Safety system 개발에 Software tool 을 선택하고 사용할 때 고려해야 할 사항들
- 2.6 Software Tool Documentation
  - Software Tool에 대해 문서화 되어 있어야 하는 사항들에 대한 고려
- 2.7 Software Tool Development, Qualification, and Dedication
  - System 개발에 사용하기 위해 Software tool을 개발하거나, qualification 하거나 dedication 할 때 고려해야 할 사항들
- 2.8 Software Tool Quality Assurance
  - Software tool의 qualify assurance 를 확인 할 때 고려되어야 하는 사항들
- 2.9 Software Tool Training
- 2.10 Software Tool Configuration Management
- 2.11 Software Tool Review, Approval, and Acceptance Criteria

Task 3 Report : Technical Basis for RG

- 2.7 Software Tool Development, Qualification, and Dedication
  - 17. Future RG might consider endorsing, adopting or adapting the methods of TR-10 25243 for the dedication of commercial-grade design and analysis tools
  - 18. Future RG might consider adapting the TR-102348
  - 와 같이 dedication을 할 경우 고려되어야 할 사항에 대해 언급

- 2.11 Software Tool Review, Approval, and Acceptance Criteria
  - Phase of commercial dedication
    - Tool qualification을 위해TR-1025243에서 명시하고 있는 critical characteristics에 대해 정의해야 한다
    - CASE tool, development tool의 critical characteristics 는 extensive testing 과 performance history를 통해 검증 되어야 한다
    - Dedication 과정에서 나타나는 모든 critical characteristics는 software tool의 dedication에 필수적임을 확인해야 한다
    - 등
  - Commercial dedication phase에서는 dedication 을 받아들이기 위해 고려되어야 할 사항을 언급

The END

END

Conclusion

NOT1	inverter	102 ps
IBUF1	inverting tri-state buffer	122 ps
NAND2	not(AND2)	112 ps
NOR2	not(OR2)	122 ps
AND2	2 input AND	142 ps
OR2	2 input OR	152 ps
NAND3	3 input NAND	132 ps
NOR3	3 input NOR	142 ps
AND3	3 input AND	162 ps
OR3	3 input OR	172 ps
NAND4	4 input NAND	152 ps
NOR4	4 input NOR	162 ps
AND4	4 input AND	182 ps
OR4	4 input OR	192 ps
NAND5	5 input NAND	172 ps
NOR5	5 input NOR	182 ps
AND5	5 input AND	202 ps
OR5	5 input OR	212 ps
NAND6	6 input NAND	192 ps
NOR6	6 input NOR	202 ps
AND6	6 input AND	222 ps
OR6	6 input OR	232 ps
NAND7	7 input NAND	212 ps
NOR7	7 input NOR	222 ps
AND7	7 input AND	242 ps
OR7	7 input OR	252 ps
NAND8	8 input NAND	232 ps
NOR8	8 input NOR	242 ps
AND8	8 input AND	262 ps
OR8	8 input OR	272 ps
XOR2	2 input XOR	172 ps
XNOR2	2 input XNOR	172 ps

LINTING

Linting

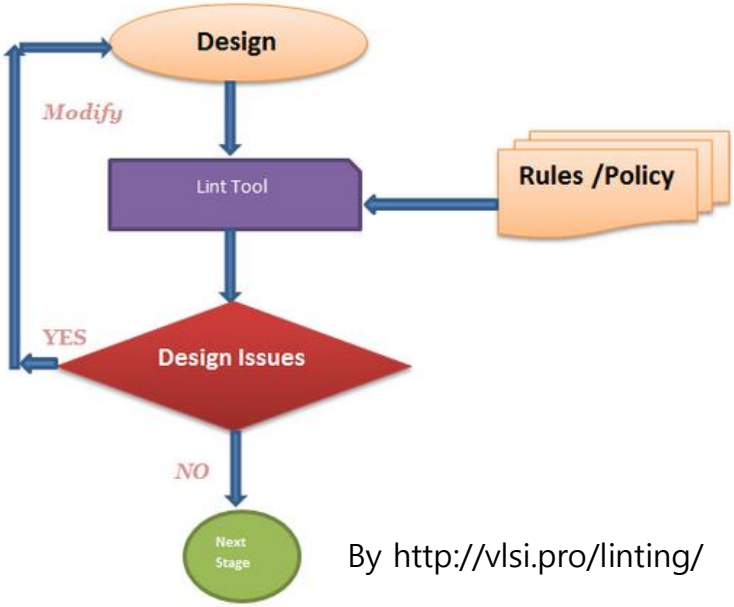
- Linter program checks static errors or potential errors and coding style guideline violations
  - variables being used before being set
  - division by zero
  - conditions that are constant
  - calculations whose result is likely to be outside the range of values representable in the type used
  - Mixed lananguage
  - Coding style check
  - Etc

- 일반적으로 FPGA 개발에서는 RTL design에 적용됨

- IEC 60880 같은 safety life cycle 에서 static analysis 때 적용
  - 합성 도구와는 별개로 독립적으로 적용
  - 합성 도구에서 syntax check 는 수행

RTL Linting

- Synthesis 이전에 Linting을 수행하면?
  - Static error의 가능성을 발견하고, false alarm 이 존재할 가능성이 있지만 사용자가 미리 수정 할 수 있음
- 컴파일러 verification 의 간접 방법으로 사용?
  - 합성 이전의 문제 발견을 위해 코드 체크하는 검증 방법으로 사용
- 상용 Linter program을 Safety-critical system 개발에 사용한다면
  - 시뮬레이터와 같은 분석 도구 수준의 dedication이 필요할 것으로 생각 (TR-1025243)



By http://vlsi.pro/linting/

Linting

- Level 1 : Rule Checking , Coding Style Checking - 코드의 syntax를 분석
- Level 2 : OOO analysis - 코드를 CFG/DFG 등의 형태로 변환하여 각종 품질 관련 분석을 수행
- Level 3 : static analysis - 코드에 대한 정적분석을 통해 실제 execution 時 발행할 수 있는 중요한 오류(Divide by Zero, Null Pointer 등)를 예측

Linting – SPYGLASS lint

- 제품 별로 다른 레벨의 linting 지원

SpyGlass Lint Features	Lint	Lint Turbo	Advanced Lint
Structural Lint (Syntax, Semantics, Synthesis, Simulation etc.)	👍	👍	
Functional Lint (Bit-width, Signed-unsigned assignment etc.)		👍	
Physical Lint (Large-Mux, Large Fanout / Fanin Cones, Synthesized/unsynthesized memories etc.)		👍	
Hierarchical Flow with abstract models (Top-down/Bottom-up)		👍	
FSM analysis to identify deadlocked and unreachable states			👍
Synchronous FIFO analysis to report underflow and overflow			👍
Dead code analysis			👍
Cyclomatic complexity analysis			👍

Linting – SPYGLASS lint

- FSM 관련 규칙
  - Unreachable or deadlock state
  - Dead transitions
  - Asynchronous inputs
  - FSM metrics : number of states/transitions, number of inputs/outputs, depth of states 등은 FSM 의 verifiability 에 영향을 미침
  - FSM design style
- Level 1 : FSM metrics, FSM design style
- Level 2 :
- Level 3 :

FUNCTIONAL SAFETY

IEC 61508 Functional Safety

- 전자, 전기 시스템의 기능 안전을 위한 표준
  - 특정 분야에 구애 받지 않은 전반적인 요구사항
  - E/E/PE safety-related system의 기능 안전성을 달성하기 위해 필요한 관리 및 기술적 활동을 명시
- Safety Life Cycle
  - 기능 안전 달성을 위한 활동을 체계적으로 관리하기 위해 제안 및 채택
  - 7.5 전체 안전 요구사항 : Hazard & Risk analysis를 통해 E/E/PE safety-related system, 기타 기술 안전 관련 시스템, 외부 리스크 감소 설비에 대하여 안전기능 요구사항 및 완전무결성 요구사항의 측면에서 전체 안전 요구사항에 대한 명세서를 개발함으로써 기능 안전성을 달성
    - 각 위험원에 대해 요구되는 기능안전성을 확보하기 위해서 필요한 안전기능들이 명시 되어야 함
    - 리스크 감소 측면에서, 안전무결성 요구사항 (SIL) 이 각 안전기능에 대해 명시되어야 한다
- 61508-3 requirements 중 소프트웨어 개발
  - 7.4.2.11 표준화된 소프트웨어 또는 기존에 개발된 소프트웨어가 설계단계에서 활용된다면, 해당 소프트웨어를 분명하게 파악해야 한다. 소프트웨어 안전 요구사항 명세를 만족하는데 대한 소프트웨어 적합성은 그 근거가 제시 되어야 한다.
  - 개발에 사용되는 언어, 컴파일러, 형상관리 도구, V&V 도구 세트는 SIL 에 따라 선택 되어야 한다
  - SIL 수준에 따라 확증 인증서를 보유한 번역기/컴파일러를 가져야 함
  - 충족되지 못하면 그 타당성을 문서화 되어야 함
  - 부록으로 정적분석의 몇몇 항목에 대해 표로 표시하고 있음

Functional Safety Certification

- **SIL(Safety Integrity Level) : 제품의 안전 기능에 요구되는 신뢰도 수준**
  - **Using Performance Measures, probability of the safety function operation**

Safety-Integrity Level (SIL)	High demand rate (dangerous failures/hr)	Low demand rate (Probability of failure on demand)
4	$\geq 10^{-9}$ to $< 10^{-8}$	$\geq 10^{-5}$ to $< 10^{-4}$
3	$\geq 10^{-8}$ to $< 10^{-7}$	$\geq 10^{-4}$ to $< 10^{-3}$
2	$\geq 10^{-7}$ to $< 10^{-6}$	$\geq 10^{-3}$ to $< 10^{-2}$
1	$\geq 10^{-6}$ to $< 10^{-5}$	$\geq 10^{-2}$ to $< 10^{-1}$

Functional Safety Certification

- **Standards for providing the requirements for the functional safety system**
  - **IEC 61508 : functional safety of electrical, electronic, and programmable electronic equipment**
  - **IEC 61513 : for NPP system**
  - **IEC 60880 : for category A software**
  - **IEC 62138 : for category A software**
  - **ISO 26262 : for automotive**

- **This product receives IEC-61508 SIL2 certification**
  - 내압방폭 구조로서 폭발 위험지역에 설치하여 가연성, CO2, CO, N2O가스를 연속적으로 감지

적외선 타입 가스감지기

CERTIFICATE

CERTIFICATE NO FS/71/220/14/0030 PAGE 1/1

'LICENCE HOLDER'
GASTRON CO.,LTD
18-8, DOGEUMDANJI1(IL)-GIL, SANGROK-GU, ANSAN-SI, GYEONGGI-DO, KOREA

'MANUFACTURING PLANT'
GASTRON CO.,LTD
18-8, DOGEUMDANJI1(IL)-GIL, SANGROK-GU, ANSAN-SI, GYEONGGI-DO, KOREA

PROJECT NO/-ID	LICENSED TEST MARK	CERT. REPORT NO.
F1MW		F1MW0003

Tested according to	IEC 61508: 2010
Certified product(s)	Infrared Type Gas Detector
Model(s)	GIR-3000
Technical Data and Parameter	Type B device with HFT=0 for the particular Safety Functions : Suitable for safety related systems in low demand mode up to and including SIL 2
Specific Requirements	The certificate is for type approval and based on voluntary tests. Any changes to the design, materials, components or processing may require repetition of some of the qualification tests in order to retain type approval. The certification report is an integral part of this certificate. All requirements and constraints of the current valid revision of this report shall be met.

Certification Body
for Functional Safety
SGS-TÜV Saar GmbH

Munich, 2014-02-11

Marcus Rau

SafeTI™ 소프트웨어 개발 프로세스, ISO 26262 및 IEC 61508 "기능 안전" 표준에서 ASIL D 및 SIL 3 레벨 인증 취득

2015-02-12 오전 10:26:38 · 편집부

Hercules™ MCU 소프트웨어 컴포넌트를 위한 새로운 SafeTI 인증 지원 패키지로 "기능 안전성" 개발 및 인증 지원

TI(대표이사 켄트 전)는 자사의 SafeTI™ "기능 안전" 소프트웨어 개발 프로세스가 ISO 26262 및 IEC 61508 준수 소프트웨어 컴포넌트 개발에 적합하다고 인증 받았음을 발표했다. 이 프로세스는 품질 및 안정성 규격에 대한 적합성을 평가하는 국제 공인 독립 평가 기관인 TÜV NORD(독일기술검사협회)에서 심사하였다.

더불어 TI는 인증된 소프트웨어 개발 프로세서를 기반으로 새로운 SafeTI 인증 지원 패키지(CSP, Compliance Support Package)를 개발하였으며, 현재 Hercules™ 마이크로컨트롤러(MCU) 소프트웨어 컴포넌트에 사용되고 있다. CSP는 Hercules 소프트웨어를 이용하는 고객들이 자사의 최종 시스템의 "기능 안전성" 인증을 더욱 수월하게 달성할 수 있도록 하기 위해 개발되었다.

SafeTI CSP는 정적 및 동적 분석 테스트 결과, 규격 적합성에 대한 코드 추적가능성(code traceability to requirements), 코드 커버리지, 코드 품질 지수 등을 포함하고 있다. 고객들은 이 CSP를 이용함으로써 소프트웨어 검증 작업에 대한 수고를 줄이고, 최종 시스템의 "기능 안전성" 인증을 보다 쉽게 달성할 수 있다.

TI는 CSP 개발에 LDRA(Liverpool Data Research Associates) 소프트웨어 분석 툴 수트를 이용하고 있다. 또한, 이들 CSP는 LDRAunit®을 활용한 테스트 자동화 유닛(Test Automation Unit)을 포함하며 고객들은 그들의 환경에 이 유닛 레벨 테스트 사례를 채실행할 수 있다. 이들 CSP는 HALCoGen(Hardware Abstraction Layer Code Generator) 디바이스 드라이버와 Hercules MCU의 SafeTI 진단 라이브러리에 이용할 수 있다.

이러한 TÜV 인증 SafeTI "기능 안전" 소프트웨어 개발 프로세스와 이를 적용한 SafeTI CSP, 그리고 최근 출시된 인증 Hercules TMS57011x/12x 및 RM46x MCU는 향후 고객들이 기능 안전 애플리케이션을 간편하게 개발할 수 있도록 도와주는 포괄적인 SafeTI 설계 패키지로, TI의 고객 지원을 위한 노력을 잘 설명해주고 있다.

공급 시기

TI의 HALCoGen 디바이스 드라이버와 SafeTI Hercules 진단 라이브러리에 이 CSP를 이용함으로써 고객들은 제품의 출시 시간을 단축하고 검증 작업에 대한 수고를 줄이며, 소프트웨어 안전 작업을 간소화할 수 있다. 현재 이들 CSP 평가판 뿐만 아니라 1인용 또는 멀티용 정식 라이선스도 이용 가능하다.

IEC 60880 고려사항

- Software tool 선택은 (개발에 사용되는) 60880의 1~12 chapter의 요구사항을 만족하거나 15 chapter의 assessment를 만족해야 함 => dedication 관점과 비슷하게 사용됨
  - 60880의 전체적인 내용과 dedication에서 사용하고 있는 그런 critical characteristics를 통한 criteria와 잘 매핑을 시켜보면서 두개의 연관성에 대해 고려해 보고 생각 할 수 있을 것으로 판단됨

- 적용되어야 하는 assessment수준은 tool의 type에 따라 달라짐
  - 1. compiler, translator
  - 2. verification tools
  - 3. os
  - 4. development support systems (e.g. word processor?)
  - 5. version control tool (e.g. svn)
  - 각각의 분류에 따른 수준에 대한 언급 부족

- Compiler, translator의 optimization
  - Should be avoided
  - 사용 한다면, 컴파일 결과에 대해 test, verification, validation 반드시 수행

COMMON POSITION EXAMPLE

Common Position

- **Licensing of safety critical software for nuclear reactors**
  - **It is *"Common position of international nuclear regulators and authorized technical support organisations"***
  - **Common technical positions on a set of important licensing issues**

- **Task force, which contains 7 countries, establish documents for licensing issues of safety critical software (Licensing issues of safety critical software for nuclear reactors)**
  - **Belgium, Germany, Canada, Spain, United Kingdom, Sweden, Finland**

- **In the later, the U.S. NRC has participated in the meetings of the task force**

This document should neither be considered as a standard, nor as a new set of European regulations, nor as a common subset of national regulations, nor as a replacement for national policies. It is the account, as complete as possible, of a common technical agreement among

- **National regulations may have additional requirements or different requirements, but hopefully in the end no essential divergence with the common positions.**

Common Position

- This documents consists of involved issues, common positions, recommended practices about each licensing issues

- It provides 23 issues about licensing
  - 1.1 Safety Demonstration
  - 1.2 System Classes, Function Categories and Graded Requirements for Software
  - 1.3 Reference Standards
  - 1.4 Pre-existing Software (PSW)
  - 1.5 Tools
  - 1.6 Organizational Requirements
  - 1.7 Software Quality Assurance Program and Plan
  - 1.8 Security
  - 1.9 Formal Methods
  - 1.10 Independent Assessment
  - 1.11 Graded Requirements for Safety Related Systems (New and Pre-existing Software)
  - 1.12 Software Design Diversity
  - 1.13 Software Reliability
  - 1.14 Use of Operating Experience
  - 1.15 Smart Sensors and Actuators

  - 2.1 Computer Based System Requirements
  - 2.2 Computer System Architecture and Design
  - 2.3 Software Requirements, Architecture and Design
  - 2.4 Software Implementation
  - 2.5 Verification
  - 2.6 Validation and Commissioning
  - 2.7 Change Control and Configuration Management
  - 2.8 Operational Requirements

1.4 Pre-existing Software – Issues Involved

- **Issues involved**
  - **A set of issues about licensing**

- **Issues about 1.4 pre-existing software**
  - **The functional behavior and non-functional qualities of the PSW is often not clearly specified and documented**
  - **It is not certain that developing under safety life cycle like IEC 60880**
  - **The operational experience of the PSW are not often enough to compensate for the lack of knowledge on the PSW (information about product and development process)**

1.4 Pre-existing Software – Common Position

- **Common Position**
  - **A set of common positions on the basis for licensing and evidence which should be sought by task forces**

- **Common positions about 1.4 pre-existing software**
  - **The functions that have to be performed by PSW, shall be clearly and unambiguously specified**
  - **The code version of PSW shall be clearly identified**
  - **The interfaces (the user or other software) shall be clearly identified**
  - **The PSW shall have been developed and maintained according to QA standards and software development process**
  - **Documentation and source code shall be available if modification**
  - **Documents of quality assurance plan and development process shall be available**
  - **Conditions for accepting**
    - **Verify the functions performed by the PSW about requirements specification**
    - **The PSW functions shall be validated by testing**
  - **Defects which are found during validation shall be analyzed**

1.4 Pre-existing Software – Recommended Practices

- **Recommended Practices**
  - **Consensus on best design and licensing recommended practices by task forces**

- **Recommended Practices about 1.4 pre-existing software**
  - **Operational experience may be regarded as evidence to validation or verification**
  - Configuration of the PSW;
    - Functions used;
    - Types and characteristics of input signals, including the ranges and, if needed, rates of change;
    - User interfaces;
    - Number of systems.
  - Demand rate and operating time data should include:
    - Elapsed time since first start-up;
    - Elapsed time since last release of the PSW;
    - Elapsed time since last severe error (if any);
    - Elapsed time since last error report (if any);
    - Types and number of demands exercised on the PSW.
  - Error reports should include:
    - Descriptions and dates of errors, severity;
    - Descriptions of fixes.
  - Release history should include:
    - Dates and identifications of releases;
    - Descriptions of faults fixed, functional modifications or extensions;
    - Pending problems.