**Assignment #1** – Design of Adders

Sae-Byuk Shin Young-Sun Yoon Jin-Bae Kim  
Supercomputing Laboratory @Yonsei (The 3rd Engineering 532)

October, 2016

|  |  |
| --- | --- |
| ASSIGNED: | 2016-10-15 |
| DUE: | 2016-11-06 (23:59) |
|  | Submission after due date: **DO NOT ALLOW**  You are to do this assignment **ALONE!** |

1. 개요

본 프로젝트는 컴퓨터 구조의 핵심을 이루는 구성요소 중 하나인 연산 회로(Arithmetic circuit) 구현을 직접 수행하고자 한다. 특히, 기본 논리 gate를 활용하여 연산 회로 중 주요 역할을 수행하는 고속 Adder들을 설계해보고, Adder의 종류 별로 특징들을 비교하는 것을 목표로 한다.

1.1. 과제 목적  
❖ 다양한 종류의 Adder들을 구현해보고, design-tradeoffs에 대해 고려해볼 수 있는 기회를 갖는다.

1.2. 과제 내용  
❖ Carry-Skip(Bypass)/Carry-Lookahead Adder를 구현한다.  
❖ 각 Adder들을 4/8-bit operand case에 대해 구현하고 비교한다.

1.3. 제출 기한  
❖ 2016년 11월 06일 오후 11:59까지(YSCEC 서버 시간 기준. 마감 기한 이후로부터의 과제 제출은 허용되지 않음)

1.4. 제출 방법  
❖ 보고서 파일(pdf, word, odt, hwp)과 Verilog HDL 코드(.v) 파일이 포함된 프로젝트 폴더의 압축파일  
 (\*.zip, \*.7z, \*.tar 확장자의 압축 파일만 허용함. 알집을 비롯한 유사 압축 프로그램을 사용하는 것은 절대 권장하지 않는다.)  
 (권장하지 않은 압축 프로그램의 사용 및 압축/문서 형식 등으로 인해 발생하는 불이익은 본인이 모두 감수해야 한다.)

1.5. 제한 사항  
❖ Altera-ModelSim 10.1d  
 (Assignment #0에서 설치를 진행하였던 버전의 ModelSim을 사용한다.)  
❖ Verilog (VHDL이 아님!)  
 (Hardware description language로는 오직 Verilog HDL만 인정한다. 문법 표준은 Altera-ModelSim 10.1d의 Compiler를 기준.)  
❖ 본 프로젝트는 개인 과제임  
 (타인의 코드를 그대로 복제하여 제출하거나, 변수 명만 교체하여 제출하는 행위는 엄히 금함)  
 (과제 대행 서비스 의뢰와 같은 부정 행위 적발 시, 본 프로젝트는 0점 처리되며 추가적인 징계 및 불이익이 있을 수 있음)  
❖ 인터넷 문서, 논문, 혹은 특정 도서를 참고하였을 경우, 보고서 마지막의 Reference 항목에 밝혀야 함  
 (Text/Image/Table 등 발췌 및 인용하였던 대상에 상관없이 모두 출처를 밝히시오.)  
 (보고서에 인용을 표기하였더라도, 그대로 해당 문서의 글들을 통째로 복사하여 사용해도 된다는 것은 아닙니다. 이를 유념하세요.)  
 (동일 내용으로 인해 학생간 표절 의심 발생시, 이를 명확히 해소할 수 있는 근거가 됩니다.)

2. Adders

Adder의 종류에는 여러 가지가 있다. 이번 프로젝트에서는 아래의 Adder들 중, design tradeoff들에 대해서 느껴볼 수 있는 2가지 Adder들을 구현하고 비교해보는 것을 목표로 한다. 각 Adder들은 추구하는 바(성능/전력소모/가격/면적 등)에서 차이를 보이며, 때로는 두 종류 이상의 Adder의 주요 feature들을 조합하여 hybrid adder로 재설계하기도 한다.

1. Carry-Skip(Bypass) Adder
2. Carry-Lookahead Adders

이번 프로젝트에서는 위의 목록에 있는 Adder들을 구현해야 한다.

2.1. 과제 수행 내역  
(프로젝트를 위해 구현 및 수행해야 할 세부 목록은 아래와 같다.)  
❖ [프로젝트 전반]   
 (1) 배포되는 프로젝트 파일을 open하여 프로젝트를 진행한다.  
 i. File-open 항목에서 파일 형식을 Project Files (\*.mpf)로 변경한 뒤 midterm.mpf 파일을 open한다.  
 (2) 위에서 제시한 두 가지 Adder들을 Verilog HDL로 구현해야 한다.  
 i. 각 Adder 종류 별로 2-input 4/8-bit Adder의 경우에 대해서 구현해야 한다.  
 (3) Verilog로 구현될 모든 모듈들은 **Gate level description**의 형태로 구현되어야 한다.  
 i. 조교에 의해 미리 정의된 gate들을 활용하여 구현해야 한다. 기본 gate는 조교가 정의하여 배포한다.  
 예) mxor2 xor\_1(S, A, B)  
 (4) 배포된 gate 파일에 정의된 gate delay를 활용하여 최대지연 및 전체 회로의 성능을 측정한다.  
 i. 성능의 비교를 위해서 필수적인 부분이므로, 반드시 배포된 gate를 사용해야 한다.  
 주의) 모든 Verilog 코드 파일 최 상단에 **`timescale 1ps/1ps**와 같이 timescale을 정의해야 한다.  
 ii. 이를 준수하지 않을 시, 매우 치명적인 감점 사유가 될 수 있다.  
 iii. 제공되는 gate들은 아래의 [표 1] Gate Delay 표를 따른다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Gate |  | Delay(ps) |  |  |  |
| NOT1 | Inverter | 50ps |  |  |  |
| NAND2 | 2 input NAND | 100ps | NAND3 | 3 input NAND | 150ps |
| NOR2 | 2 input NOR | 100ps | NOR3 | 3 input NOR | 150ps |
| AND2 | 2 input AND | 120ps | AND3 | 3 input AND | 170ps |
| OR2 | 2 input OR | 120ps | OR3 | 3 input OR | 170ps |
| XOR2 | 2 input XOR | 140ps | XOR3 | 3 input XOR | 250ps |

[표 1] Gate Delay 표

(5) Testbench 모듈의 경우 배포된 Testbench 양식에 호환되도록 구성하되, 입력 스트림을 스스로 구성한다.  
 i. 각 Adder의 종류 별로 **각기 다른 8가지 정도의 입력**에 대해서 성능 측정을 해본다.  
 ii. **입력 스트림은 각 Adder의 최대 지연 시간을 보일 수 있는 연산의 경우를 포함하고 있어야 한다.**  
 (6) 프로젝트 전체를 압축하여 제출해야 하며, 재-컴파일이 가능한 코드를 포함하여야만 한다.  
 i. 채점 과정에서 다시 컴파일이 이루어지며, 컴파일 이후 동작이 불가할 시 점수가 부여되지 않는다.  
 ii. 제출 전, 꼭 올바른 코드 및 프로젝트를 압축하였는지 재차 확인하는 것을 권장한다.  
 (7) 주요 부분에 대해서는 주석을 통해 코드의 의미를 전달할 수 있도록 한다.  
 i. 모든 코드 라인에 대해서 주석을 요구하지는 않지만, 중요한 부분에 대해서는 주석을 달아야 한다.   
 (8) Adder의 종류 별로 Verilog HDL 코드 파일들(\*.v)을 분리해야 한다.  
 i. 전체 프로젝트를 하나의 모듈 파일로만 구현하는 것은 절대 권장하지 않는다.   
 (9) 가급적 최대지연시간을 최소화하는 방향으로 모듈을 설계하도록 한다.   
 i. 최대지연시간 및 게이트 사용 개수에 따라 가산점이 부여될 수 있다(최대 10%).  
  
❖ [보고서 작성]  
 (1) 프로젝트 개요  
 i. 프로젝트의 전반적인 사항에 대해 요약하여 서술한다.  
 ii. 각 Adder들의 비교 결과에 대해서도 정리하여 간단하게 서술한다.  
 (2) 프로젝트 배경 및 이론 요약  
 i. 본 프로젝트를 통해 설계한 Adder들의 작동 원리 및 설계 등에 대해서 서술한다.  
 (구현 해야 할 Adder들에 대해 충분히 조사한 후 여기에 서술하길 권장함)  
 (3) 프로젝트 구현 내용 및 성능 측정/비교  
 i. 실험할 2가지 Adder 에 대해 간단한 구조 설명.  
 ii. 각 Adder들을 4/8-bit case에 대해서 실험한 결과를 비교 (성능, 소자 사용 개수 등)  
 a. 성능에 대한 비교를 진행한 후, n-bit case에 대한 각 Adder들의 성능을 일반화 해본다.  
 (4) 결론  
 i. (3)을 통해 얻을 수 있는 결과 분석 및 프로젝트 전반적인 사항에 대한 정리/요약  
 (5) 참조(References)  
 i. 프로젝트 수행 및 보고서 작성 도중 참고한 사항들에 대해 출처 표기  
  
 다수의 n-bit case에 대해서 구현을 진행해야 하므로, 최대한 코드의 재활용도가 높고 가독성이 좋은 스타일로 각 Adder들을 구현하는 것을 권장한다.

3. 프로젝트 관련 추가 정보

❖ 과제에 대한 질문 사항은 가급적 YSCEC 학생 게시판을 활용하는 것을 권장한다.

❖ 과제의 세부적인 안내 사항이 수정될 예정이니, 수시로 YSCEC 공지를 확인하시길 바랍니다.

❖ 과제 수행 시 필요한 gate 및 delay 시간은 조교들이 작성하여 배포할 예정입니다.