세부과제 연차실적 보고서

연차실적 보고서

과제유형	1. 기초미래선도형 (O) 2. 공공인프라형 ( ) 3. 산업화형 ( )				
대과제명	초연결 통신 기초원천 기술 개발				
세부과제명	양자컴퓨팅 플랫폼 및 비용효율성 향상에 대한 연구개발				
세부과제 책임자	소속 및 부서	초연결통신연구소 양자창의연구실	직위 (직급)	실장 (선임연구원)	
	성명	최 병 수			
총연구기간	2015년 01월 01일 부터 2017년 12월 31일 까지 (36개월)				
당해연도 연구기간	2016년 01월 01일 부터 2016년 12월 31일 까지 (12개월) (2차년도)				
총연구비	정부출연금	1,571,438 천원	당해년연구비	정부출연금	535,719 천원
	민간부담금	23,750 천원		민간부담금	10,250 천원
	계	1,595,188천원		계	545,969 천원
참여인력(M/Y)	총 연 구 기 간	9 명 (9.0 M/Y)			
	당해연도 연구기간	6 명 (3.9 M/Y)			

참여기관	기관명	연구책임자	기관명	연구책임자
참여연구기관	고려대학교 산학협력단	허준		
	한양대학교 산학협력단	이진형		
위탁연구기관				
키워드 (6~10개)	양자컴퓨팅, 양자통신, 양자ICT, 양자정보, 운영체제, 플랫폼			

정부출연금사업 연차평가 보고서를 제출합니다.

2016 년 12 월 23 일

세부과제책임자 : 최 병 수 (인)
직 할 부 서 장 : 황 승 구 (인)

한국전자통신연구원장 귀하

목차

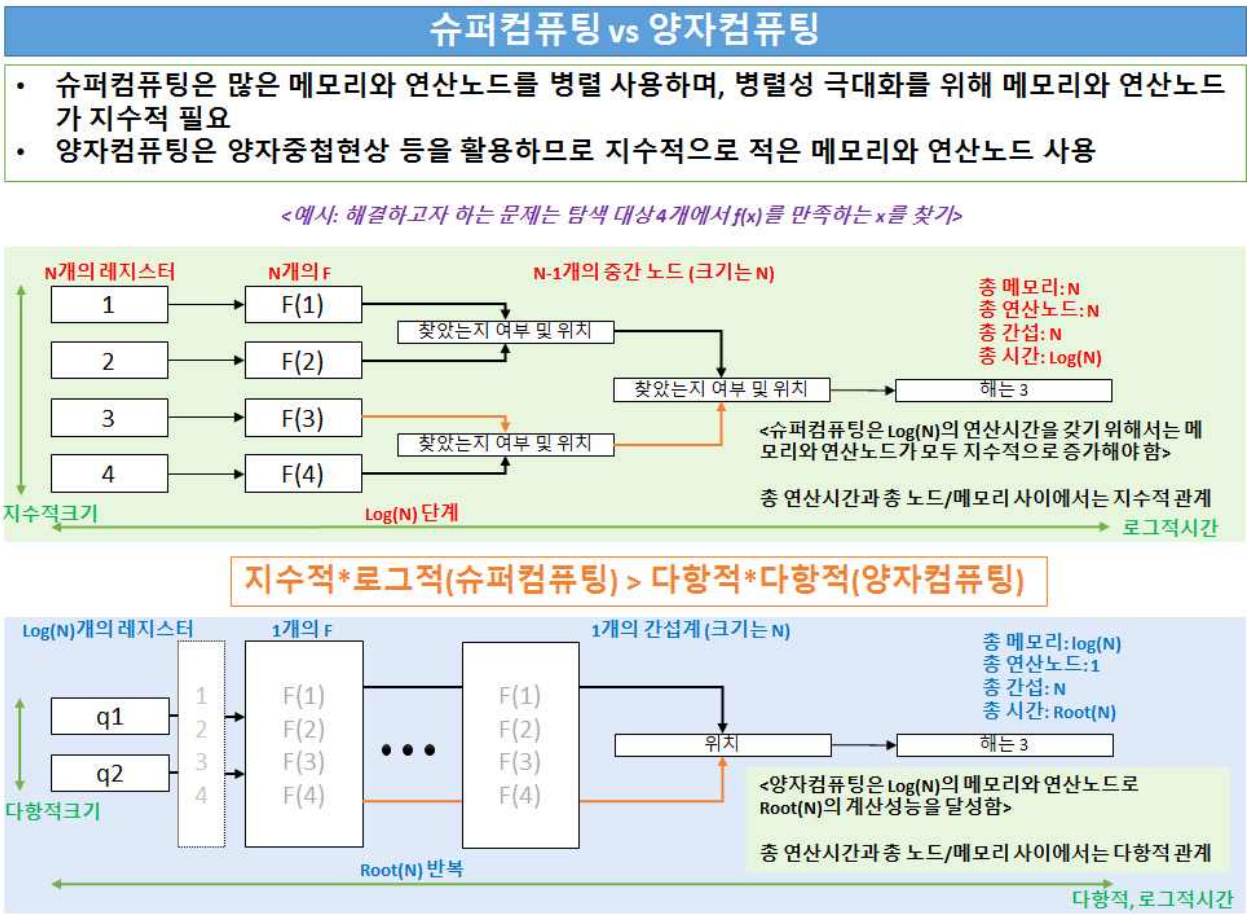
제1장 연구개발 배경 및 목표
1.1 양자컴퓨팅 개요 ---------- 3
1.2 양자컴퓨팅 주요 문제점 ---------- 5
1.3 연구개발 목표 ---------- 6
1.4 연차별 연구개발 목표 수준 ---------- 8
1.5 중장기적 연구개발 로드맵 ---------- 9
제2장 연구개발 내용 및 결과
2.1 연구개발 동향 ---------- 10
2.2 양자컴퓨팅 플랫폼 구축 현황 ---------- 11
2.3 비용효율성 향상 연구개발 현황 ---------- 36
2.4 연구개발 목표 달성정도 ---------- 47
제3장 향후 연구개발 계획
3.1 양자컴퓨팅 플랫폼 구축 ---------- 48
3.2 비용효율성 향상 ---------- 52
제4장 향후 활용 방안 ---------- 55
제5장 결론 및 제언 ---------- 57

연구개발 실적 ---------- 58
소요비용 ---------- 59
참고문헌 ---------- 61
약어표 ---------- 62

제1장 연구개발 배경 및 목표

1.1 양자컴퓨팅 개요

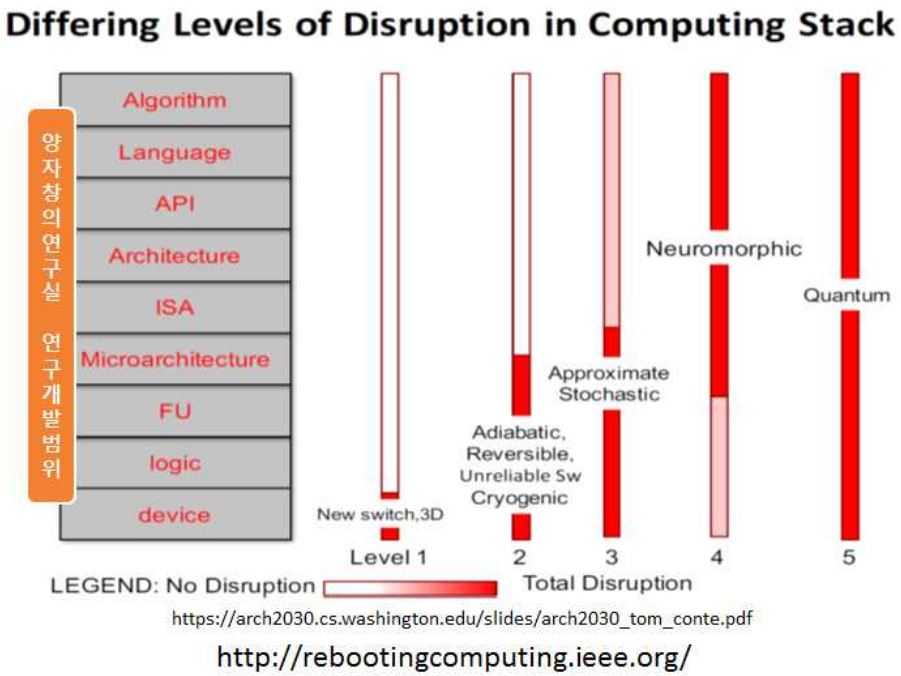
양자컴퓨팅은 양자역학적인 특성(간섭, 중첩, 얽힘, 관측붕괴등)을 갖는 정보에 기반한 컴퓨팅 방식을 의미한다. 이러한 양자컴퓨팅은 통상적으로 사용되는 컴퓨팅 방식과는 달리 표현하고자 하는 정보를 양자병렬적으로 표현할 수 있기 때문에, 필요한 메모리량과 이에 따른 계산량이 크게 감소할 수 있다. 아래 그림은 통상적으로 사용되는 슈퍼컴퓨팅과 양자컴퓨팅의 계산측면에서의 차이를 간략하게 설명하고 있다.



이러한 양자컴퓨팅은 예상되는 성능 향상 효과가 매우 높지만, 실제로 이를 구현하는 과정은 상당히 어렵다. 아래 그림은 양자컴퓨팅 구현 과정이 다른 비통상적 컴퓨팅 구현에 비해서 얼마나 많은 부분을 고려해야 하는지를 보여주고 있다. 양자정보를 실제로 구현하는 소자수준부터 이를 프로그래밍할 수 있는 프로그래밍 환경까지를 모두 새롭게 만들어야 하기 때문에, 여타 새로운 컴퓨팅 방법들에 비해서 상대적으로 구현의 난이도가 높은 편이다.

양자컴퓨팅 구현의 어려움

- **일반적인 차세대 컴퓨팅은 컴퓨팅의 일부를 변경하는 연구개발 필요 (Level1 ~ Level4)**
- **양자컴퓨팅은 양자정보 실현소자, 양자병렬성 (중첩, 간섭, 얽힘) 활용 알고리즘, 이를 지원하는 컴퓨터 명령어 등 대부분 영역에서 연구개발 필요 (Level5)**

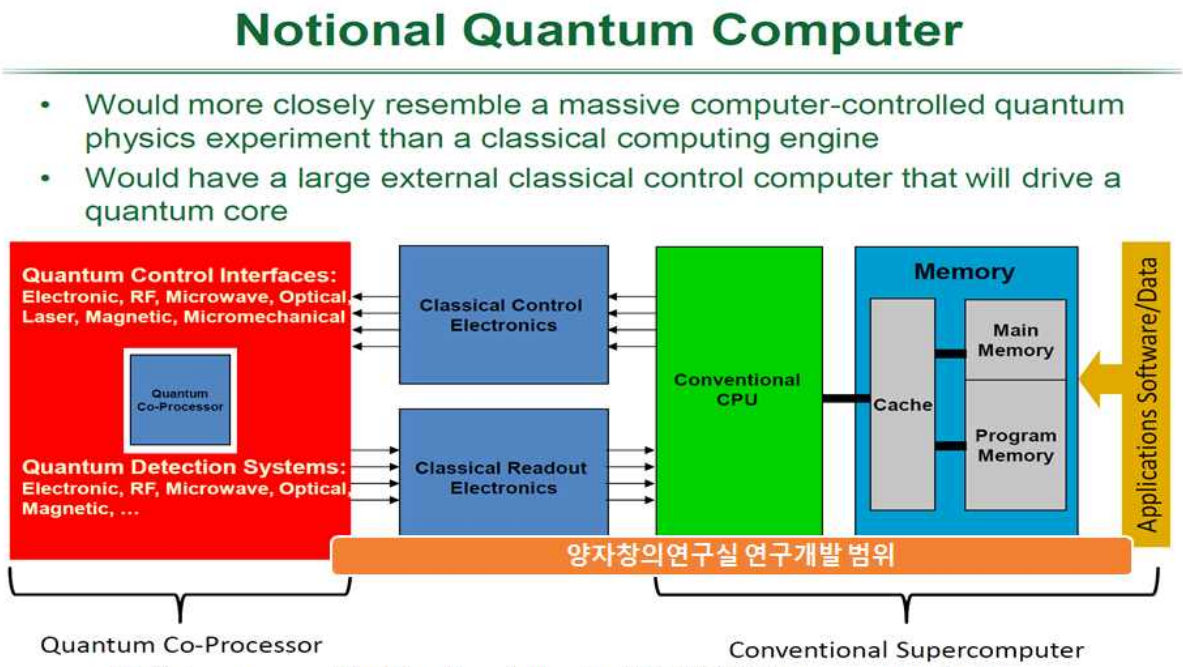


http://rebootingcomputing.ieee.org/

이러한 양자컴퓨팅을 실제로 구현하는 과정은 양자정보를 표현하고 이를 제어하는 기능을 포함하는 양자코프로세서 (quantum co-processor)와 이를 실제로 제어하여 양자알고리즘을 수행할 수 있도록 하는 슈퍼컴퓨팅시스템 (conventional supercomputer), 그리고 이들 사이를 상호 연결해 주는 연결장치 (interface)로 구성된다. 아래 그림은 이러한 과정을 개념적으로 도식화한 것이다. 이에 따라서, 양자컴퓨터를 구현한다는 것의 의미는, 양자정보를 표현하고 제어하는 양자정보장치와 더불어 이를 제어하기 위해서 사용되는 슈퍼컴퓨팅 시스템까지를 포함한다.

양자컴퓨팅 개념도

- **양자컴퓨팅은 기존 슈퍼컴퓨팅에서 초고속 연산을 담당하는 co-processor로 우선 활용 예상**
  ※ D-Wave는 아날로그 (오류보정, 결함허용 방식 미적용, 낮은 신뢰도) 1세대 양자컴퓨팅 모델임
  - (Quantum Co-Processor) 양자정보를 구현한 소자와 이를 제어하는 양자장치. ALU, 연산Lib 구현불가
  - (Conventional Supercomputer) Quantum Co-Processor에서 ALU, 연산Lib등을 동작시키는 운영체제 탑재
  - (Conventional-Quantum Interface) Quantum Co-Processor와 Conventional Supercomputer 연결장치



http://science.energy.gov/~/media/ascr/ascac/pdf/meetings/201604/2016-0405-ascac-quantum-02.pdf

1.2 양자컴퓨팅 주요 문제점

1.2.1 물리적 큐빗과 게이트의 높은 오류율

이러한 양자컴퓨팅 시스템을 구현하는 과정은 개념적으로 간단하지만, 실질적으로는 많은 문제점이 있다. 이 과정에서 가장 심각한 문제는 양자정보를 제어하고 유지하는 과정에서의 물리적 오류율이 상당히 높다는 점이다. 이는, 양자컴퓨팅에서 요구하는 양자알고리즘의 최소 단계를 양자정보가 지원하지 못하거나, 양자게이트의 오류율이 높아서 최종 연산결과의 신뢰도가 낮아지는 문제점을 야기한다. 이에 따라서, 비로 많은 수의 물리적 큐빗을 확보하더라도 일반 사용자가 기대하는 성능을 달성하기 어렵게 된다.

1.2.2 오류보정, 결함허용 방식을 이용한 물리적 큐빗과 게이트의 높은 오류율 해결

이러한 문제점을 해결하기 위해서는 다수의 물리적 큐빗을 사용하여 전체적으로 오류율이 낮은 논리적 큐빗을 만들고, 이에 기반하여 논리적 만능 게이트를 구현하는 결함허용연산방식이 적용되어야 한다. 특히, 이 과정에서 결함허용적 오류보정이 성공하는데 필요한 기본 조건으로써 물리적 큐빗과 게이트의 오류율이 일정값보다 낮아지면 되기 때문에, 실제로 양자정보소자 측면에서는 기술적 목표치가 유한 값으로 확정되어 있다.

1.3 연구개발 목표

○ 연구개발 목표 1: 양자컴퓨팅 플랫폼 구축

(1) 배경

양자컴퓨팅 플랫폼은 주어진 임의의 양자알고리즘을 주어진 임의의 양자정보소자를 이용하여 구동하고, 이에 따른 예상 성능 등을 분석하는 소프트웨어 시스템을 의미한다. 또한, 이러한 플랫폼을 이용하여 다양한 양자 컴퓨터 구조, 양자정보소자, 오류보정-결함허용방식, 알고리즘, 컴파일방식 등을 평가할 수 있다. 또한 주어진 양자컴퓨터 시스템이 수행 가능한 양자컴퓨팅의 활용문제의 범위 등을 파악할 수 있다. 아울러, 이러한 양자컴퓨팅 플랫폼은 양자HW모듈들에 대한 제어시스템으로 활용되어 양자컴퓨터를 실제로 구동하는데 사용된다.

(2) 목표

본 연구에서는 다양한 알고리즘을 다양한 양자정보소자시스템에서 구동할 수 있도록 하는 기능과 양자컴퓨팅 구동에 따른 예상 성능을 파악하는 기능을 갖는 양자컴퓨팅 플랫폼을 구축한다.

○ 연구개발 목표 2: 양자컴퓨팅의 비용효율성 향상

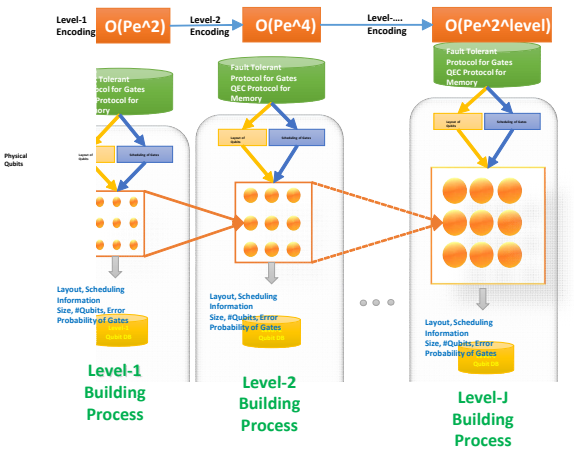
(1) 배경

오류보정과 결함허용 방식은 양자컴퓨팅의 연산결과 신뢰도를 높인다는 측면에서는 매우 중요한 핵심기술이긴 하지만, 이 과정에서 다수의 물리적 큐빗과 게이트를 사용해야 하므로 전체적으로 양자컴퓨팅 시스템의 크기와 논리적 게이드 별 구동 시간이 증가하는 문제점을 야기한다. 아래 그림은 오류보정, 결함허용 과정이 적용되어서 양자컴퓨팅의 계산 성능이 매우 나빠지는 사례를 보여주고 있다. 이에 더불어 양자컴퓨팅 시스템 측면에서는 논리적/물리적 큐빗간 상호 정보교환과정에서의 지연시간, 컴파일 과정에서의 소요 게이트의 증가 등과 같이 다양한 이유로 양자컴퓨팅의 계산시간이 매우 늘어난다. 따라서, 양자컴퓨팅의 실질적인 성능효과는 이론적 수준에서 예상하는 것과는 다르게 매우 낮은 계산성능을 갖게 된다 (아래 그림 참조). 따라서, 양자컴퓨팅이 이론적인 기대성능을 달성하기 위해서는 이와 같은 낮은 계산성능(본 연구에서는 낮은 비용효율성으로 정의)을 해결해야 한다.

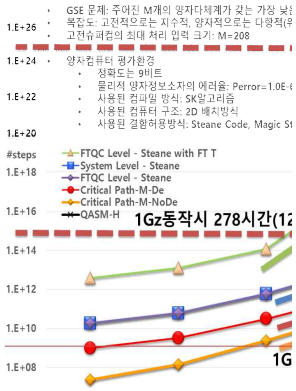
(2) 목표

본 연구에서는 이렇게 낮은 양자컴퓨팅의 비용효율성을 높이는 것을 핵심이론적 연구로 하며, 특히 결함허용방식, 컴파일방식, 컴퓨터구조방식, 논리적 빌딩블럭 설계방식 등에서 성능향상에 대한 연구를 진행한다.

- **아날로그 양자컴퓨팅은 슈퍼컴퓨팅 소자에 비해 10^{17}배 높은 잡음을 갖는 소자(큐빗과 게이트)와 이를 제어하는 장치로 구성 (특수 목적에만 동작)**
  ※ D-Wave가 상용화한 2000개+ 큐빗의 양자컴퓨팅에서 개별 큐빗 오류율은 10^{-3} (비트는 10^{-20} 오류율)
- **디지털 양자컴퓨팅은 아날로그 양자컴퓨팅의 잡음문제를 해결하는 결함허용기술 등과 범용으로 활용하기 위한 新운영체제 필요**
  ※ (결함허용기술) 큐빗의 잡음문제와 오류보정코드에 내재된 잡음문제로 야기되는 결함을 보정하는 기술
  ※ **(핵심연구1: 비용효율성 향상 기술) 결함허용 등에 따른 평균시간↑문제 최소화 기술**
  ※ **(핵심연구2: 플랫폼 기술) 핵심원천기술을 포함, 실시간으로 동작하는 운영체제 및 검증 개발환경 기술**



<결함허용방식과 비용효율성 >



< 결함허용기반 디지털 양자컴퓨팅에서의 낮은 계산성 문제 >

1.4 연차별 연구개발 목표 수준

본 연구에서는 위와 같은 연구개발 목표를 대상으로 하여, 아래 테이블과 같이 연차별 연구개발 목표를 따른다.

1년차인 2015년에는 플랫폼 구조에 대한 설계와 더불어 비용효율성과 관련한 기본 데이터를 추출하는 것을 목표로 한다.

2년차인 2016년(현재 년도)에는 양자컴퓨팅 플랫폼을 구성하는 핵심 구성요소들을 확보하고, 비용효율성과 관련하여서는 10배 정도 향상시키는 기술을 확보하는 것을 목표로 한다.

3년차인 2017년에는 기존 플랫폼의 비용등과 같은 특성분석용 수준을 확장하여, 양자컴퓨터를 실제로 구동하는 시스템 수준으로 확장하는 것을 목표로 한다. 비용효율성과 관련하여서는 최종적으로 1000배 이상을 향상시키는 것을 목표로 한다.

과제명	양자컴퓨팅 플랫폼 및 비용효율성 향상에 대한 연구개발		
연구개발 내용	1. 결함허용기반 양자컴퓨팅 플랫폼 구축 **(신뢰성, 안정성 확보 기술)** 2. 양자컴퓨팅의 비용효율성 향상을 통해서 양자컴퓨터의 계산성능 극대화 방법론 개발 **(양자계산 우수성, 활용성 확보 기술)**		
기간	2015.1~2017.12 (1단계 3년, 현재 2년차)		
참여인력	전일제 연구원: 2.75 M/Y (2015.9~2016.12)		
연차별 연구목표 및 내용	연도	연구목표	연구내용
	2015	플랫폼 설계	• 평가 대상 알고리즘 선정 • 평가 대상 오류보정코드 선정 • 평가 대상 소자기술 선정 • 평가 대상 컴퓨터구조 선정
		비용효율성 문제확인	• 비용특성분석
	2016	플랫폼 100% 완성	• 컴파일, 시스템, 임의 빌딩블럭, 기본 빌딩블럭 합성 부분 완성
		비용효율성 10배 향상	• 컴파일, 시스템, 빌딩블럭 비용 특성 분석 및 비용감소 방법론 제안 • 1가지 알고리즘 1가지 오류보정코드 1가지 소자기술 기준 비용분석
	2017	플랫폼 성능 개선	• 실시간 수준 운영체제로 확장
		비용효율성 추가 100배 향상	• 각 부분별 비용효율성 10배 향상 • 3가지 알고리즘 3가지 오류보정코드 3가지 소자기술 기준 비용분석

1.5 중장기적 연구개발 로드맵

양자컴퓨팅과 관련한 중장기 연국개발 로드맵은 크게 5년 단위로, 1단계 신뢰성 향상을 위한 결함허용 양자컴퓨팅 플랫폼 구현, 2단계 슈퍼컴을 뛰어넘는 양자컴퓨팅 시스템 구축, 3단계 양자계산학습 시스템, 4단계 양자인공두뇌 시스템 구축을 목표로 한다. 이러한 연구개발 계획은 중기적 측면에서 다시 3년 단위로 하여, 양자창의연구실 차원에서 아래 도표와 같이 연구개발 로드맵을 갖고 있다. 본 연구개발 과제는 이중에서 1단계 3년 (2015~2017)을 대상 기간으로 하며, 주요 연구개발 내용은 양자컴퓨팅 플랫폼 구축과 비용효율성 향상이다. 2단계에서는 양자컴퓨팅 플랫폼의 지속적 개선, 비용효율성의 지속적 향상, 보안성 추가, 연결성 추가 등을 목표로 한다. 3단계에서는 다수의 논리적 큐빗을 갖는 양자컴퓨팅 시스템 구축을 목표로 하며, 슈퍼컴퓨팅 대비 높은 계산성능 달성을 목적으로 한다.

양자컴퓨팅 중장기 연구개발 로드맵 (양자창의연구실)

	기능측면	2015~2020	2020~2025	2025~2030	2030~2035
SW	지능성			계산학습분야에서 고전적 접근 돌파 수준	최초의 인공두뇌 구현 수준
	활용성		계산과학분야에서 고전 슈퍼컴 돌파 수준	암호해독, SW검증 분야에서 고전 슈퍼컴 돌파 수준	계산집중형 문제에서 일반화 진행
	연결성		컴퓨터의 인터넷접속수준	빅데이터의 인터넷접속수준	우주밖의 인공위성수준
	보안성	보안연산기능	보안DB기능	인터넷과 접속과정에서 보안	
	계산성	10000배 향상 기술	10000배 향상 기술	계속적으로 향상	
HW	논리 큐빗수	1개 구현	100개 이상	빅데이터 구현 가능 수준	두뇌수준에서 요구량 수준
	시스템구조	단일소자기반	다중소자기반	3세대 큐빗 기반	
	크기	대규모	대규모	소형화 시작	소형화 계속
플랫폼형태		**SW는 임의알고리즘을 임의 정확도로 임의소자에서 구동 가능한 수준이지만, HW는 1개의 논리적 큐빗수준**	**충분한 수준의 논리적 큐빗을 사용할 수 있으며, 계산과학분야에서 고전컴퓨팅을 뛰어넘는 수준**	**계산학습에서 고전적 접근법을 완전히 뛰어넘는 수준**	**인공두뇌를 최초로 구현하는 수준**

<양자창의연구실 연구개발 로드맵>

	기능측면	2015~2017	2018~2020	2021~2023
SW	활용성	계산과학측면 기본 성능 분석	조기버젼평가 및 성능개선점 파악	계산과학에서 고전슈퍼컴 돌파 수준
	연결성			인터넷연결기능 추가
	보안성		보안연산기능 추가	보안DB기능추가
	계산성	1000배 향상 기술	1000배 향상 기술	100배 향상 기술
HW	논리 큐빗수	1개 구현	10개 구현	100개 이상
	시스템구조	단일소자기반	단일소자기반	다중소자기반
플랫폼형태		**임의알고리즘, 임의정확도, 임의소자**	**다수의 논리적 큐빗 실시간으로 구동**	**계산과학에서 고전슈퍼컴 돌파 수준**

제2장 연구개발 내용 및 결과

2.1 연구개발 동향

- **플랫폼 관련 연구개발 동향**

(1) 국내 연구개발 동향

- 양자 컴퓨터 구현 관련 연구개발이 본격화 되면서, 양자 컴퓨터 시스템의 구조(architecture)에 대한 논의가 활발하게 진행되고 있다. 양자 컴퓨터를 개발하는데 있어서 정보를 저장하고 처리하는 큐빗들을 어떤 구조상에 어떤 방식으로 배치할 것인가, 큐빗들 사이의 정보이동은 어떤 방식으로 구성할 것인가 등의 문제들에 대한 고민을 심도 있게 진행하고 있으나 현재까지 국내에서는 본 과제와 같이 소프트웨어 측면에서의 연구는 이루어지지 않고 있다. 특히 양자 컴파일러에 대한 연구는 본 연구진을 제외하고는 찾아보기 어렵다.
- 국내의 경우에는 전반적으로 양자정보구현과 관련한 하드웨어 부분에서의 연구가 주를 이루고 있다.

(2) 국외 연구개발 동향

- 국외에서는 미국의 국가기관, 대학 및 기업체를 중심으로 양자 컴퓨터 시스템에 대해 연구를 하고 있다. IARPA의 경우 LogiQ 사업을 통해 여러 연구기관들을 지원하고 있으며, UCSB, Stanford 연구팀에서 대규모 (large scale)양자 컴퓨터를 구축하기 위해서 필요한 여러 요소들에 대해서 연구하였고 (Metodi11, Metodi08, Metodi05a, Metodi05b), 또한 양자 컴퓨터의 구현을 계층적으로 나눠서 필요한 요소들에 대한 연구결과를 발표하였다. (Jonse12). 최근 들어 각광을 받고 있는 surface code 라는 양자 오류 정정 부호의 경우, 그 부호가 2d 격자 모양의 큐빗 구조상에 특화되어 있어서, surface code 기반으로 양자 컴퓨터를 개발하고 있는 IBM, Google 등에서 관련 연구 성과를 내고 있다. 마이크로소프트의 경우 Microsoft Research Station Q의 QuArC group이 Language-Integrated Quantum Operations: LIQUi|>라는 소프트웨어 시뮬레이션 플랫폼을 공개하였다[MS]. 이 프로그램은 양자 알고리즘의 작은 입력값(30큐빗 이하)에 대하여 시뮬레이션이 가능하다. 일례로 현재 27개 큐빗, 50만여 개의 게이트, 5일의 러닝타임을 사용하여 13 bit의 소인수분해 시뮬레이션이 가능함을 보인 적이 있다. IBM 연구팀은 올해 IBM Quantum Experience 라는 시스템을 구축하였다[IBM]. 이 시스템은 superconducting 큐빗을 이용하여 5 큐빗의 클라우드 기반의 양자 플랫폼으로 양자 어셈블리어인 QASM 파일을 입력으로 사용하여 시뮬레이션 할 수 있다.

- 비용효율성 향상 관련 연구개발 동향

(1) 국내 연구개발 동향

- 실제로 양자컴퓨팅의 비용효율성과 관련한 연구개발 내용은 거의 전무한 실정이다.

(2) 국외 연구개발 동향

- 2011년경부터 양자컴퓨팅의 계산성능과 성능검증에 대한 미국 정부 과제가 수행되었다. 이 과정에서는 주어진 알고리즘을 주어진 하드웨어에서 구동하는 경우 얼만큼의 시간과 크기, 그리고 정확도가 가능한지를 분석하는 프로그램이었다.
- 이러한 정부차원의 과제와는 별도로, 컴파일과정, 컴퓨터구조설계과정, 연산회로, 결함허용회로 최적화 등과 같은 부분적 구성요소들에서의 비용효율성 향상에 대한 연구개발은 현재 많은 기관/대학에서 지속적으로 진행되고 있다. 일례로 Microsoft의 경우에는 컴파일과정에서의 어셈블리 코드 크기의 최소화와 관련한 연구를 진행중에 있다.

2.2 양자컴퓨팅 플랫폼 구축 현황

- 전체 시스템 구조 요약

◇ 최외각 입출력

(1) 입력

- 평가대상 양자정보소자: 본 연구에서는 크게 3가지 양자정보소자를 대상으로 하고 있다. 반도체기반 양자점 큐빗과 초전도 큐빗기술, 그리고 광기반 광큐빗기술을 대상으로 하고 있다. 현재까지 적용 가능한 소자 기술은 양자점큐빗기술이다.
- 평가대상 양자알고리즘: 본 연구에서는 크게 3가지 양자알고리즘을 대상으로 하고 있다. 양자계산과학분야에서 사용되는 ground state estimation 알고리즘, 암호해독분야에서 사용되는 shor 알고리즘, 그리고 SW검증분야에서 사용되는 quantum boolean formula 알고리즘이다. 이 3가지 알고리즘에 대한 양자프로그램은 확보되었다.
- 평가대상 오류보정코드: 본 연구에서는 크게 3가지 오류보정코드를 대상으로 하고 있다. block코드 분야에서는 steane코드와 bacon-shor코드를, topological코드 분야에서는 surface코드를 고려하고 있다. 현재까지 적용가능 코드는 steane코드이다.

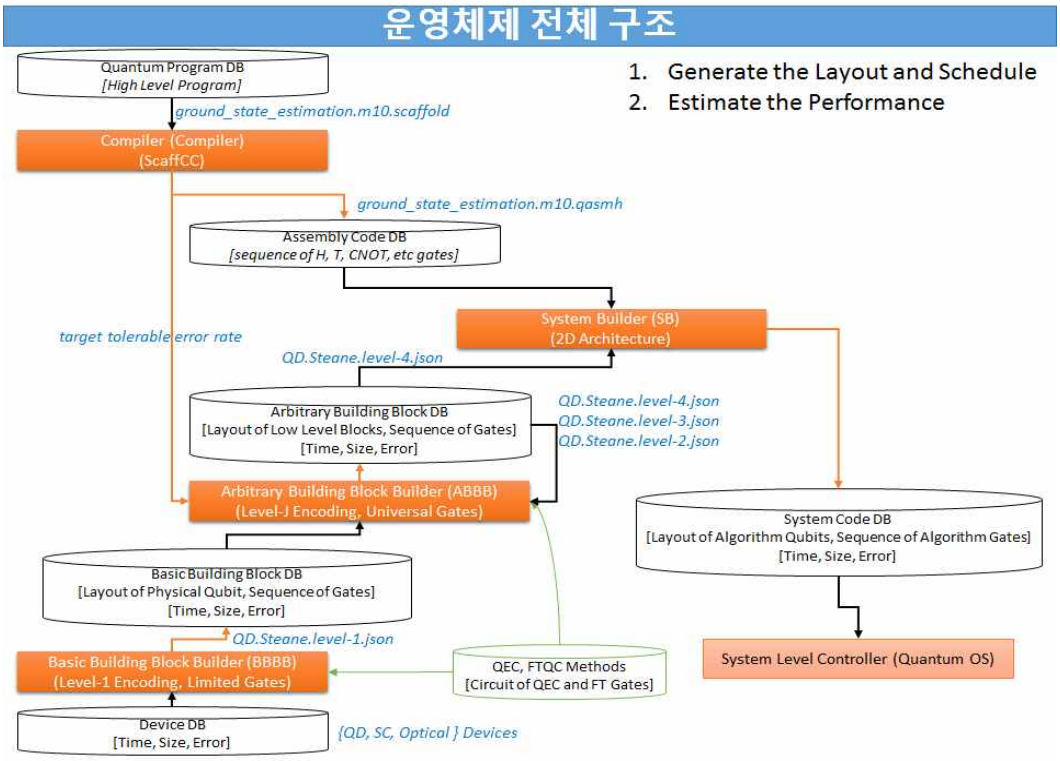
(2) 출력

- 1회 계산시간: 주어진 알고리즘, 주어진 소자기술, 주어진 오류보정 코드를 대상으로 하여 1회 계산하는데 걸리는 총 소요시간을 분석한다.
- 1회 연산정확도: 1회 연산과정을 거친후에 연산결과를 신뢰도를 평가한다.
- 평균 계산 시간: 1회 계산시간 * 1/1회 연산정확도를 계산하여, 평균적으로 소요되는 연산시간을 분석한다.
- 시스템의 크기: 설계된 양자컴퓨팅의 물리적 크기를 분석한다.

◇ 내부 동작 흐름

(1) 내부구조 흐름

- 아래 그림은 본 연구개발 과정에서 현재까지 구축한 양자컴퓨팅 플랫폼, 특히 운영체제 영역을 도식화한 것이다.



(2) 각 부분별 주요 입출력

주요 구성요소	입력	출력
컴파일러	❍ 양자컴퓨터 프로그램	❍ H, S, T, CNOT, X, Y, Z, Prep., Meas. 등으로 구성된 어셈브리코드 ❍ 계층적 코드 또는 순차적 코드 ❍ 어셈블리 코드수준에서의 큐빗기준 최대 허용 오류율
시스템 합성기	❍ 어셈블리코드 ❍ ABB블럭들	❍ ABB수준에서의 시스템코드
임의 빌딩블럭 합성기	❍ BBB블럭들 ❍ 양자오류보정코드 ❍ 어셈블리 코드수준에서의 큐빗기준 최대 허용 오류율	❍ ABB블럭들 ❍ ABB블럭이 수행되는 과정을 구현하는 빌딩블럭 수준 코드
기본 빌딩블럭 합성기	❍ 물리적 양자정보소자기술 ❍ 양자오류보정코드	❍ BBB블럭 ❍ BBB블럭이 수행되는 과정을 구현하는 물리적 수준 코드

(3) 각 부분별 주요 기능과 성능

주요 구성요소	기능/성능	결과값의 의미
컴파일러	❍ 주어진 양자프로그램을 결함허용적으로 구현가능한 게이트들의 조합으로 분해	❍ 양자어셈블리코드의 생성
	❍ 만들어진 어셈블리 코드를 기준으로 큐빗의 단위시간당(싸이클 타임) 최대 허용 오류율 분석	❍ 개별 큐빗을 기준으로 단위 시간당 허용하는 최대 오류율을 분석하여, 결함허용과정에서 적용되어야 할 오류보정, 결함허용의 강도를 결정함
	❍ critical path 분석 기능	❍ 어셈블리코드에서 최악의 경로에 대한 정보. 이를 통해서 이상적 상황에서 양자컴퓨터의 최고성능에 대한 예측 가능
	❍ 개별 모듈별 호출 횟수	❍ 개별 모듈에 대한 통계치를 분석하도록 하여, 컴퓨터구조 등의 합성과정에서 최적화 기본 자료로 활용
	❍ QX Simulator	❍ 양자어셈블리코드에 대한 검증과정에서 활용 ❍ 이를 통해서 컴파일 엔진이 생성하는 어셈블리 코드의 유효성을 검증함

주요 구성요소	기능/성능	결과값의 의미
시스템 합성기	❍ 계층적 배치 기능	❍ 모듈별 배치 기능을 지원하여, 어셈블리코드를 구성하는 모든 모듈을 시스템 수준에서 배치하는 기능
	❍ 계층적 분석 기능	❍ 각 모듈별 동작시간, 연산결과 정확도, 크기 등을 분석 ❍ 최종적으로는 주어진 어셈블리코드에 대해서 최종적으로 연산시간, 정확도, 크기등을 분석가능
	❍ 시스템 코드 생성 기능	❍ 양자컴퓨터를 구동하는 과정에서 구동의 대상이 되는 ABB블럭들의 배치와 시간적으로 수행해야 하는 조작열을 생성하는 기능

주요 구성요소	기능/성능	결과값의 의미
임의 빌딩블럭 합성기	❍ 기본 빌딩블럭 (Basic Building Block, BBB)의 배치 기능	❍ 주어진 오류보정회로, 결함허용 프로토콜에 맞추어서 BBB를 배치하는 기능
	❍ 성능 분석 기능	❍ 주어진 논리적 게이트를 수행하는 소요되는 절대시간, 정확도, 크기 등을 추출

주요 구성요소	기능/성능	결과값의 의미
기본 빌딩블럭 합성기	❍ 물리적 큐빗의 배치 기능	❍ 주어진 오류보정회로, 결함허용 프로토콜에 맞추어서 물리적 큐빗을 배치하는 기능
	❍ 성능 분석 기능	❍ 주어진 논리적 게이트를 수행하는데 소요되는 절대시간, 정확도, 크기 등을 추출

- 컴파일 부분

◇ 최외각 입출력

(1) 입력 : 양자 알고리즘 (Quantum Algorithms)

(2) 출력 : 양자 어셈블리 코드 (Quantum Assembly Codes) 및 논리적 빌딩블럭의 최대허용오류율 (Maximum Tolerable Error Rate)

- 현재 QFT, Ground state estimation, Shors, Boolean formula 등의 양자 알고리즘이 Scaffold 언어로 작성되어 있으며, 이러한 양자 알고리즘들이 컴파일 부분의 입력값이 된다. 예를 들면, ground_state_estimation.m10.scaffold (GSE 알고리즘), shors.n512.scaffold (쇼어의 알고리즘), boolean_formula.x2y2.scaffold (부울형식 알고리즘)등의 양자 알고리즘이 있다. 컴파일 부분의 출력인 양자 어셈블리 코드는 계층적 구조인 QASMH와 평면적 구조인 QASMF의 두 가지 형태로 산출된다. 논리적 빌딩블럭의 최대허용오류율은 큐빗 수(K)와 작동에 필요한 스텝수를 나타내는 Critical path(Q)에 의한 KQ 모델에 의하여 계산되어진다. 이 모델에서 허용하는 오류율은 조금 더 엄밀한 계산을 통하여 약간 더 향상될 수 있다.

◇ 내부 동작 흐름

- 컴파일 부분의 동작은 양자 알고리즘을 작성한 A.scaffold 파일을 입력으로 받아서, -h와 -f의 옵션을 통하여 A.qasmh나 A.qasmf 파일을 출력으로 얻게 된다. 또한, 논리적 빌딩빌럭에서 요구되는 최대 허용 오류율 값을 직접 작성하여 만든 -kq나 -im와 같은 옵션의 사용으로 얻을 수 있다 (그림 8. 참고).

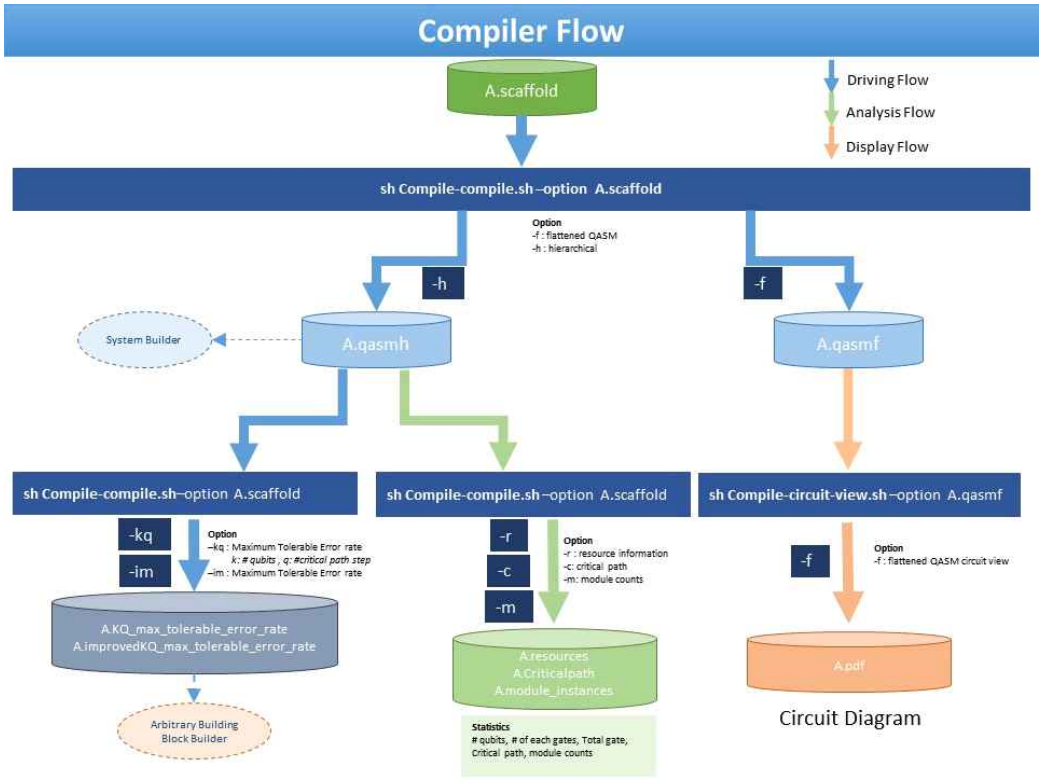


그림 8. 컴파일 부분의 동작 흐름도

- 컴파일 부분에서 입력된 A.scaffold로부터 컴파일 부분의 출력인 A.qasmh와 A.qasmf를 얻기까지에는 여러 단계를 거치며, 각 단계마다 중간에 생성되는 값들은 .ll 파일의 형태로 저장된다. 현재 컴파일 단계에서는 $(FILE)12.ll 파일까지의 중간 단계를 거쳐서, $(FILE).resources,$(FILE).qasmh, $(FILE).qasmf 등이 생성된다(그림 9. 참고).

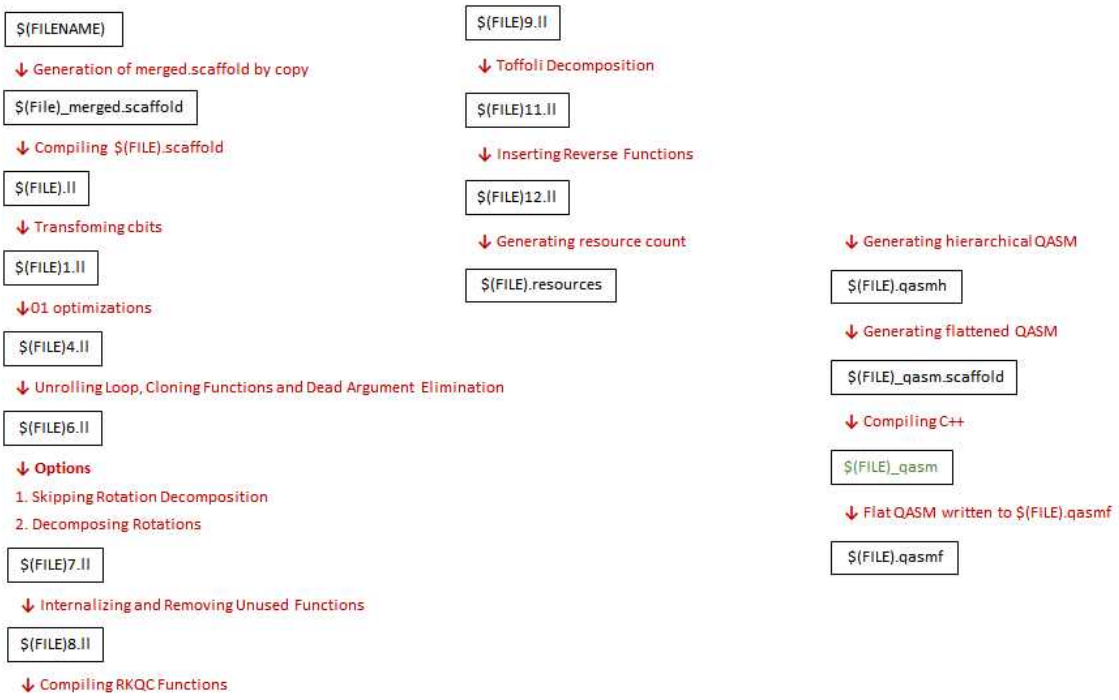


그림 9. 컴파일 부분의 내부 동작 흐름도

◇ 기능

(1) 동작 기능

- 고급언어로 작성된 양자 알고리즘을 양자 기계어인 QASM으로 변환
- 논리적 빌딩블럭에서 요구되는 최대허용오류율을 산출
- 양자 알고리즘을 구현하는 데 필요한 큐빗 수와 기본 게이트 수 등의 자원 산출
- 양자 기계어의 계층적 구조(QASMH) 산출 및 평면적 구조(QASMF) 산출
- 양자 기계어의 평면적 구조에 대한 부분적인 시각화(Circuit Diagram)

◇ 성능

- 세 가지의 주요 활용 알고리즘에 대한 컴파일이 가능하다. 구체적으로 살펴보면, ground_state_estimation.m10.scaffold (GSE 알고리즘), shors.n512.scaffold (쇼어의 알고리즘), boolean_formula.x2y2.scaffold (부울 형식 알고리즘)등의 알고리즘들에 대한 컴파일이 가능하다. 이중 가장 중요하게 보이는 주요 활용 알고리즘인 GSE 알고리즘에 대하여 아래와 같은 성능을 보인다.

1) M=100, b=3에 대한 GSE 알고리즘의 QASMH 파일 생성 가능
2) M=40, b=3에 대한 GSE 알고리즘의 QASMF 파일 생성 가능
3) M=40, b=3에 대한 GSE 알고리즘의 자원분석(Resource) 파일 생성 가능
4) M=40, b=3에 대한 GSE 알고리즘의 Critical path 파일 생성 가능
5) 83개의 단일 큐빗 게이트에 대한 시각화 가능

- GSE 알고리즘을 scaffCC 컴파일 프로그램으로 실행되도록 작성한 소스코드인 ground_state_estimation.m10.scaffold을 컴파일하는 실행화면은 그림 10.에 나타난다.

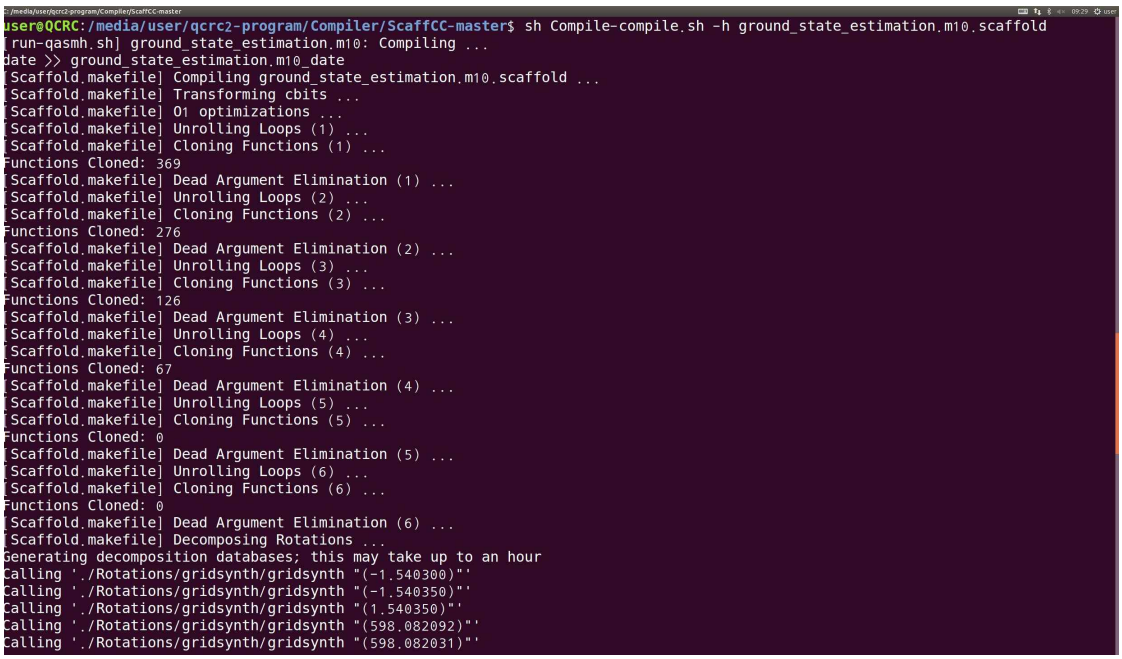


그림 10. GSE 알고리즘의 실행화면

- ground_state_estimation.m10.scaffold을 컴파일 한 실행결과 화면은 그림 11.에서 확인할 수 있다.

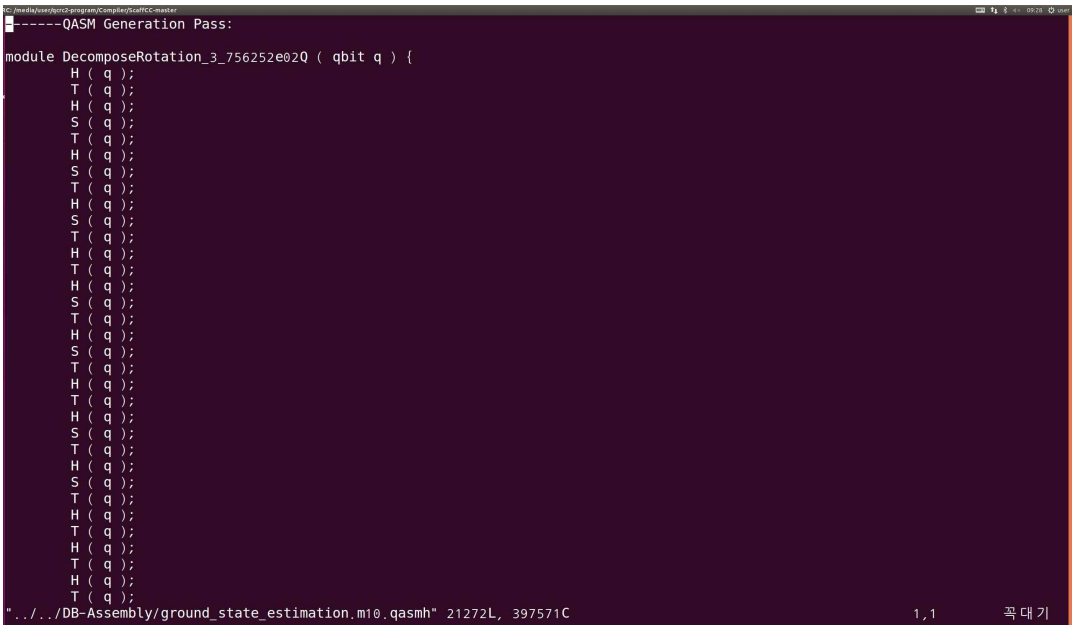


그림 11. GSE 알고리즘의 실행결과 화면

- 시스템 합성 부분

◇ 최외각 입출력

(1) 빌드업(Buildup)

1) 입력

- 컴파일 과정을 통해서 생성된 양자 어셈블리 코드
- 임의 빌딩 블록 파트에서 분석된 양자 게이트들의 실행 시간과 동작 오류율
- 양자 컴퓨팅 시스템의 구조(architecture)

2) 출력

- 양자 어셈블리 코드의 모듈별 시스템 코드
- 양자 어셈블리 코드의 모듈별 분석 내용
  a. 통계 자료(큐빗, 모듈, 양자 게이트)
  b. 양자 회로의 수행 정확성(fidelity)

(2) 분석(Analysis)

1) 입력

- 빌드업 과정을 통해서 생성된 시스템 코드
- 임의 빌딩 블록 파트에서 분석된 양자 게이트의 실행시간과 동작 오류율

2) 출력

- 통계 수치(큐빗 개수, 게이트별 호출 횟수, 모듈별 호출 횟수)
- 실행 관련 시간 정보(전체 게이트 사이클 수, 예상 양자 알고리즘 구동 시간)
- 양자 회로의 수행 정확성(fidelity)

(3) 구동(Execution)

1) 입력

- 빌드업 과정을 통해서 생성된 시스템 코드
- 임의 빌딩 블록 파트에서 분석된 양자 게이트들의 실행 시간과 동작 오류율
- 양자 컴퓨팅 시스템의 구조

2) 출력

- 양자 컴퓨터 하드웨어 제어 신호

◇ 내부 동작 흐름

- 시스템 합성은 시스템 코드 빌드업, 시스템 코드 분석 그리고 시스템 코드 구동 등 세 가지 기능을 수행한다.

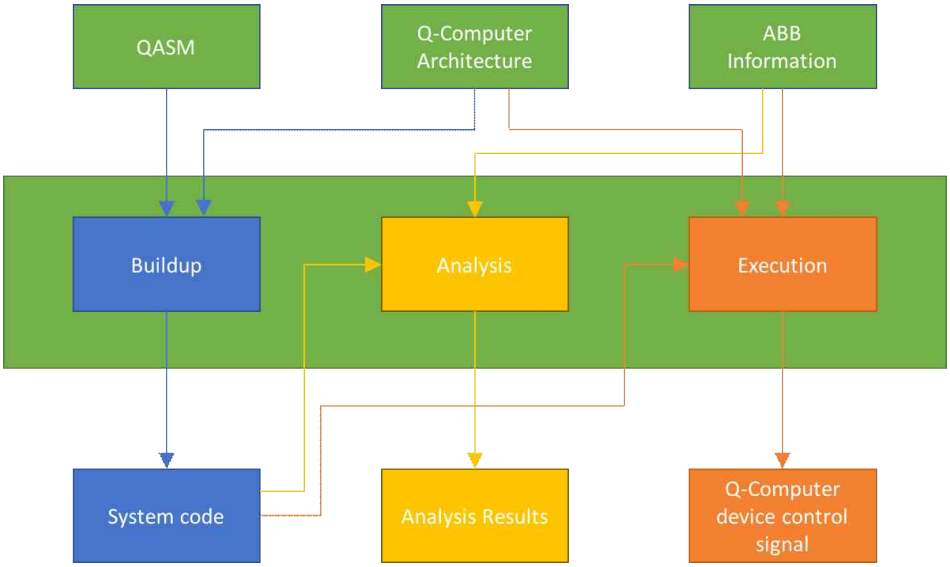


그림 12. 시스템 합성 툴(시스템 빌더)의 전체적인 동작 흐름

(1) 시스템 코드 빌드업

- 양자 알고리즘(양자 어셈블리 코드)을 특정한 양자 컴퓨터 시스템 하드웨어 상에 매핑하는 과정이다. 양자 어셈블리 코드, 양자 컴퓨터 시스템 구조(architecture)와 임의 빌딩 블록의 정보를 이용해서 선택한 양자 컴퓨터 시스템 상에서 알고리즘을 구동시키는 명령어 코드(시스템 코드)를 생성한다.

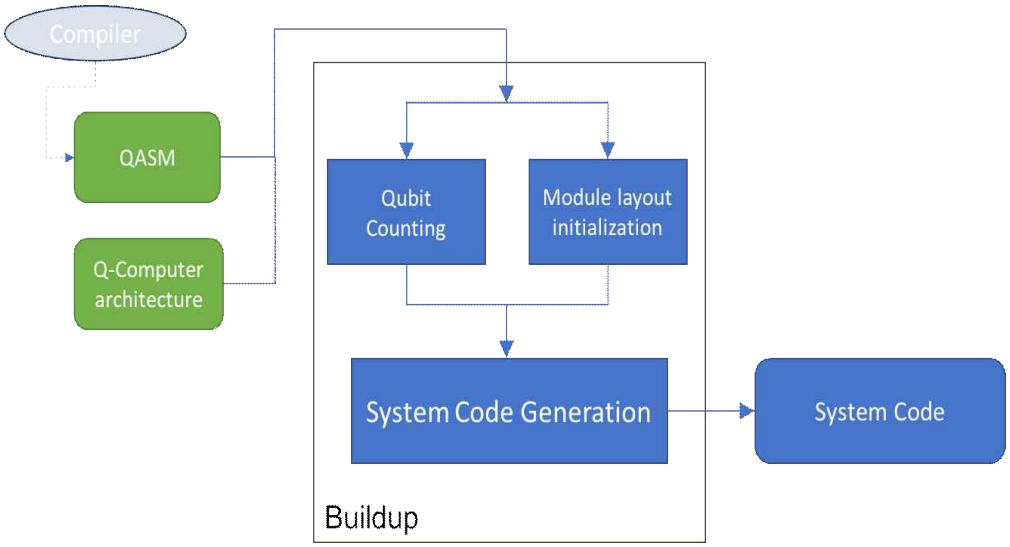


그림 13. 시스템 코드 빌드업 흐름도

(2) 시스템 코드 분석

- 빌드업 과정에서 생성된 시스템 코드를 정량적으로 분석하여 해당 양자 알고리즘을 해당 양자 컴퓨팅 시스템 상에서 구동하는데 필요한 자원량에 대한 통계 수치(큐빗

개수, 게이트별의 호출 횟수 등), 알고리즘 구동 관련 정보(전체 게이트 사이클 수, 예상 알고리즘 구동 시간)와 알고리즘 수행 정확성(fidelity)을 구하는 과정이다.

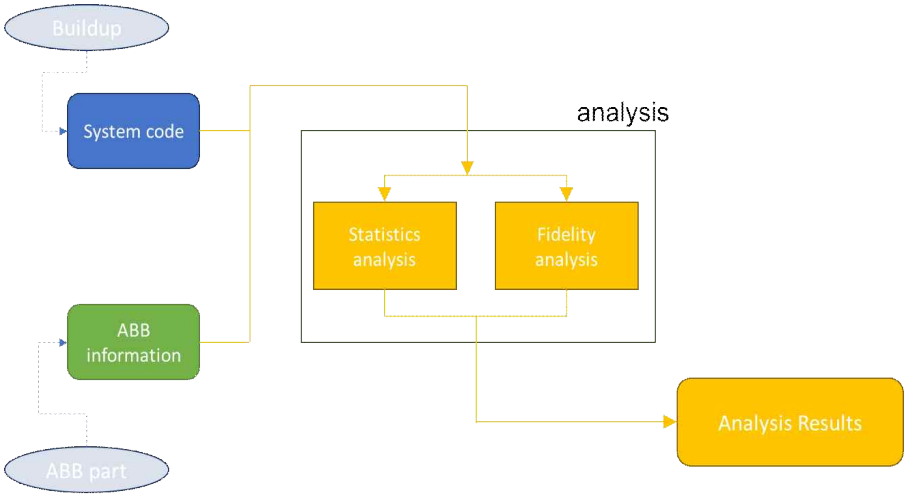


그림 14. 시스템 코드 분석 흐름도

(3) 시스템 코드 구동

- 양자 알고리즘을 해당 양자 컴퓨터 시스템 상에서 구동하기 위해서 시스템 빌드업 과정에서 생성된 시스템 코드의 명령어를 이용해서 양자 컴퓨터 시스템의 디바이스를 컨트롤 하는 부분이다. 양자 알고리즘을 구동하는 것은 해당 알고리즘(컴퓨터 프로그램)의 main 모듈을 실행시키는 것에 해당하므로 main 모듈의 시스템 코드를 읽어서 개별 명령어들을 처리하도록 한다. 해당 명령어가 양자 게이트인 경우에는 이에 해당하는 디바이스 제어 신호를 발생시키고, 만약 다른 서브 모듈인 경우 해당 모듈의 시스템 코드 파일을 열어서 해당 시스템 코드의 명령어들을 처리하도록 한다. 서브 모듈의 구동이 종료되면 이전 모듈의 시스템 코드로 돌아가 남아있는 명령어들을 처리한다.

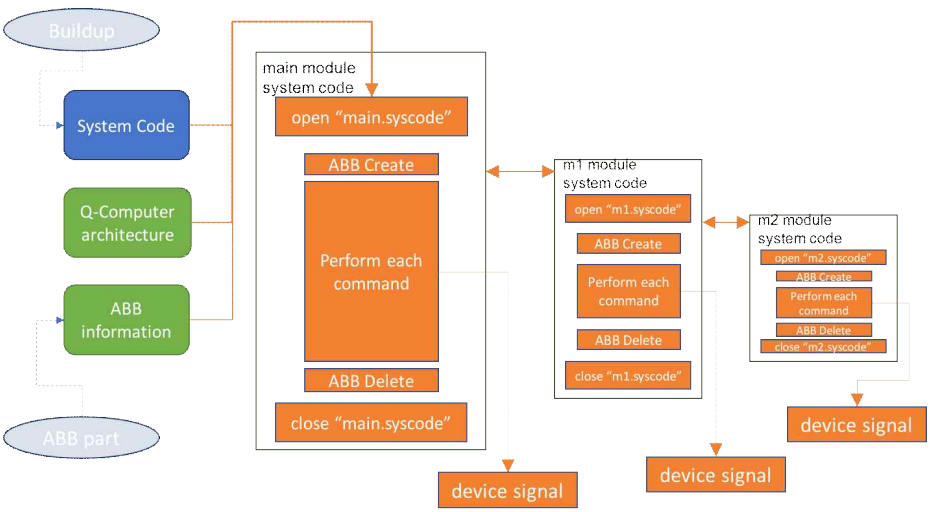


그림 15. 시스템 코드 구동 흐름도

◇ 기능

(1) 양자 시스템 코드 빌드업(시스템 코드 생성)

- 입력으로 선택된 양자 컴퓨터 시스템의 구조(양자 컴퓨터 회로) 상에 양자 알고리즘(양자 어셈블리 코드)을 매핑하여 해당 양자 컴퓨터 시스템의 구조에 종속적인 양자 어셈블리 코드(시스템 코드라 호칭)를 생성한다. 시스템 코드 빌드업 과정은 다음의 절차로 진행된다.

1) 큐빗 개수 확인:

- 양자 어셈블리 코드는 양자 컴퓨터 프로그래머가 작성한 프로그램 코드를 컴파일한 결과물로 프로그램에서 사용된 큐빗의 개수를 어셈블리 코드로부터 한눈에 확인하기 어렵다. 모듈 내 지역변수 큐빗의 경우 해당 큐빗의 선언부에서 그 개수를 명확히 확인할 수 있지만, 매개변수 큐빗의 경우에는 해당 모듈을 호출한 모듈의 분석을 통해서 확인할 수 있다. 본 시스템 합성 구현에서는 2 round scan 방식을 통해서 해당 알고리즘에 사용된 매개변수, 지역변수 큐빗의 개수를 확인한다. 참고로 이 단계에서 말하는 큐빗의 개수는 양자 시스템 코드 분석단계에서 말하는 큐빗의 통계 수치와는 다르다.
- 첫 번째 라운드에서는 각 모듈의 선언부를 확인하여 매개변수 큐빗들의 구조를 파악한다. 몇 개의 큐빗 배열들이 매개변수로 전달되었는지 확인한다. 이 과정은 양자 어셈블리 코드의 첫 번째 모듈부터 마지막 모듈까지 순차적으로 진행한다(Forward scan). 두 번째 라운드에서는 매개변수로 전달하는 각 큐빗 배열의 사이즈를 확인한다. 즉, 첫 번째 라운드에서는 매개변수를 전달받는 모듈입장에서 큐빗들의 구조를 파악했다면, 두 번째 라운드에서는 매개변수를 전달하는 모듈입장에서 큐빗 행렬의 구체적 사이즈를 확인한다. 이 과정은 마지막 모듈(main 모듈)부터 가장 처음 모듈의 순서로 진행된다(Backward scan).

2) 모듈 배치 초기화:

- 양자 컴퓨터 시스템의 구조에 의해서 양자 컴퓨터 회로의 구조(본 구현에서는 2D)가 선택되면, 앞선 단계에서 확인한 모듈별 큐빗 개수에 맞도록 양자 컴퓨터 회로를 초기화 한다. 특히, 큐빗의 배치가 중요한데, 현재의 시스템 합성 구현에서는 FCFS 방식을 채택하였다. 즉 양자 어셈블리 코드에 등장하는 순서대로 해당 큐빗을 배치한다.

top [0]	top [1]	top [2]	bottom [0]	bottom [1]	bottom [2]	bottom [3]
bottom [4]	bottom [5]	bottom [6]	bottom [7]	bottom [8]	bottom [9]	bottom [10]
bottom [11]	bottom [12]	bottom [13]	bottom [14]	bottom [15]	bottom [16]	bottom [17]
bottom [18]	bottom [19]	bottom [20]	bottom [21]	bottom [22]	bottom [23]	bottom [24]
bottom [25]	bottom [26]	bottom [27]	bottom [28]	bottom [29]	bottom [30]	bottom [31]
bottom [32]	bottom [33]	bottom [34]	bottom [35]	bottom [36]	bottom [37]	bottom [38]
bottom [39]	dummy	dummy	dummy	dummy	dummy	dummy

그림 16. 모듈 레이아웃 예제(GSE 알고리즘 입력 M=40 경우의 main 모듈, 43개의 지역변수 큐빗이 이용됨.)

3) 양자 어셈블리 코드 명령어 별 시스템 코드 명령어 생성:

- 모듈별로 큐빗들의 배치가 종료되면, 모듈별로 양자 어셈블리 코드 명령어들을 읽고 해당 명령어에 맞게 시스템 코드를 생성한다. 시스템 코드를 생성할 때, 필요에 따라 적절한 전처리/후처리 과정이 필요하다.
- one-qubit gate(PrepZ, MeasZ, Pauli, H, T, S 등) 경우에는 대상 큐빗에 해당 게이트를 인가하는 내용의 시스템 코드를 생성하면 된다. 반면, two-qubit gate(CNOT)의 경우엔 전처리 과정이 필요한데, 사용하는 CNOT gate 가 인접한 두 큐빗을 대상으로 하는 local gate 이기 때문이다. 따라서 만약 CNOT gate의 대상이 되는 큐빗들이 인접해 있지 않을 경우 큐빗의 이동을 통해서 대상 큐빗들이 인접하여 위치하도록 해주어야 한다. 큐빗 이동은 SWAP gate를 인가하여 수행할 수 있다. 즉, CNOT gate를 수행하기 위해서 적절한 SWAP gate 들의 수행이 사전에 수행되어야 한다. 그리고 해당 SWAP gate 의 수행도 시스템 코드 상에 반영되어야 한다.
- 모듈 호출의 경우엔 양자 게이트의 경우보다 복잡하다. 양자 어셈블리 코드에서 모든 처리과정은 모듈 단위로 이루어지고, 메모리상에 큐빗들도 모듈 단위로 배치된다. 즉, 현재 시스템 빌드업 중인 모듈과 해당 모듈에서 호출하는 모듈은 메모리상에서 임의의 위치에 배치될 수 있다. 따라서 모듈 호출의 경우 먼저 현재 모듈에서 큐빗 일부를 피호출 모듈(called module) 내 매개변수 큐빗의 영역으로 매개변수 전달(parameter passing)을 수행해야 한다(forward parameter passing). 그리고 전달된 큐빗 상에서 해당 모듈의 명령어들을 수행한다. 해당 모듈의 명령어 수행이 종료되면 앞서 매개변수로 전달되었던 큐빗들을 기존의 모듈로 다시 전달해주어야 한다(backward parameter passing). 즉, 모듈 호출의 경우 전처리 과정으로 forward parameter passing 을 수행하고, 후처리로 backward parameter passing 을 수행해주어야 한다.

(2) 양자 시스템 코드 분석:

- 양자 시스템 코드의 분석과정은 양자 어셈블리 코드 내에 모듈이 나타나는(기록되어있는) 순서대로 진행한다. 이는 디지털 컴퓨터 프로그래밍의 경우처럼 현재 시점에서 호출하고자 하는 모듈 또는 함수는 사전에 이미 선언되고 정의되어 있어야하기 때문이다.

1) 통계 수치 분석:

- 해당 양자 알고리즘 수행을 위해서 필요한 자원(큐빗, 양자 게이트 등)을 정량적으로 분석한다. 먼저 어셈블리 코드에서 가장 첫 번째 나타나는 모듈의 통계 분석(큐빗, 게이트 개수)을 수행하고, 이후에 두 번째 모듈의 통계 분석을 시작한다. 만약 두 번째 모듈이 첫 번째 모듈을 호출하는 부분을 포함하고 있으면 첫 번째 모듈의 분석 내용이 두 번째 모듈 분석에 포함되어야 한다. 즉, *i* 번째 모듈의 통계 분석과정에서 1 ~ *i*-1 번째 모듈의 통계 분석 결과가 반영될 수 있다. 이런 일련의 귀납적 분석을 진행하여 마지막 main 모듈의 분석에는 양자 어셈블리 코드 내 다른 모든 모듈들의 통계 분석 결과가 반영되게 된다.

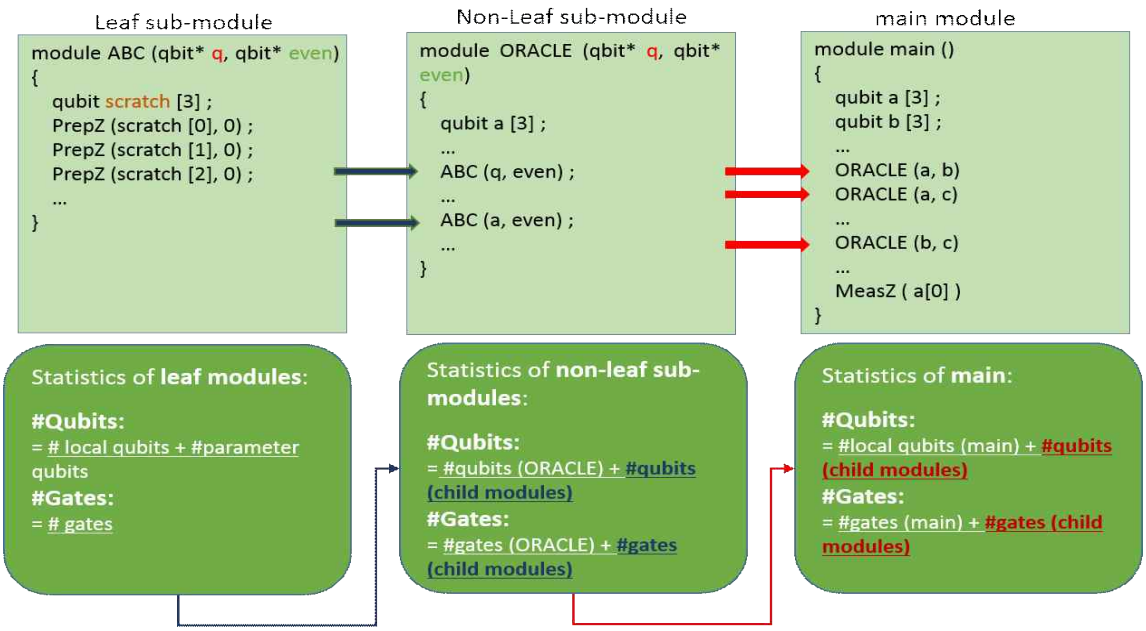


그림 17. 귀납적으로 수행하는 양자 시스템 코드의 정량 분석

2) 양자 알고리즘 수행 정확성(fidelity) 분석:

- noisy 양자 컴퓨팅 디바이스에 대한 정보를 바탕으로 해당 양자 알고리즘의 수행 정확성(fidelity) 분석할 수 있다. 수행 정확성 계산을 위한 그림 18.의 원리들을 적용한다. 통계 수치 분석과 마찬가지로 수행 정확성 분석도 양자 어셈블리 코드에 등장하는 모듈들의 순서대로 귀납적으로 진행한다.

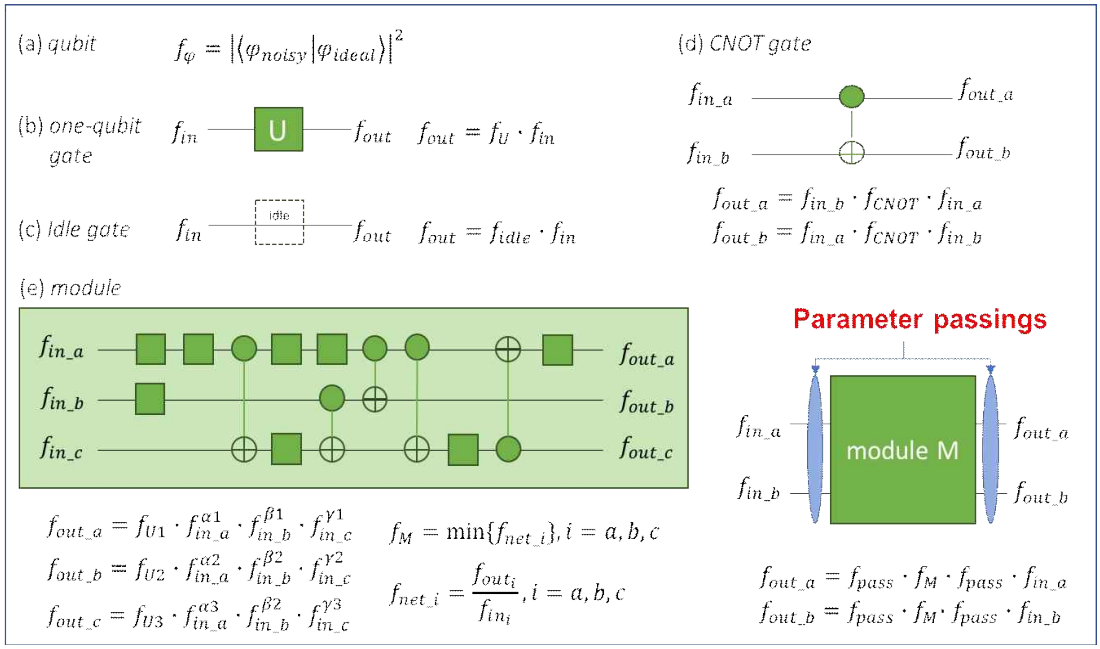


그림 18. 양자 알고리즘 수행 정확성(Fidelity) 산출 원리

(3) 양자 알고리즘 구동

- 생성된 양자 시스템 코드의 구동을 제어하는 컨트롤러 역할. 해당 시스템 코드의 명령어 분석을 통해서 양자 컴퓨팅 디바이스에 명령을 전달하는 역할을 수행한다. 디지털 컴퓨터 프로그램의 경우처럼 양자 어셈블리 코드의 경우도 main 모듈의 시작부터 마지막까지 명령어를 처리하는 것으로 양자 알고리즘을 구동할 수 있다. 시스템 빌드업 과정을 통해서 생성된 시스템 코드는 모듈별로 개별 파일에 존재하므로, 양자 알고리즘의 구동은 main 모듈의 시스템 코드 파일을 읽고 명령어 처리하고 다른 모듈을 호출하는 부분에서는 해당 모듈의 시스템 코드 파일을 읽고 명령어를 처리하도록 한다. 이 과정은 재귀적으로 구성되어 있다.
- 현재까지의 시스템 합성 구현에서 양자 알고리즘의 구동은 미완성 단계로 구동을 위한 더욱 자세한 기능들은 향후 구현할 예정이다.

◇ 성능

(1) 빌드업

- 시스템 합성 툴은 빌드업 가능한 양자 알고리즘에 제한을 두지 않는다. 다만, 시스템 합성 툴이 구동되는 디지털 컴퓨터의 시스템 사양(특히, 메모리 크기)에 따라서 빌드업 가능한 양자 알고리즘의 입력 크기는 결정될 수 있다. 양자 어셈블리 코드 파일은 매우 큰 사이즈를 가지는데, 빌드업 과정에서 해당 양자 어셈블리 코드 내용이 메모리상에 적재되어야 하므로 디지털 컴퓨터에 장착된 메모리 크기에 따라서 빌드업 과정이 원활하지 않거나 때로는 불가능 할 수 있다. 참고로 현재까지 연

구실에서 컴파일 가능한 입력 크기의 양자 알고리즘의 경우는 해당 문제가 발생하지는 않는다.

- 시스템 합성 툴의 빌드업 실행시간의 경우 양자 어셈블리 코드의 모듈의 개수에 큰 영향을 받는 것으로 파악되고 있다. 양자 시스템 코드 빌드업 과정과 분석 과정 등이 모듈단위로 이루어지기 때문이다. 따라서 모듈의 개수가 증가할수록 툴의 구동시간이 증가하게 되는데, 현재 연구실의 주요 목적 알고리즘인 GSE 의 경우 입력 크기(M)에 따라서 모듈의 개수가 다항적으로 증가하기 때문에 입력 크기가 증가함에 따라서 빌드업 실행시간이 크게 증가한다. 아래 그림 8 에서는 GSE 알고리즘의 입력 크기에 따른 모듈 수 증가를 보여주고, 그림 9 에서는 GSE 알고리즘의 빌드업 시간 증가를 보여주고 있다.

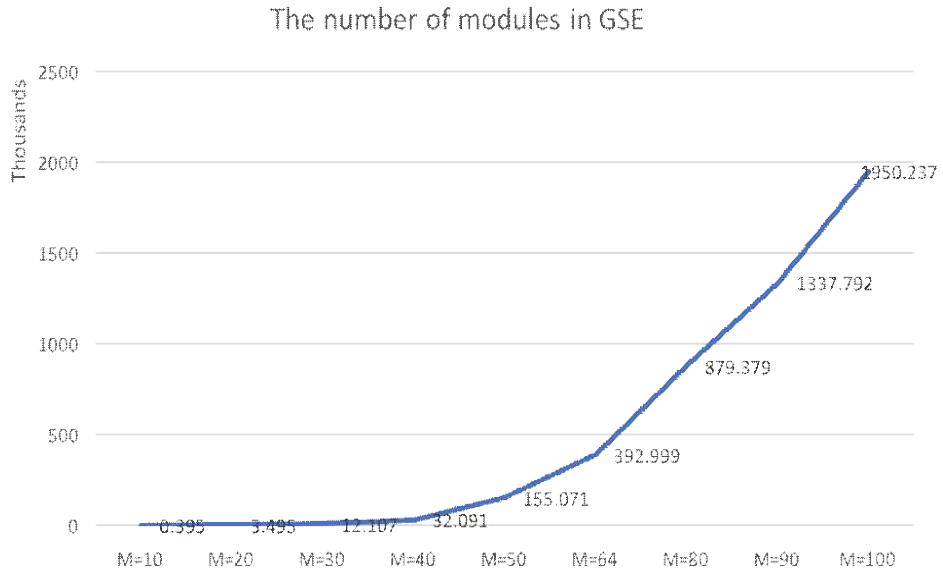


그림 19. GSE 알고리즘의 입력 크기(M=10~100)에 따른 모듈의 개수

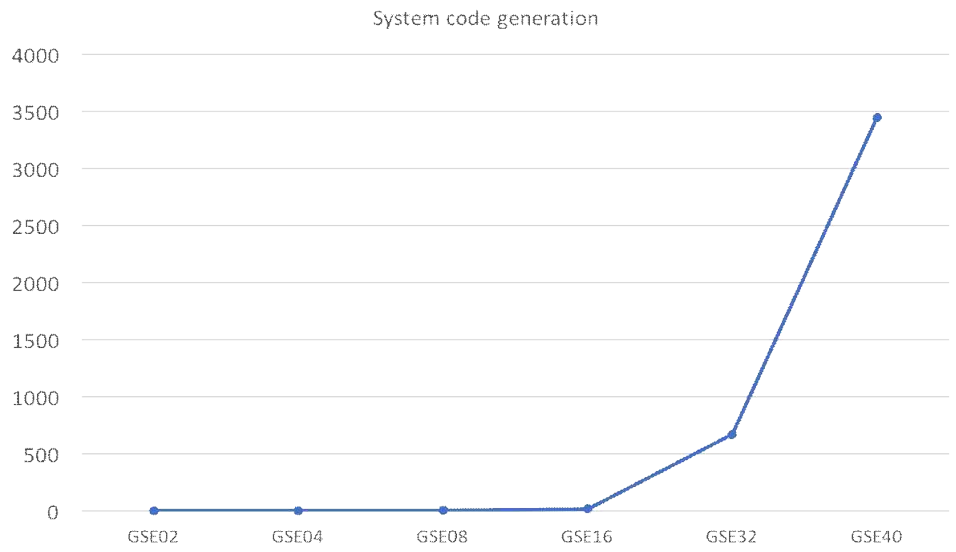


그림 20. GSE 시스템 코드 빌드업에 소요된 시간(3.5GHz CPU, 64G RAM), 단위 초.

(2) 시스템 코드 분석

- 임의 입력의 양자 알고리즘의 시스템 코드 분석이 가능하다. 다만 분석하고자 하는 대상에 따라서 효율성은 다르다. 시스템 코드의 통계 수치 산출하는 기능은 대부분의 연산이 덧셈으로 구성되어 있어서 빠르게 해당 기능을 수행할 수 있다. 반면, 양자 알고리즘 수행 정확성(fidelity)을 산출하는 기능은 배정도 정밀도를 가지는 실수들의 곱셈으로 이루어져 있고 또한 시스템 코드의 모든 명령어를 대상으로 하기 때문에 상대적으로 높은 연산 복잡도를 가진다. 또한 수행 정확성을 산출하는 기능은 symbolic computing을 지원하여 양자 알고리즘의 수행 정확성을 실제 수치 값이 아닌 큐빗과 게이트의 오류율의 함수형태로 표현할 수 있다. 하지만 해당 기능은 수행 정확성을 수치로 산출하는 것보다 더욱 높은 연산 복잡도를 가진다.

(3) 시스템 코드 구동

- 시스템 코드의 구동 기능은 아직 완전하게 구현되지 않아서 그 성능을 언급하기엔 이른 단계이다. 다만, 구동의 부분은 모든 모듈의 시스템 코드 파일을 읽고 처리하는 부분이 재귀적으로 이루어지는데, 따라서 많은 시스템 코드 파일들(그림 19. 참조)을 읽는 부분이 시스템 코드 구동 성능의 효율성을 떨어뜨리는 것으로 파악된다.

- 임의빌딩블럭/기본빌딩블럭 부분

◇ 최외각 입출력

(1) 입력

- 컴파일 부분 : 최대 허용 오류율(GSE 알고리즘)
- 기본빌딩블럭 부분 : Quantum dot 소자 기술을 바탕으로 한 기본빌딩블럭의 각 기능, 게이트 별 오류율 및 동작 시간

(2) 출력

- 임의빌딩블럭의 각 기능, 게이트 별 오류율 및 동작 시간

◇ 내부 동작 흐름

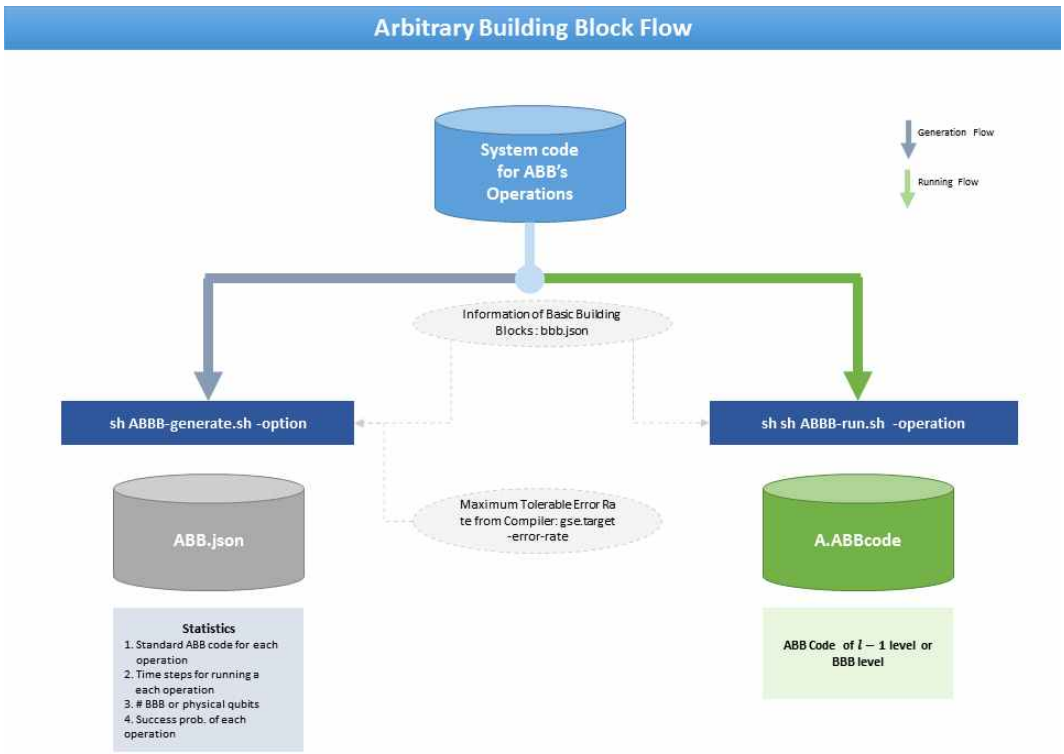


그림 21. 임의빌딩블럭 내부 동작 흐름도

- 임의빌딩블럭의 레벨을 1부터 구축하여 각 기능별 오류율을 컴파일 부분에서 요구하는 최대 허용 오류율과 비교한다. 가장 성능이 좋지 않은 기능의 성능이 최대 허용 오류율보다 좋을 때까지 임의빌딩블럭의 레벨을 상향시키며 임의빌딩블럭의 정보를 구축한다.

◇ 기능

(1) 동작 기능

1) 1개 소자기술 성능 평가 가능 : Quantum dot
2) 1개 양자오류정정부호 성능 평가 가능 : Steane code
3) 컴파일 부분에서 계산한 최대허용오류율에 따른 임의 레벨의 임의빌딩블럭 데이터 평가 가능
4) 비용 분석 및 구동을 위한 임의빌딩블럭의 각종 내부동작기능 구현 완료
5) 임의빌딩블럭 데이터 구축을 위한 스크립트 작성 완료
   - ./ABBB-generate.sh -D:QD -QEC:Steane -Terr:MaxTolerableErrorRate

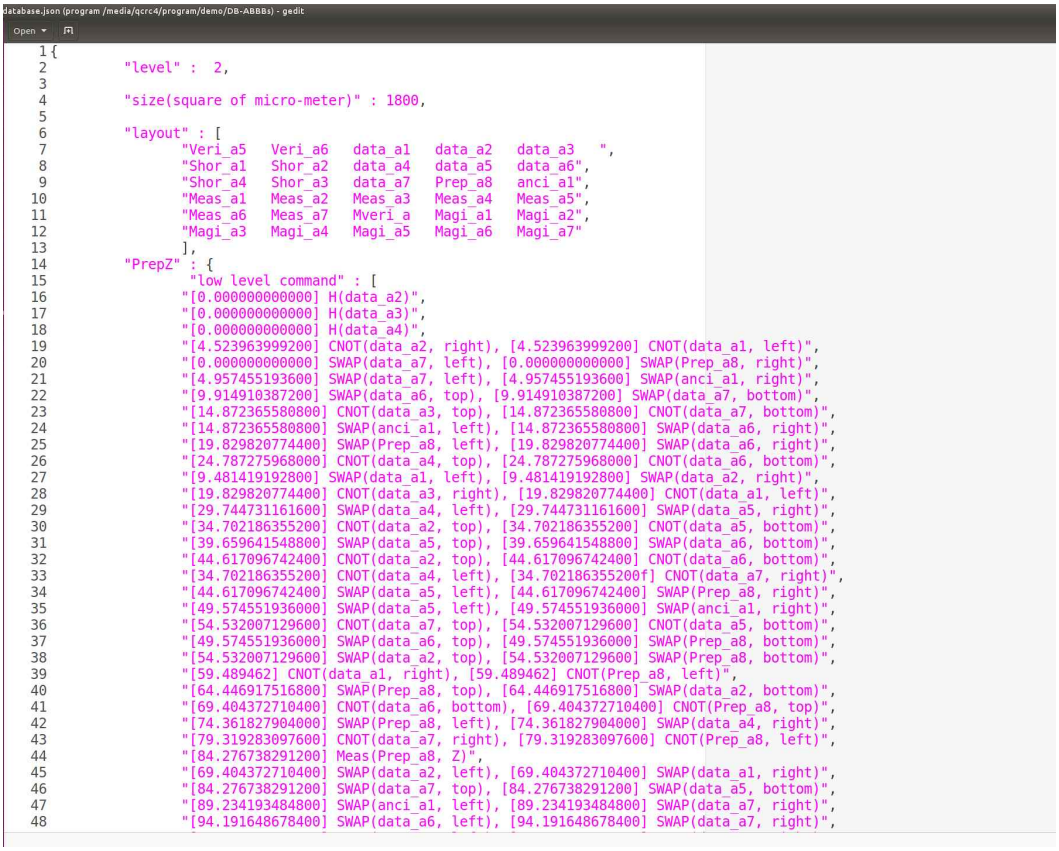
```
qcrc4: /media/qcrc4/program/demo/program
qcrc4@qcrc4:/media/qcrc4/program/demo/program$ ./ABBB-generate.sh -D:QD -QEC:Steane -Terr:ground_state_estimation.m40.improvedKQ_max_tolerable_error_rate
```

그림 22. 임의빌딩블럭 데이터 구축을 위한 스크립트

- 해당 스크립트를 실행 시킬 경우 최대허용오류율에 따라 임의빌딩블럭의 레벨을 증가 시키면서 각 레벨의 임의빌딩블럭 데이터를 구축한다.

```
qcrc4: /media/qcrc4/program/demo/program
[166229.083785456431] SWAP(data_a2, left), [166229.083785456431] SWAP(Veri_a6, right)
[166229.083785456431] SWAP(data_a2,Veri_a6)
[167250.653099194024] SWAP(data_a3, left), [167250.653099194024] SWAP(Veri_a6, right)
[167250.653099194024] SWAP(data_a3,Veri_a6)
[174541.474953619589] CNOT(Shor_a2, left), [174541.474953619589] CNOT(Veri_a6, right)
[174541.474953619589] CNOT(Shor_a2,Veri_a6,left,right)
[169293.791726669209] SWAP(data_a7, left), [169293.791726669209] SWAP(Shor_a4, right)
[169293.791726669209] SWAP(data_a7,Shor_a4)
[172498.336326144403] SWAP(Veri_a5, left), [172498.336326144403] SWAP(Shor_a4, right)
[172498.336326144403] SWAP(Veri_a5,Shor_a4)
[174541.474953619589] CNOT(Shor_a3, top), [174541.474953619589] CNOT(Shor_a4, bottom)
[174541.474953619589] CNOT(Shor_a3,Shor_a4,top,bottom)
[175563.044267357182] CNOT(Shor_a1, bottom), [175563.044267357182] CNOT(Veri_a6, top)
[175563.044267357182] CNOT(Shor_a1,Veri_a6,bottom,top)
[175563.044267357182] CNOT(Shor_a4, left), [175563.044267357182] CNOT(Veri_a5, right)
[175563.044267357182] CNOT(Shor_a4,Veri_a5,left,right)
[176584.613581094774] Meas(Veri_a5, Z)
[176584.613581094774] Meas(Veri_a6, Z)
verification successful
[150535.769586994109] SWAP(data_a5, left), [150535.769586994109] SWAP(data_a4, right)
[150535.769586994109] SWAP(data_a5,data_a4)
[168272.222412931616] SWAP(data_a6, left), [168272.222412931616] SWAP(data_a4, right)
[168272.222412931616] SWAP(data_a6,data_a4)
[176584.613581094774] CNOT(Shor_a1, left), [176584.613581094774] CNOT(data_a4, right)
[176584.613581094774] CNOT(Shor_a1,data_a4,left,right)
[169293.791726669209] SWAP(data_a6, left), [169293.791726669209] SWAP(data_a5, right)
[169293.791726669209] SWAP(data_a6,data_a5)
[177606.182894832367] SWAP(data_a4, left), [177606.182894832367] SWAP(data_a5, right)
[177606.182894832367] SWAP(data_a4,data_a5)
[178627.752208569960] CNOT(Shor_a1, left), [178627.752208569960] CNOT(data_a5, right)
[178627.752208569960] CNOT(Shor_a1,data_a5,left,right)
[178627.752208569960] SWAP(data_a4, left), [178627.752208569960] SWAP(data_a6, right)
[178627.752208569960] SWAP(data_a4,data_a6)
[179649.321522307553] SWAP(data_a5, left), [179649.321522307553] SWAP(data_a6, right)
[179649.321522307553] SWAP(data_a5,data_a6)
[180670.890836045146] CNOT(Shor_a1, left), [180670.890836045146] CNOT(data_a6, right)
[180670.890836045146] CNOT(Shor_a1,data_a6,left,right)
[177836.120728968381] SWAP(Veri_a5, left), [177836.120728968381] SWAP(data_a7, right)
[177836.120728968381] SWAP(Veri_a5,data_a7)
[181692.460149782739] CNOT(Shor_a1, top), [181692.460149782739] CNOT(data_a7, bottom)
[181692.460149782739] CNOT(Shor_a1,data_a7,top,bottom)
[182714.029463520332] Meas(Shor_a1, X)
[175563.044267357182] Meas(Shor_a2, X)
[175563.044267357182] Meas(Shor_a3, X)
[176584.613581094774] Meas(Shor_a4, X)
Correction operator is IIIIIII
Do nothing
[181692.460149782739] SWAP(data_a6, top), [181692.460149782739] SWAP(Veri_a5, bottom)
[181692.460149782739] SWAP(data_a6,Veri_a5)
[182714.029463520332] SWAP(data_a3, top), [182714.029463520332] SWAP(Veri_a5, bottom)
```

그림 23. 임의빌딩블럭 데이터 구축 실행 화면




```
{
        "level" :  2,

        "size(square of micro-meter)" : 1800,

        "layout" : [
                "Veri_a5   Veri_a6   data_a1   data_a2   data_a3   ",
                "Shor_a1   Shor_a2   data_a4   data_a5   data_a6",
                "Shor_a4   Shor_a3   data_a7   Prep_a8   anci_a1",
                "Meas_a1   Meas_a2   Meas_a3   Meas_a4   Meas_a5",
                "Meas_a6   Meas_a7   Mveri_a   Magi_a1   Magi_a2",
                "Magi_a3   Magi_a4   Magi_a5   Magi_a6   Magi_a7"
                ],
        "PrepZ" : {
                 "low level command" : [
                "[0.000000000000] H(data_a2)",
                "[0.000000000000] H(data_a3)",
                "[0.000000000000] H(data_a4)",
                "[4.523963999200] CNOT(data_a2, right), [4.523963999200] CNOT(data_a1, left)",
                "[0.000000000000] SWAP(data_a7, left), [0.000000000000] SWAP(Prep_a8, right)",
                "[4.957455193600] SWAP(data_a7, left), [4.957455193600] SWAP(anci_a1, right)",
                "[9.914910387200] SWAP(data_a6, top), [9.914910387200] SWAP(data_a7, bottom)",
                "[14.872365580800] CNOT(data_a3, top), [14.872365580800] CNOT(data_a7, bottom)",
                "[14.872365580800] SWAP(anci_a1, left), [14.872365580800] SWAP(data_a6, right)",
                "[19.829820774400] SWAP(Prep_a8, left), [19.829820774400] SWAP(data_a6, right)",
                "[24.787275968000] CNOT(data_a4, top), [24.787275968000] CNOT(data_a6, bottom)",
                "[9.481419192800] SWAP(data_a1, left), [9.481419192800] SWAP(data_a2, right)",
                "[19.829820774400] CNOT(data_a3, right), [19.829820774400] CNOT(data_a1, left)",
                "[29.744731161600] SWAP(data_a4, left), [29.744731161600] SWAP(data_a5, right)",
                "[34.702186355200] CNOT(data_a2, top), [34.702186355200] CNOT(data_a5, bottom)",
                "[39.659641548800] SWAP(data_a5, top), [39.659641548800] SWAP(data_a6, bottom)",
                "[44.617096742400] CNOT(data_a2, top), [44.617096742400] CNOT(data_a6, bottom)",
                "[34.702186355200] CNOT(data_a4, left), [34.702186355200f] CNOT(data_a7, right)",
                "[44.617096742400] SWAP(data_a5, left), [44.617096742400] SWAP(Prep_a8, right)",
                "[49.574551936000] SWAP(data_a5, left), [49.574551936000] SWAP(anci_a1, right)",
                "[54.532007129600] CNOT(data_a7, top), [54.532007129600] CNOT(data_a5, bottom)",
                "[49.574551936000] SWAP(data_a6, top), [49.574551936000] SWAP(Prep_a8, bottom)",
                "[54.532007129600] SWAP(data_a2, top), [54.532007129600] SWAP(Prep_a8, bottom)",
                "[59.489462] CNOT(data_a1, right), [59.489462] CNOT(Prep_a8, left)",
                "[64.446917516800] SWAP(Prep_a8, top), [64.446917516800] SWAP(data_a2, bottom)",
                "[69.404372710400] CNOT(data_a6, bottom), [69.404372710400] CNOT(Prep_a8, top)",
                "[74.361827904000] SWAP(Prep_a8, left), [74.361827904000] SWAP(data_a4, right)",
                "[79.319283097600] CNOT(data_a7, right), [79.319283097600] CNOT(Prep_a8, left)",
                "[84.276738291200] Meas(Prep_a8, Z)",
                "[69.404372710400] SWAP(data_a2, left), [69.404372710400] SWAP(data_a1, right)",
                "[84.276738291200] SWAP(data_a7, top), [84.276738291200] SWAP(data_a5, bottom)",
                "[89.234193484800] SWAP(anci_a1, left), [89.234193484800] SWAP(data_a7, right)",
                "[94.191648678400] SWAP(data_a6, left), [94.191648678400] SWAP(data_a7, right)",
```

그림 24. 레벨 2 임의빌딩블럭 데이터 파일

- 각 레벨의 임의빌딩블럭의 데이터가 구축이 완료되면 ABB_level_database.json 파일에 임의빌딩블럭의 레벨, 크기, 배열, 각 기능별 명령어 및 수행 시간, 각 기능별 성능이 저장된다.

(2) 내부 동작 기능 상세

1) SWAP 게이트

- 명령어 : SWAP(ABB_name, tar1, tar2, opt)
- 옵션 : rl - SWAP 게이트에 사용되는 임의빌딩블럭의 원소가 서로 가로 선상에 존재
  tb - SWAP 게이트에 사용되는 임의빌딩블럭의 원소가 서로 세로 선상에 존재
- 임의빌딩블럭 원소 단위의 SWAP 게이트로 동작 시 SWAP(tar1, left), SWAP(tar2, right) 혹은 SWAP(tar1, top), SWAP(tar2, bottom)가 출력된다.

2) CNOT 게이트

- 명령어 : CNOT(ABB_name, control, target)
- 임의빌딩블럭의 원소 단위의 CNOT으로 control 원소와 target 원소가 멀리 떨어져 있을 경우 target 원소를 control 원소에 인접하게 SWAP하여 CNOT 게이트를

진행한다. 이에 해당하는 SWAP 게이트의 출력문이 출력되며 최종적으로 CNOT(control, target, position of control, position of target)가 출력된다.

3) Logical gate preparation

- 명령어 : LOpPre(ABB_name, opt)
- 옵션 : r - 해당 임의빌딩블럭이 임의빌딩블럭 간 연산에 있어서 오른쪽에 위치
  l - 해당 임의빌딩블럭이 임의빌딩블럭 간 연산에 있어서 왼쪽에 위치
  t - 해당 임의빌딩블럭이 임의빌딩블럭 간 연산에 있어서 위쪽에 위치
  b - 해당 임의빌딩블럭이 임의빌딩블럭 간 연산에 있어서 아래쪽에 위치
- 해당 기능은 임의빌딩블럭 간 연산을 위한 준비과정이다. 임의빌딩블럭 내 데이터 원소들을 옵션에 따라 임의빌딩블럭의 해당 모서리에 나열시킨다. 이때 사용되는 SWAP 게이트의 출력문이 출력된다.

4) Logical CNOT 게이트

- 명령어 : LCNOT(controlABB_name, targetABB_name, opt)
- 옵션 : rl - 논리적 CNOT 게이트 연산에 있어서 target 임의빌딩블럭이 오른쪽, control 임의빌딩블럭이 왼쪽에 존재
  lr - 논리적 CNOT 게이트 연산에 있어서 target 임의빌딩블럭이 왼쪽, control 임의빌딩블럭이 오른쪽에 존재
  tb - 논리적 CNOT 게이트 연산에 있어서 target 임의빌딩블럭이 위쪽, control 임의빌딩블럭이 아래쪽에 존재
  bt - 논리적 CNOT 게이트 연산에 있어서 target 임의빌딩블럭이 아래쪽, control 임의빌딩블럭이 위쪽에 존재
- 임의빌딩블럭 간 CNOT 게이트 연산이다. 해당 연산을 수행하기 전에 각 임의빌딩블럭에 있어서 LOpPre(*)가 자동적으로 선행된다. 각 임의빌딩블럭에서 서로 대응되는 데이터 원소 간 CNOT 게이트 연산이 이루어지며 데이터 원소의 수가 임의빌딩블럭의 배열의 가로, 세로보다 클 경우 SWAP 게이트 연산을 통해 재배치하여 CNOT 게이트 연산을 수행한다. 해당 SWAP 게이트 출력문이 출력되며 최종적으로 CNOT(Control, Target, data element)가 출력된다.

5) Hadamard 게이트

- 명령어 : Hada(ABB_name, target, opt)
- 옵션 : p - Hadamard 게이트 연산이 임의빌딩블럭 내 원소 수준에서 수행
  l - 임의빌딩블럭에 논리적 Hadamard 게이트 연산을 수행
- 옵션이 p일 경우 H(target)가 출력되며 옵션이 l일 경우 임의빌딩블럭 내 모든 데이터 원소에 대하여 H(data1)~H(data$n$)이 모두 출력된다.

6) Z 게이트

- 명령어 : Z(ABB_name, target, opt)
- 옵션 : l - Z 게이트 연산이 임의빌딩블럭 내 원소 수준에서 수행
  p - 임의빌딩블럭에 논리적 Z 게이트 연산을 수행
- 옵션이 p일 경우 Z(target)가 출력되며 옵션이 l일 경우 임의빌딩블럭 내 모든 데이터 원소에 대하여 Z(data1)~Z(data$n$)이 모두 출력된다.

7) X 게이트

- 명령어 : X(ABB_name, target, opt)
- 옵션 : l - X 게이트 연산이 임의빌딩블럭 내 원소 수준에서 수행
  p - 임의빌딩블럭에 논리적 X 게이트 연산을 수행
- 옵션이 p일 경우 X(target)가 출력되며 옵션이 l일 경우 임의빌딩블럭 내 모든 데이터 원소에 대하여 X(data1)~X(data$n$)이 모두 출력된다.

8) Y 게이트

- 명령어 : Y(ABB_name, target, opt)
- 옵션 : l - Y 게이트 연산이 임의빌딩블럭 내 원소 수준에서 수행
  p - 임의빌딩블럭에 논리적 Y 게이트 연산을 수행
- 옵션이 p일 경우 Y(target)가 출력되며 옵션이 l일 경우 임의빌딩블럭 내 모든 데이터 원소에 대하여 Y(data1)~Y(data$n$)이 모두 출력된다.

9) S 게이트

- 명령어 : S(ABB_name, target, opt)
- 옵션 : l - S 게이트 연산이 임의빌딩블럭 내 원소 수준에서 수행
  p - 임의빌딩블럭에 논리적 S 게이트 연산을 수행
- 옵션이 p일 경우 S(target)가 출력되며 옵션이 l일 경우 임의빌딩블럭 내 모든 데이터 원소에 대하여 S(data1)~S(data$n$)이 모두 출력되며 논리적 Z 게이트 연산이 페이즈 조정을 위해 추가적으로 수행된다.

10) T 게이트

- 명령어 : T(ABB_name, target)
- 원소 단위의 T 게이트 연산으로 T(target)를 출력한다.

11) Cat state preparation

- 명령어 : Catprep(ABB_name)
- 7 큐빗 캣 상태를 준비하는 기능으로 3 큐빗이 검증 과정에 사용된다. 매회 만들어진 캣 상태가 검증을 통과할 때까지 반복 수행된다.

12) Cat state decoding
- 명령어 : Catdec(ABB_name)
- 7 큐빗 캣 상태를 디코딩하는 연산으로 준비과정을 역연산 한 뒤 첫 번째 큐빗을 Z basis로 측정하여 해당 결과를 이용한다.

13) Controlled-M 게이트
- 명령어 : ConM(ABB_name, oper)
- 옵션 : Controled M 게이트 연산을 위한 게이트 스트링
- 결함허용적 ConM 게이트 연산으로 7 큐빗 캣 상태가 ancilla 상태로 사용된다. 각 데이터 원소에 대응되는 7 큐빗 캣 상태의 원소 위치에 따라 SWAP 게이트를 통해 위치를 조정하여 해당하는 SWAP 게이트 출력문을 출력한다. 최종적으로 ConM(control, target, operation, position of control, position of target)을 출력한다.

14) Relayout 연산
- 명령어 : Relayout(ABB_name)
- 임의빌딩블럭의 원소들을 최초의 배열로 되돌리는 연산이다. 각 원소들을 원래 위치로 되돌리는데 SWAP 게이트 연산이 사용되며 이에 해당하는 출력문이 출력된다.

15) Measurement
- 명령어 : Meas(ABB_name, target, opt)
- 옵션 : z - 임의빌딩블럭 내의 원소를 Z basis로 측정
  x - 임의빌딩블럭 내의 원소를 X basis로 측정
- 임의빌딩블럭 내 원소 수준의 측정 연산으로 Meas(target, opt)를 출력한다.

16) Logical Measurement
- 명령어 : Lmeas(ABB_name, opt1, opt2, oper)
- 옵션1 : c - 임의빌딩블럭이 Z,X와 같은 computational basis로 관측
  n - 임의빌딩블럭이 임의의 연산에 대해서 관측
- 옵션1 : z - 임의빌딩블럭을 Z basis로 측정
  x - 임의빌딩블럭을 X basis로 측정
- oper : 임의의 연산에 대한 스트링
- 논리적 measurement 연산으로 각 옵션에 따라 임의빌딩블럭 내 모든 데이터 원소에 Meas(target, opt) ~ Meas(target, opt)를 출력한다.

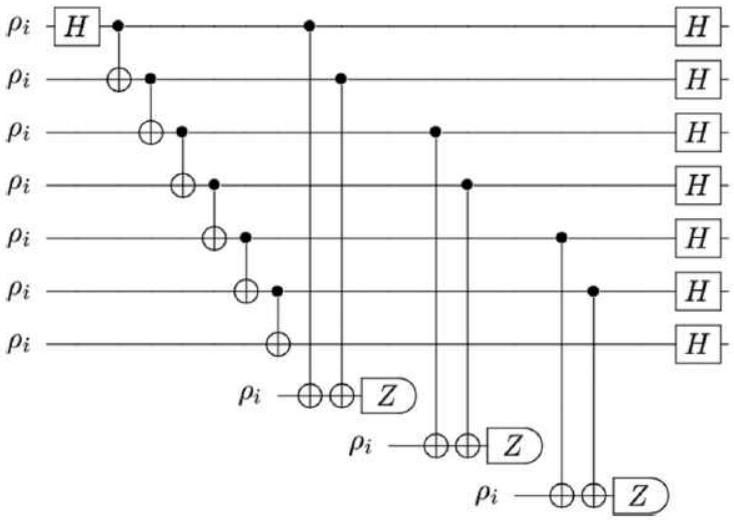


그림 25. 7-큐빗 Shor 상태 생성 회로도 [Weinstein13]

17) 7-qubit Shor state preparation
- 명령어 : Shor7prep(ABB_name)
- 7 큐빗 Shor 상태 준비 과정으로 7 큐빗 Shor 상태가 제대로 준비되었음을 검증하기 위해 3 큐빗이 사용된다.

18) 7-qubit Shor state decoding
- 명령어 : Shor7dec(ABB_name)
- 7 큐빗 Shor 상태를 측정하여 디코딩하는 연산으로 해당 신드롬의 패리티를 출력한다. 패리티가 짝수일 경우 오류가 발생하지 않은 것이고 패리티가 홀수일 경우 에러가 발행한 것이다.

19) Magic state preparation
- 명령어 : MState_prep(ABB_name)
- 논리적 0 상태와 7 큐빗 Shor 상태를 사용하여 논리적 T 게이트 연산을 위한 Magic 상태를 준비하는 기능이다. 논리적 0 상태에 7 큐빗 Shor 상태를 control로 하여 Controlled-ZSX 게이트 연산을 하고 이에 해당하는 출력문들을 출력한다. 이후에 7 큐빗 Shor 상태를 디코딩하여 신드롬의 패리티를 확인한다. 이 과정을 같은 신드롬이 나올 때까지 두 번 이상 반복한다.

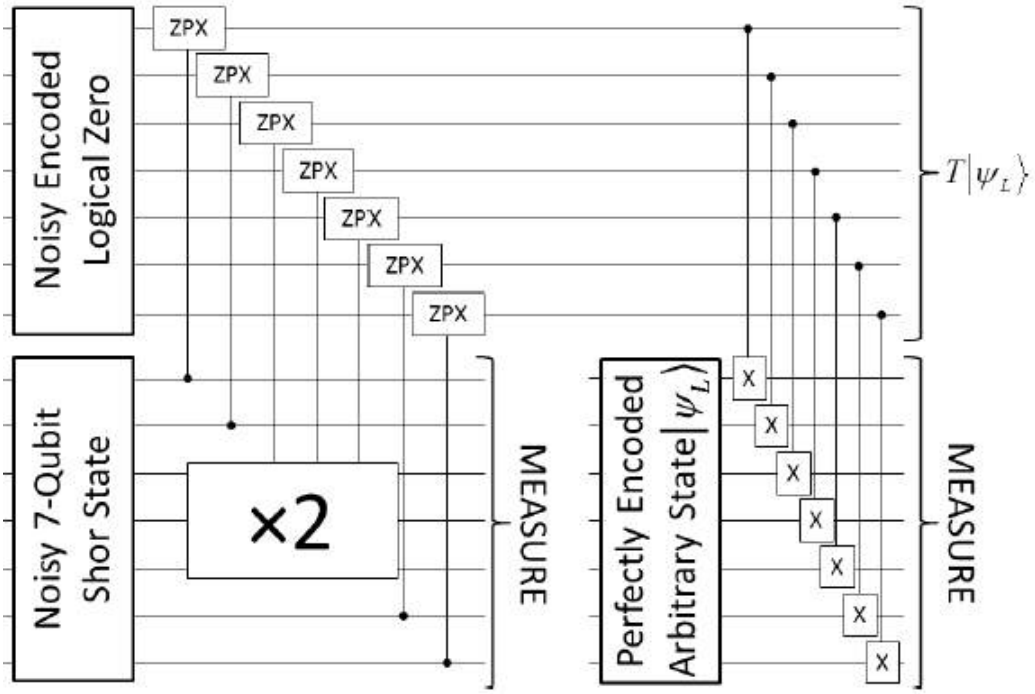


Figure 1. Implementation of [[7,1,3]] QEC code T-gate.

그림 26. 논리적 T 게이트 연산 회로도 [Weinstein15a]

20) 논리적 T 게이트

- 명령어 : LogT(ABB_name)
- 논리적 T 게이트 연산으로 MState_prep(ABB_name)의 결과를 사용한다. Magic 상태와 데이터 원소 간에 CNOT 게이트 연산해준 뒤 해당 임의빌딩블럭을 논리적 Z basis 측정을 한다. 측정 결과가 0이면 논리적 T 게이트 연산이 제대로 수행된 것이며 1일 경우 magic 상태에 논리적 X 게이트 연산을 해줄 경우 논리적 T 게이트 연산이 제대로 수행된다. 각 과정에 사용되는 연산들의 출력문을 출력한다.

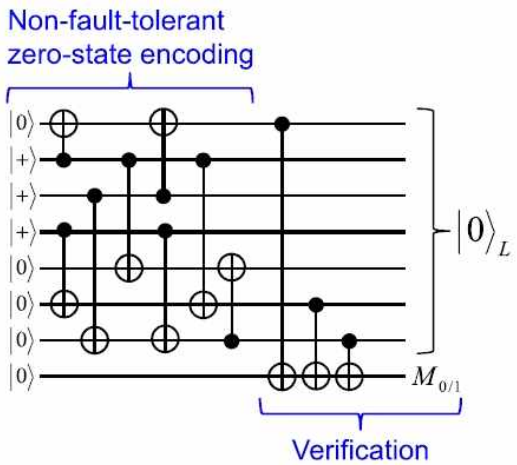


그림 27. Steane 부호의 논리적 0 상태 준비 회로 [Goto16]

21) Preparation

- 명령어 : Prep(ABB_name)
- 논리적 0 상태를 준비하는 기능이다. 1개의 큐빗이 검증 과정에 사용되며 검증 과정을 통과할 때까지 반복 수행한다.

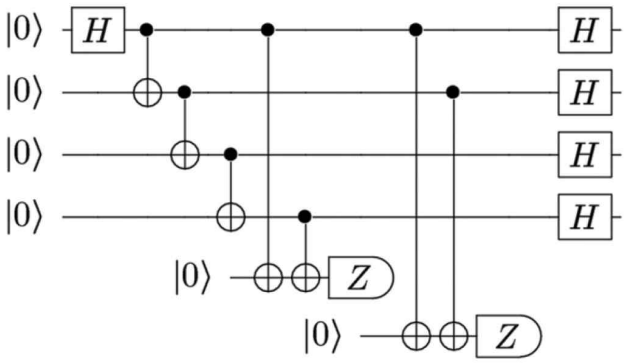


그림 28. Syndrome 측정에 사용되는 4-큐빗 Shor 상태 생성 회로도 [Weinstein12]

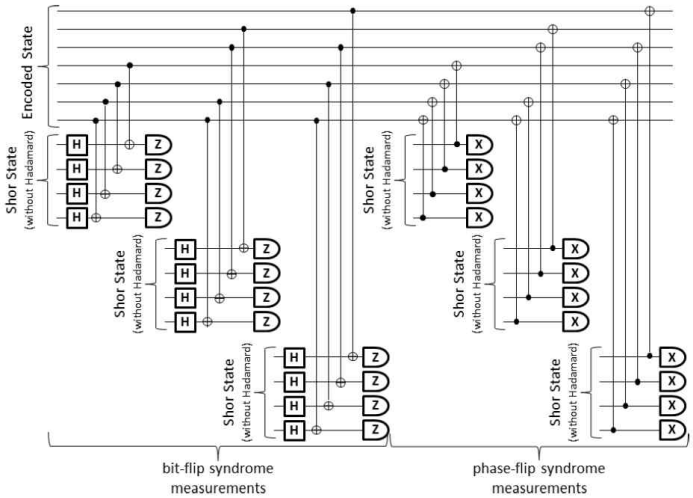


그림 29. 4-큐빗 Shor 상태를 사용한 신드롬 측정 회로도 [Weinstein15b]

22) Syndrome Measurement
- 명령어 : Synd_Meas(ABB_name)
- 4 큐빗 Shor 상태를 사용하여 해당 임의빌딩블럭의 신드롬을 추출한다.

23) Error correction operation
- 명령어 : err_cor(ABB_name)
- 해당 임의빌딩블럭을 오류 정정하는 기능이다. 신드롬 측정 기능을 사용하여 같은 신드롬을 얻을 때까지 최대 3회 반복 수행하며 3회 수행 시 각 신드롬들을 majority rule에 따라 결정한다. 추출 된 신드롬에 대응되는 오류 정정 연산을 임의빌딩블럭의 데이터 원소에 연산하며 각 과정에 사용되는 기능들의 출력문을 출력한다.

◇ 성능

- 컴파일 부분에서 입력되는 최대허용오류율에 따라 임의빌딩블럭의 성능이 결정된다. 일례로 $m=40$, $b=3$인 Ground State Estimation 알고리즘의 경우 임의빌딩블럭의 성능은 다음과 같다.

임의빌딩블럭 최고 level	3	
임의빌딩블럭 크기	27000μm^2	
성능	오류율	시간(sec)
QEC	2.8065e-23	892.932
Preparation	3.1941e-24	967.037
SWAP	5.8232e-20	1021.569
CNOT	1.7363e-15	1021.569
Measurement	-	1251.507
Controlled-ZSX	2.7723e-14	931.468
H	1.1926e-43	931.468
Z	1.1926e-43	931.468
X	1.1926e-43	931.468
Y	1.1926e-43	931.468
S	1.4226e-38	931.468
T	2.9608e-38	1883.460

표 1. GSE M=40, b=3에서의 임의빌딩블럭 성능

2.3 비용효율성 향상 연구개발 현황

- 컴파일 수준에서의 비용분석

◇ 비용 산출 방식

- 양자 어셈블리어 수준에서의 시간 비용 = 전체 게이트 수 x 기본 게이트의 예상 작동 시간(1 ns)
- GSE 알고리즘의 경우 전체 게이트 수와 step수(critical path)의 차이가 크지 않다. 이것은 그만큼 병렬 계산의 효과가 작다는 것을 의미한다. 따라서 GSE 알고리즘의 컴파일 과정에서 계산되는 비용은 주로 전체 게이트 수로 계산될 수 있다. 현재 GSE 알고리즘의 컴파일을 통한 전체 게이트 수는 M=40, b=3 일 때 약 1.2×10^{10} 정도 계산된다.

◇ 입력크기 M에 따른 어셈블리코드의 변화

- GSE 알고리즘의 경우, 입력크기 M에 따라 필요로 하는 어셈블리코드가 점진적으로 변화한다. 이러한 경향은 M=2에서부터 M=40까지 2 간격으로 구체적인 데이터를 생성해 봄으로써 알 수 있다(그림 30. 참고).

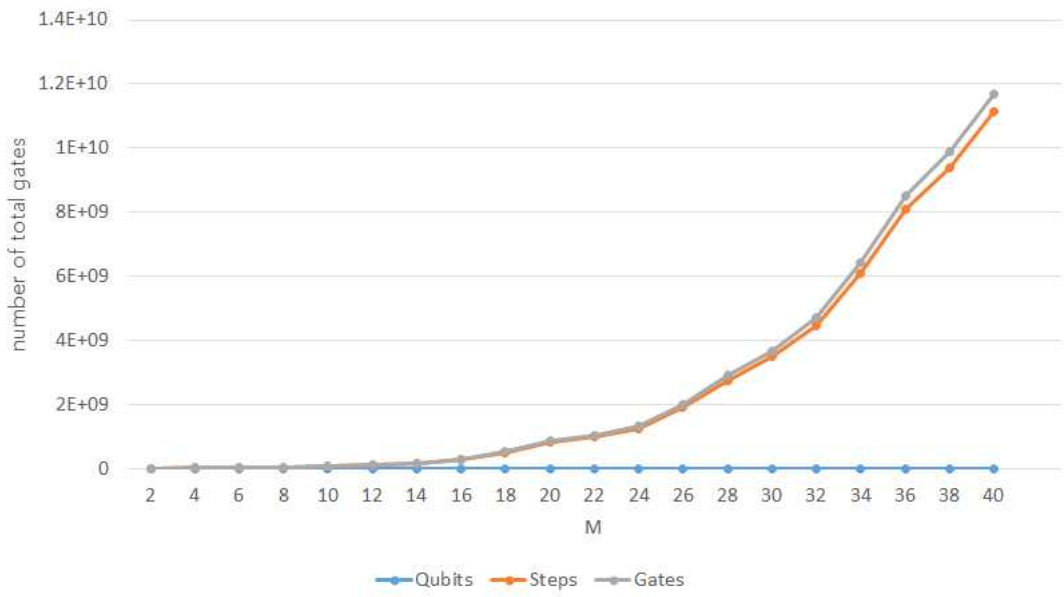


그림 30. 입력크기 M에 따른 어셈블리코드의 변화

- 특히, M=40에 대한 구체적인 컴파일 결과는 다음의 표 2.에서 확인할 수 있다.

출력값의 형태	출력 데이터
큐빗 수	43 큐빗
qasmh 파일의 크기	40.5 MB
qasmf 파일의 크기	248.4 GB
전체 게이트 수	11,708,037,960
critical path의 길이	11,125,233,307
최대허용오류률(KQ 모델)	$2.09036640507 \times 10^{-12}$
최대허용오류률(향상된 KQ 모델)	$2.40661934825 \times 10^{-11}$
qasmf의 시각화	83개의 단일 큐빗 게이트 시각화 가능

표 2. GSE 알고리즘의 M=40에 대한 컴파일 결과들에 대한 데이터

- 표 2.에서 볼 수 있듯이 GSE 알고리즘의 M=40의 컴파일 결과 중 하나인 qasmf 파일을 크기를 보면, 248.GB 임을 알 수 있다. GSE 알고리즘의 입력값 M이 증가함에 따라서 qasmf에 대한 저장 공간의 크기는 급격하게 증가하므로 이를 저장하기 위해서는 많은 양의 저장 공간이 필요하다. 또한, 그림 31.에서 볼 수 있듯이, GSE 입력값 M=100에 대한 컴파일 시간은 275시간, 즉, 11일 넘게 걸리는 것을 볼 수 있다. 따라서 GSE 알고리즘의 큰 입력값에 대한 계산을 위해서는 많은 지금의 컴퓨팅 자원이 필요하다. 이러한 상황에서 보면, 슈퍼컴퓨터의 사용이 이러한 문제를 일정 부분 해결해 줄 수 있을 것으로 기대된다.

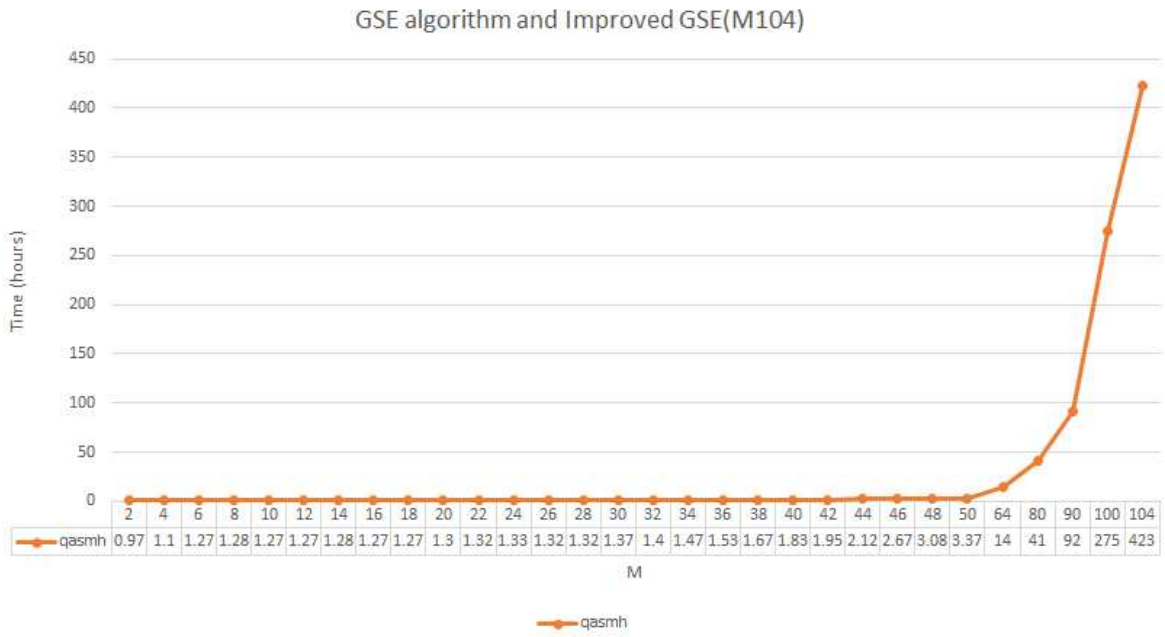


그림 31. GSE 알고리즘의 컴파일 시간

◇ 비용에 영향을 끼치는 주요 인자들

(1) 임의의 회전 각도에 대한 단일 큐빗 게이트의 근사적 분해 방법

- 현재 주로 사용되고 있는 근사적 분해 방법은 Grid synthesis 분해 방법과 SQCT 분해 방법의 두 가지이다. Grid synthesis 분해 방법이 더 최근에 개발된 방법으로 더 효율적이며, 실행 시간도 더 빨라서, 현재 ScaffCC 컴파일 프로그램에는 기본적으로 grid synthesis가 지정되어 있다.

(2) 특정한 회전 각도에 대한 단일 큐빗 게이트의 정확한 분해 방법

- 단일 큐빗 게이트를 근사적으로 분해하면 일반적으로 수백 개의 게이트들로 분해가 된다. 하지만, 경우에 따라서는 정확한 계산을 통하여 근사적인 방법인 아닌 직접적인 분해를 통하여 분해되는 게이트의 수를 급격하게 줄 일 수 있다.

(3) 입력크기 M

- GSE 알고리즘의 경우, M=10 이 기본으로 설정되어 있다. 그러나 지금의 컴퓨터보다 효율적으로 판단되는 M은 문헌들에 따르면 M=50 정도부터 나타난다. 특히, M=208의 분자의 경우, 광합성 연구에 중요한 부분으로 이에 대한 많은 연구가 진행되고 있다.

(4) 측정 큐빗의 수 b

- GSE 알고리즘의 경우, Phase Estimation 알고리즘을 통하여 유니터리 행렬의 고유치를 계산하는 과정을 필요로 한다. 이때, 그 고유치를 계산하기 위하여 필요한

큐빗의 수가 b로 계산되며, 이러한 b의 크기는 알고리즘의 정확도에 영향을 미친다. 또한 b에 증감에 따라서, 전체 게이트의 수가 지수적으로 변한다.

(5)단일 큐빗 게이트의 근사적 허용 한계(컴파일 에러 d)

- 양자 연산인 유니터리 행렬은 항상 정확하게 작은 유니터리 행렬들(고정된 몇 개의 유니버설 게이트들)로 분해될 수 있는 것은 아니다. 따라서 근사적인 계산이 필요하다. 이때 근사적인 계산의 기준이 d라는 파라미터로 결정된다.

- 시스템 합성 부분

◇ 비용 산출 방식

- 시스템 합성 파트에서의 주요 비용은 양자 컴퓨팅 과정에서 발생하는 통신량과 모듈 배치 방법론에 따른 구동에 필요한 큐빗의 개수이다.

(1) 통신량

- 양자 어셈블리 코드는 양자 알고리즘의 또 다른 표기 방식으로 코드 생성과정에서 양자 컴퓨팅 시스템 구조 등이 전혀 반영되지 않는다. 반면, 시스템 빌드업 과정을 통해서 생성된 시스템 코드는 특정 양자 컴퓨팅 시스템의 구조에 기반을 둔 어셈블리 코드라 할 수 있고 몇몇 상황에서 양자 컴퓨터 시스템의 구조에 매우 종속적이 된다. 예를 들면, 멀티 큐빗 게이트의 동작과 서브 모듈의 호출과 관련하여 큐빗의 이동이 필요한 경우가 발생하고 특정한 이동 경로를 통해서 이동하게 된다. 큐빗의 이동은 원래 양자 알고리즘에는 존재하지 않는 부분으로 시스템 빌드업 과정에서 추가된 비용이다. 큐빗의 이동과 관련된 SWAP 게이트의 개수로 정의하며, 빌드업 과정에서 통계 수치 분석 기능을 통해서 산출할 수 있다.

(2) 구동에 필요한 큐빗 수

- 현재 모듈의 배치와 관련하여 본 연구실에서 접근하고 있는 방식은 내포된 모듈 구조(Nested Module)이다. 양자 어셈블리 코드의 모듈들은 양자 게이트들만으로 구성된 단말 모듈과 서브 모듈의 호출을 포함하는 비단말 모듈로 구성되어 있다. 따라서 단말 모듈을 구동할 때는 해당 모듈의 큐빗들만 존재하면 되지만, 비단말 모듈을 구동할 때는 서브 모듈들의 큐빗들도 필요하다. 이 문제를 해결하기 위해서 현재는 임의의 모듈이 서브 모듈들의 큐빗들까지 포함하는 방식인 내포된 모듈 구조를 이용하고 있다. 이 방식을 이용하면 main 모듈에서 직접 호출하는 상위 단계의 모듈들의 경우 매우 많은 큐빗들을 필요로 하게 된다. 빈번하게 사용되는 단말 모듈들의 큐빗들이 해당 상위 단계 모듈들에 중복되어 존재하기 때문이다. 큐빗의 수가 충분한 양자 컴퓨터 시스템을 가정한다면 해당 방식이 알고리즘 구동 속도 측면에서 효과적이겠지만 그렇지 않은 상황에서는 큐빗들이 매우 낭비된다고 말할 수 있다. 따라서 nested module 구조상에서 구동에 필요한 큐빗의 수 또한 양자 컴퓨팅 비용으로 선정하였다.

◇ 입력크기 M에 따른 어셈블리코드의 변화

- 현재 플랫폼 성능의 기준이 되는 GSE 알고리즘을 기준으로 입력값의 증가에 따라서 SWAP 게이트의 수, 비율은 점차 증가하는데, 구체적인 수치는 아래 그림 32.과 그림 33.에서 확인할 수 있다.

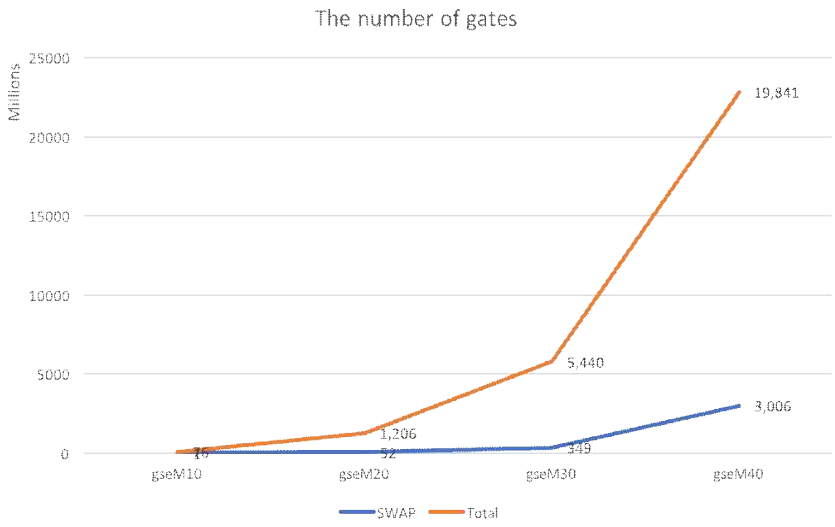


그림 32. GSE 알고리즘의 입력 크기(M=10~40)에 따라 전체 양자게이트의 개수와 SWAP 게이트의 개수

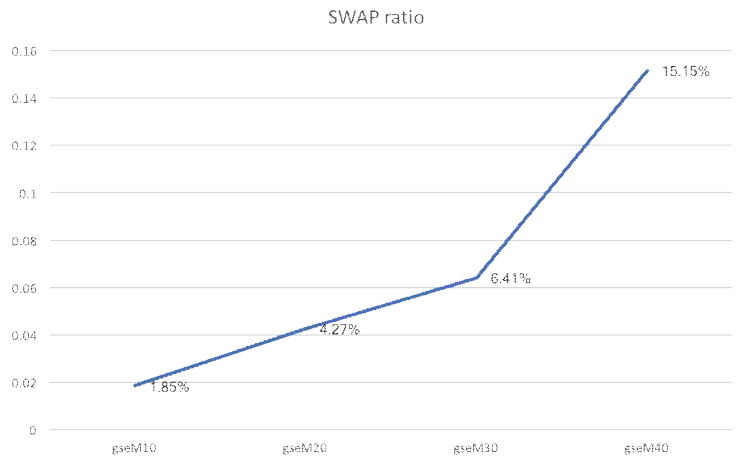


그림 33. GSE 알고리즘의 입력 크기(M=10~40)에 따라 SWAP 게이트의 비율(#SWAP/#Total)

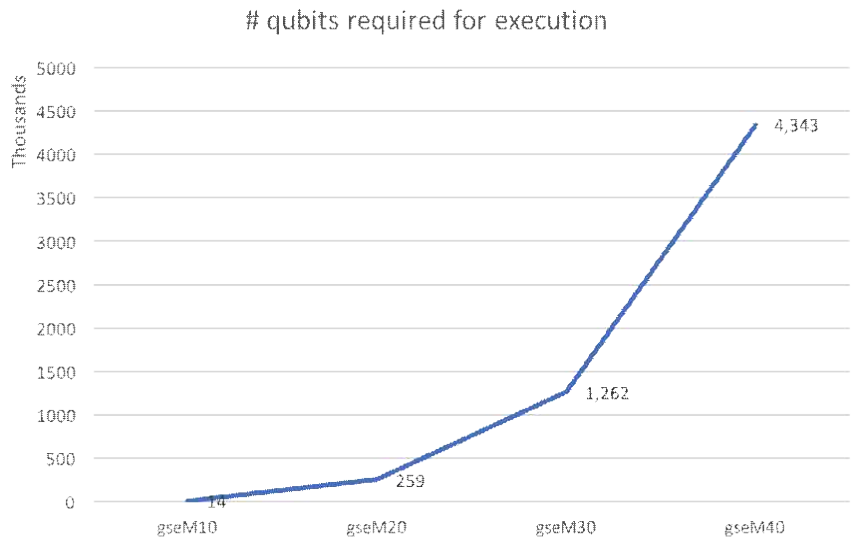


그림 34. Nested module 구조상에서 GSE(M=10~40)를 구동하기 위해 필요한 논리적 큐빗(ABB)의 수

◇ 비용에 영향을 끼치는 주요 인자들

(1) 통신량

- 큐빗의 이동에 영향을 끼치는 것은 모듈 내 큐빗들의 초기 배치이다. 그리고 모듈들 간의 배치가 중요한 영향을 끼친다. 따라서 큐빗들의 배치와 모듈들 간의 배치를 개선함으로써 큐빗의 이동 비용을 줄일 수 있다.

(2) 구동에 필요한 큐빗 개수

- 구동에 필요한 큐빗의 개수는 사용하는 모듈간의 배치 방식과 관련이 있다. 현재의 Nested module 구조상에서는 임의의 모듈이 서브 모듈에서 사용되는 모든 큐빗들을 포함하므로 알고리즘의 구동 전체적으로 큐빗들의 중복이 많다고 할 수 있다. 반면 모듈의 배치 전략을 개선함으로써 빈번하게 사용되는 단말 모듈의 경우 여러 상위 단계의 모듈들이 공유하여 사용할 수 있게 함으로써 구동에 필요한 큐빗의 개수를 감소시킬 수 있다.

◇ 비용효율성 향상 연구결과 : 서로 다른 양자 오류 정정 부호로 부호화 된 논리적 벨 양자 상태 구현 방법 특허 국내 출원

- 본 특허는 양자 부호 전송이라는 프로토콜의 핵심 자원인 논리적 벨 양자 상태를 구현하는 방법을 제안한다. 양자 부호 전송은 양자 오류 정정 부호 A 로 부호화된 양자 정보를 일련의 양자 연산을 통해서 양자 오류 정정 부호 B 로 부호화된 양자 정보로 변환을 시키는 프로토콜이다. 그리고 이 양자 부호 전송을 위해서는 논리적 벨 양자 상태라는 특수한 양자 얽힘 상태가 필수이다. 현재까지 양자 부호 전송 프로토콜의 중요성 및 활용성에 대한 연구 결과는 일부 있었지만, 그 핵심 자원인 논리적 벨 양자 상태를 구현하는 연구는 2016년 일본의 연구팀에서 발표한 방법이 유일하

다. 해당 방법은 임의의 연쇄(concatenation) 수준의 논리적 벨 양자 상태를 구현하기 위해서 수준 1 단계부터 임의 단계까지 양자 인코딩 & 얽힘 정화(entanglement purification)이라는 방법을 적용하여 논리적 벨 양자 상태를 생성한다. 얽힘 정화는 순도가 낮은 다수의 양자 얽힘 상태를 입력으로 하여 순도 높은 극소수의 양자 얽힘 상태를 만들어 내는 프로토콜로 그 비용(입력으로 필요한 양자 상태의 수, 필요한 양자 연산의 수)이 높은 프로토콜로 알려져 있다. 따라서 얽힘 정화의 비용을 줄이기 위한 연구들이 그동안 많이 진행되어 왔다.

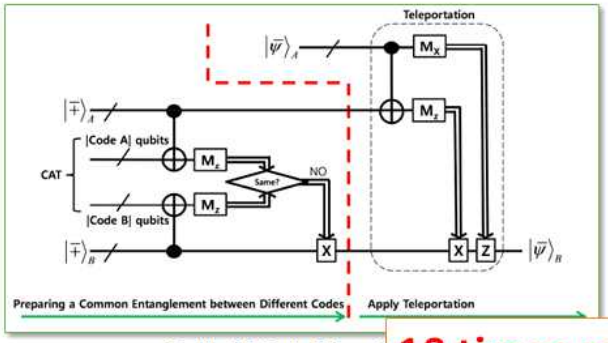
- 본 특허에서 우리는 1단계의 논리적 벨 양자 상태를 구현하기 위해서 얽힘 정화를 필요로 하지 않는 방법을 제안한다. 임의의 연쇄 단계 논리적 벨 양자 상태를 구현하는 방식은 귀납적인 접근방법으로, 연쇄 i 단계의 논리적 벨 양자 상태를 구현하기 위해서 i-1 단계의 논리적 벨 양자 상태가 다수 필요하다. 따라서 1 단계의 논리적 벨 양자 상태를 구현하는데 있어서 양자 얽힘 정화를 이용하지 않음으로써 기존의 방법에 비해서 비용 감소 효과가 충분히 있다.

- 본 특허의 핵심 원리는 벨 양자 상태가 XX 와 ZZ 라는 안정화 연산자(stabilizer operator)를 갖는다는 점이다. 따라서 XX 와 ZZ 연산자에 안정화되어 있는 양자 상태를 만드는 데, 먼저 XX 라는 연산자에 안정화되어 있는 상태를 준비한다. 이 상태는 양자 컴퓨팅에 많이 사용되는 초기화 상태로 해당 상태를 효율적으로 만드는 방법들이 많이 알려져 있다. 그 이후 해당 상태를 ZZ 안정화 연산자로 측정을 한 뒤, 안정화되어 있는 상태만을 취하면 된다. ZZ 안정화 연산자로 측정을 하는 것은 일련의 CNOT 연산을 수행한 뒤, 양자 상태 벡터들의 패리티를 확인하여 패리티가 짝수이면 취하고, 홀수 이면 사후 연산(post-processing)을 통해서 짝수 패리티를 갖도록 보정을 하게 된다. 이런 일련의 과정에서 일부 양자 오류가 발생할 수 있으므로 주기적으로 오류 정정을 수행하고, 또 필요시 반복 수행하여 1 단계의 논리적 벨 양자 상태를 구할 수 있다.

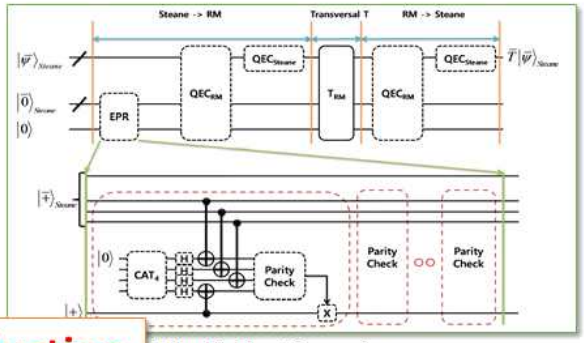
- 다음 그림은 본 특허를 이용하는 경우 Bacon-Shor 기반에서의 논리적 T 게이트를 Steane 기반에서의 논리적 T 게이트로 바꾸는 과정을 예시한다. 그림에서 알 수 있듯이, 코드 변환 방법을 이용하는 경우에는 Bacon-Shot 코드에서의 T 게이트 비용이 10배 이상 감소하는 것을 확인할 수 있다.

목표 1: 비용효율성 1000배 향상 기술 (2/3) – 실적

- **(결함허용 방식으로 Hetero EPR Preparation 적용, 동작 시간 10배 감소) 국내, 국외 3극 특허 출원 중**
  ※ Bacon-Shor 코드기반 결함허용 방식에서 비용(시간)이 높은 Non-Transversal Gate 를 비용이 낮은 Reed-Muller 코드 혹은 Steane코드 등의 반영하여 Transversal Gate로 치환
- **(컴파일, 시스템, 빌딩블럭 등에서는 비용 개선 확인 중)**



Method 1: Code Teleport



Method 2: Direct Conversion

10 times reduction

Error Rate(ϵ)	Single-code model		Dual-code models with		
	f=1.0	f=0.9	RM Code	Steane Code	RM $T\|+\rangle$ state
0.000001	5,143	26,287	2,814	2,665	2,093
0.00001	5,157	26,264	2,816	2,666	2,095
0.0001	5,304	27,046	2,838	2,677	2,113
0.001	7,045	36,306	3,070	2,788	2,307

- 임의빌딩블럭 부분

◇ 비용 산출 방식

- 성능평가조합 : 1개의 소자기술, 1개의 양자오류정정부호 대상으로 가능
- 사용되는 기본빌딩블럭의 수 : $30 \times (25)^{l-1}$
- 임의빌딩블럭의 크기 : $30 \times (25)^{l-1} \times 4(\mu m^2)$
- 각 기능별 수행 step 수

기능	Step 수
QEC	185
Preparation	192
SWAP	237
CNOT	237
Measurement	249
Controlled-ZSX	185
H	185
Z	185
X	185
Y	185
S	185
T	377

표 3. 임의빌딩블럭의 기능별 step 수

- 오류율 산출 방식 : 각 연산에 있어서 k(연산에 사용되는 하위 블록의 수), q(연산의 step 수)라 할 때 오류율 p_e는 다음과 같다

$$p_e = 1/_{kq}C_2$$

◇ 비용에 영향을 끼치는 주요 인자들

- 크기 측면 : 임의빌딩블럭을 생성하는데 쓰이는 기본빌딩블럭의 숫자. 최상위 레벨의 임의빌딩블럭과 하위 레벨의 임의빌딩블럭의 배열을 다르게 사용함으로써 크기 측면에서의 비용을 일정부분 감소시켰다.
- 시간 측면 : 각 기능별 step 수. 임의빌딩블럭의 배열을 최적화 할 경우 감소시킬 수 있을 것으로 예상된다.
- 성능 측면 : 현재의 성능 산출 과정으로는 연산에 사용되는 하위 블록의 수와 기능별 step에 영향을 받는다.

◇ 비용효율성 향상 연구결과: Analysis of Physical Requirement of Simple 3-qubit and 9-qubit QECs on Quantum-Dot and Superconductor Qubit 논문 게재 승인

- Quantum dot과 superconductor 큐빗에서의 3-큐빗, 9-큐빗 양자오류정정부호를 분석하고 이를 통해 양자오류정정부호의 실행 주기를 결정하는 연구를 진행하였다.

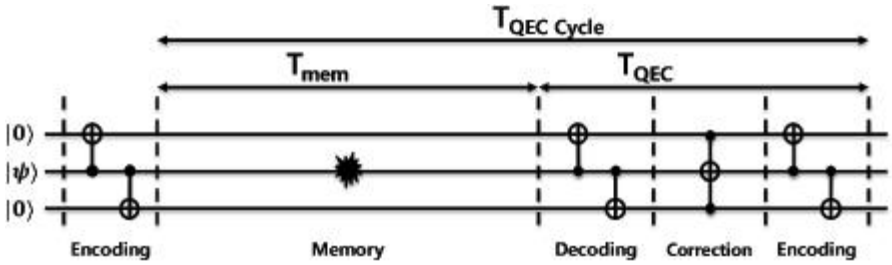


그림 36. 3-큐빗 양자오류정정부호의 주기, 메모리 시간, 양자오류정정부호 소모 시간

- 양자오류정정부호가 인코딩된 경우의 fidelity가 물리적 큐빗 그대로의 fidelity보다는 좋아야한다는 것을 기준으로 최소 메모리 시간, 최대 메모리 시간, 최적 메모리 시간을 구하고 최적 메모리 시간과 양자오류정정부호 소모 시간을 통해 양자오류정정부호의 수행 주기를 구하였다. 또한 최소 메모리 시간을 통해 물리적 큐빗의 에러 threshold를 구하였다.

변수	조건
$T_{memory,\min}$	$2T_{QEC}$
$T_{memory,best}$	$\dfrac{1-\sqrt{1-\frac{2}{3}(1+\frac{1}{9}P_e T_{QEC})}}{2P_e}$
$T_{memory,\max}$	$\dfrac{1}{2P_e}$
$T_{QEC\ cycle}$	$0.211324\dfrac{1}{P_e}$
$P_{e,th}$	$\dfrac{1}{3T_{QEC}}$

표 4. 3-큐빗 양자오류정정 부호의 메모리 시간, 주기, 에러 threshold

- 해당 결과를 이용하여 Quantum dot과 superconductor 큐빗에서의 3-큐빗 양자오류정정부호의 gain, 최소 메모리 시간, 물리적 큐빗 에러 threshold를 시뮬레이션 하였다.

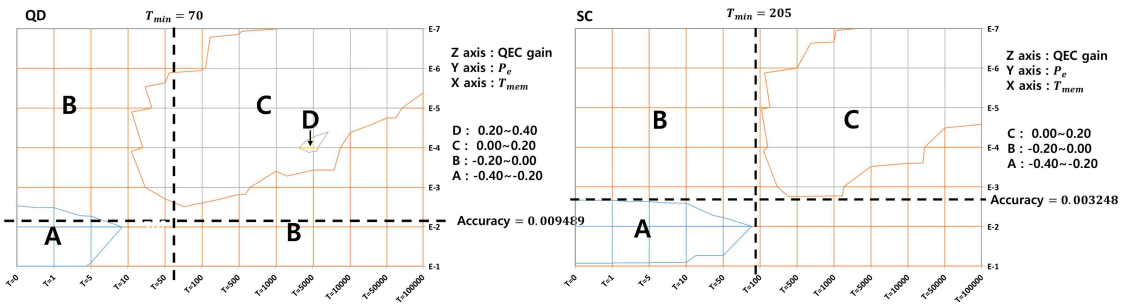


그림 37. 3-큐빗 양자오류정정부호의 QD, SC에서의 이득, $T_{\min}$, $P_{e,th}$ 그래프

- 시뮬레이션을 통해 수식을 통해 분석한 결과와 시뮬레이션 결과가 잘 부합하는 것을 확인하였고 이를 통해 물리적 큐빗의 성능과 양자오류정정부호 주기를 수식적 분석을 통해 제시할 수 있음을 확인하였다.

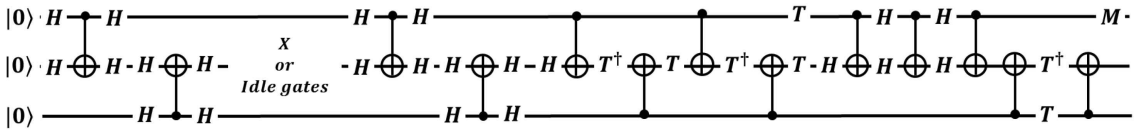


그림 38. IBM test에 사용한 3-큐빗 양자오류정정부호 회로

- 분석 결과를 IBM Quantum Experience를 통해 실질적인 시뮬레이션을 하였다. 최소 메모리 시간보다 적은 메모리 시간을 가할 경우 인코딩 된 큐빗이 그렇지 않은 물리적 큐빗보다 fidelity가 낮음을 확인하였으나 IBM Quantum Experience가 충분한 Idle 게이트를 지원하지 않아 최소 메모리 시간 이상을 확보할 수 없어 완벽한 시뮬레이션은 불가능하였다.

Fidelity	F_p	F_e
Bit flip 부호	0.984	0.852
Phase flip 부호	0.973	0.860

표 5. 최소 메모리 시간 이하의 메모리 시간에서의 양자오류정정부호 이득

- 해당 연구 결과는 양자정보분야 저널인 Physical Review A에 제출하여 게재 승인을 받았다.

2.4 연구개발 목표 달성정도

- 컴퓨팅 플랫폼 완성도
  - 빌드업 기능 측면에서는, 플랫폼의 주 구성요소인 양자컴퓨팅 운영체제는 컴파일러, 시스템합성기능, 임의 빌딩블럭, 기본 빌딩블럭 합성 기능을 모두 구현하였다.
  - 구동 기능 측면에서는, 시스템 수준에서 시스템 코드의 생성은 가능한 수준까지 개발되었다.

- 비용효율성 향상 정도
  - 결함허용 방식에서 10배정도 향상하는 방법을 제안하였다.
  - 컴파일과정, 시스템합성과정, 빌딩블럭합성 과정에서는 현재까지 개발된 분석 기능을 토대로 비용분석을 진행하였다.

- 양자컴퓨팅 성능 평가 조합
  - 현재까지는 3개의 양자알고리즘, 1개의 오류보정코드, 1개의 소자기술을 지원한다.

현재까지의 연구목표 달성도 (연구개발 주요성능 Spec)

성과지표 (주요성능 Spec)	단위	세계최고수준	기술개발 목표치('16)	목표치 산출근거	검증방법	현재 상황
① 컴퓨팅 플랫폼 완성도	100%	부분별 플랫폼 수준	**컴파일러와 시스템 수준으로써 대략 60%**	컴파일러 내재화, 시스템 수준 합성 툴 개발	사용자 수준에서의 양자컴퓨터 프로그램 환경에서의 평가	• 빌드업기능측면: 컴파일러 시스템빌더, 임의블럭 빌더 기본블럭 빌더 모두 확보 (100%) • 구동기능측면: 시스템 수준 설계 완료 세부 블럭별 구동기능 개발중 (10%) • 전체적으로 60% 수준 달성
② 비용효율성 향상 정도	배율	부분별 최소화 수준	**컴파일러, 시스템, 결함허용 수준에서 각각 10배정도**	단일게이트 분해과정, 결함허용방식의 적용횟수, 코드변환등의 방법 적용	제안된 방법의 적용 여부에 따른 평가	• 컴파일과정 검증중 (50%) • 시스템 최적 배치 고안중 (20%) • 임의블럭 최적화 방법 적용중 (30%) • 기본블럭수준 펄스최적화 진행중 (10%) • 전체적으로 33% 수준 달성 1가지(10배감소방법) 방법 제안(33%)
③ 양자컴퓨팅 성능 평가 조합	조합수	3개소자기술, 3개오류보정기술, 3개알고리즘수준	**3개알고리즘, 3개소자기술 적용 수준**	알고리즘과 소자기술 정도 확보	DB조합 추출 및 평가결과 분석	• 3개 알고리즘 평가가능(GSE, Shor, QBF) • 1개 오류보정기술적용(Steane), 2가지 (Bacon-Shor, Surface) 적용중 • 1개 소자기술(양자점), 2가지 소자기술(초전도, 광) 추가중 • 전체적으로 33% 수준 달성

제3장 후 연구개발 계획

3.1 양자컴퓨팅 플랫폼 구축

- 전체 시스템 구조 측면에서의 향후 계획

◇ 기능 측면

- 현재까지 구축된 플랫폼을 확장하여, 양자컴퓨터의 실시간 구동이 가능한 수준으로 확장할 예정이다.

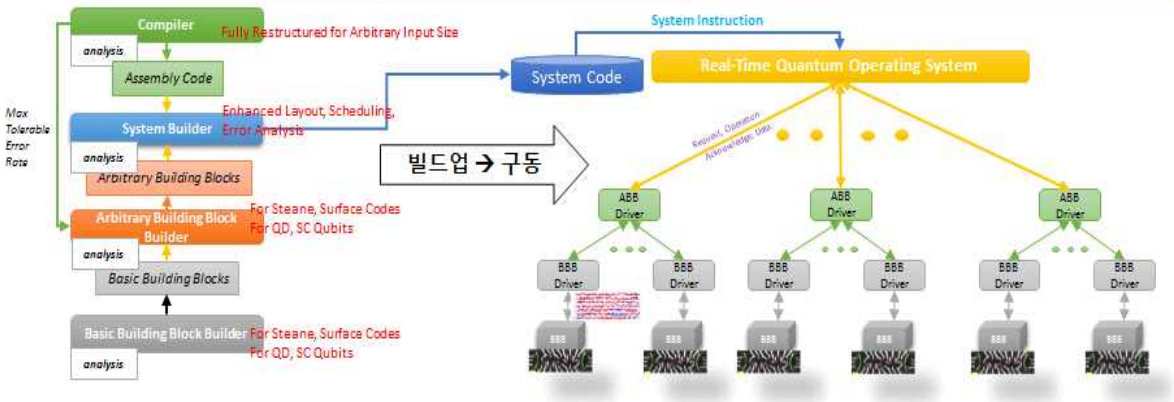
◇ 성능 측면

- 현재 고전 고성능 슈퍼컴이 해결가능한 최대 입력크기를 대상으로 한 양자컴퓨팅의 성능 평가 기능을 추가할 예정이다.

목표 2: 양자컴퓨팅 플랫폼 기술 (3/3) – '17.4월까지 계획

- **(적용범위 확장) 3가지 알고리즘, 3가지 소자기술, 3가지 오류보정 적용으로 확장**
- **(구동기능 추가) 양자 하드웨어를 실시간 제어로 확장**

Platform Part	Parameters	Final Goal (Phase 1)	2016.10	2017.4
Compiler	Algorithms	**Three Algorithms**	GSE (1/3)	Shor, Boolean Function (3/3)
	Input Size	**Arbitrary**	Arbitrary (Completed)	
	Conventional Threshold Input Size	**Support**	Yes (Completed)	
System Builder	Building Approach	**Scalable**	Flatten, Hierarchical (Completed)	
	Input Size	**Arbitrary**	Arbitrary (Completed)	
	Conventional Threshold Input Size	**Support**	Yes (Completed)	
	QEC Code Schemes	**Block Codes, Topological Codes**	Block Code(s) (1/2)	Surface Code (2/2)
Arbitrary/BasicBuilding Block Builder	QEC Codes	**Two Codes(Steane, Bacon-Shor)**	Steane (1/2)	Bacon-Shor (2/2)
	Devices	**Three Devices(QD, SC, Optical,Ion-Trap)**	QD (1/4)	QD, SC, Optical,Ion-Trap (4/4)
	Target Error Rate	**Arbitrary**	Arbitrary (Completed)	
HW Building Block	*3-qubit QEC, QD*	***(HOPE)***	*Designed*	*Confirm*
	5-qubit QEC, QD	***(HOPE)***		*Design*



- 컴파일 부분

◇ 기능 측면

(1) 기본 연산 알고리즘의 프로그래밍 및 검증

- 양자 컴퓨팅에서 많이 활용되면 가장 중요한 알고리즘 중 하나는 QFT이다. DFT는 $O(N^2)$의 복잡도를 보이고, FFT가 $O(N\log N)$가 복잡도를 갖는 데에 반해서, QFT는 $O((\log N)^2)$의 복잡도를 갖는다. 이러한 QFT 알고리즘의 컴파일에 대한 프로그램을 올바르게 작성하고, 이를 QX simulator와 같은 시뮬레이션 방법을 통하여 QFT 알고리즘의 유효성을 검증하고자 한다.

(2) 3가지의 주요 활용 알고리즘의 프로그래밍 및 검증

- 양자컴퓨터를 이용하여 큰 효과를 볼 수 있을 것으로 생각되는 주요 분야는 크게 세 부분으로 볼 수 있으며, 다음과 같이 세 가지의 주요 활용 알고리즘이 있다. 우선, 계산과학 분야에서 중요한 부분인 바닥상태 에너지 추산(GSE) 알고리즘이 있으며, RSA 암호와 같은 암호 시스템을 해독하는 데 결정적인 역할을 하는 쇼어(Shor)의 알고리즘이 있다. 또한, 소프트웨어 검증과 관련된 부울 형식(Boolean Formula)도 주요 활용 알고리즘으로 볼 수 있다. 이러한 주요 활용 알고리즘들의 유효성을 검증하고, 컴파일하여 그 컴파일 결과가 타당한지 QX simulator와 같은 검증 도구를 이용하여 그 유효성을 검증하고자 한다.

◇ 성능 측면

- 각 알고리즘별 임계 입력값 수준에서의 평가
  - 현재 ScaffCC 컴파일 프로그램에서 제공되는 알고리즘들의 입력값들은 대부분 작은 입력값만을 대상으로 하고 있다. 이 경우처럼 입력값이 작은 알고리즘의 경우, 지금의 컴퓨터를 이용해서도 빠른 시간 안에 결과를 얻을 수 있다. 따라서 양자컴퓨터의 성능이 더 좋다고 말하기가 어렵다. 하지만, 입력값이 커지면 지금의 컴퓨터로는 그 결과를 얻는데 너무 오래 걸려서 그 결과를 볼 수 없게 된다. 특히, 현대 암호 시스템은 그러한 부분을 염두하고 암호시스템이 효율적으로 구축되어 있다. 하지만, 주요 활용 양자 알고리즘의 경우에는 입력값이 커지더라도 그 결과를 얻는 데에 있어서 증가하는 정도가 작아 어느 정도 시간이 걸린 후, 그 결과를 확인할 수 있도록 알고리즘이 설계되어왔다. 따라서 양자 알고리즘을 이용한 양자 컴퓨터의 성능이 현대 컴퓨터의 성능을 뛰어넘게 되는 임계 입력값에 대한 컴파일 결과를 평가하고자 한다.

- 시스템 합성 부분

◇ 기능 측면

(1) 모듈들 간의 배치, 매개변수 큐빗들의 전달 과정

- 모듈 구조를 갖는 양자 어셈블리 코드는 비 모듈 구조 방식의 양자 어셈블리 코드에 비해서 분석의 용이성을 가지고 있다. 양자 알고리즘을 구동하는 관점에서 같은 모듈이 반복적으로 호출되지만, 실제 분석 과정은 한 번만 수행하면 되기 때문이다. 반면 비 모듈 구조 방식의 양자 어셈블리 코드는 같은 모듈이 수차례 반복되면 해당 내용이 모두 양자 어셈블리 코드의 main 모듈 안에 기술되어 있어서 분석해야할 코드의 양이 많아진다.

- 반면 모듈 구조의 양자 어셈블리 코드를 다룰 때는 모듈들 사이의 배치, 그리고 그에 따라 파생되는 매개변수 큐빗들의 전달 문제를 고려하여야 한다. 즉, main 모듈에서 큐빗 a 와 b를 매개변수로 하여 서브 모듈 A를 호출한다고 하면, 해당 모듈 A의 큐빗들은 현재 모듈(main)을 기준으로 어디에 위치해야 하는지가 명확해야 한다. 또한 매개변수 큐빗들 a 와 b를 어떤 방식을 이용해서 모듈 A 의 매개변수 큐빗들의 위치로 이동시킬 것인지도 명확하게 정의되어야 한다.

- 현재까지의 구현에서는 내포된 모듈(Nested module) 구조를 이용해서 호출하는 모듈(부모모듈)과 피호출 모듈(자식모듈) 간의 배치를 결정하고 있다. 하지만, 그 실제적인 구현 즉 매개변수 큐빗의 전달 부분은 아직 미완성단계로 남아 있어서 다음연차에서 반드시 구현해야 한다.

(2) 양자 시스템 코드 구동 관련 기능

- 빌드업 과정을 통해서 생성된 양자 컴퓨팅 시스템 코드의 구동과 관련하여 필요한 기능들을 구현한다. 시스템 빌더의 중요한 기능 중 하나는 빌드업 과정을 통해서 생성한 시스템 코드를 이용해서 직접 양자 컴퓨터 시스템의 디바이스들을 컨트롤하는 부분이다. 이를 위해서 큐빗들을 생성 및 관리 하는 부분, 시스템 코드의 개별 명령어에 따라서 목적 큐빗에 해당 명령어를 실행하는 디바이스 제어 신호를 발생시키는 기능 등이 중요하다. 현재까지의 시스템 합성 툴의 구현은 주로 빌드업 및 분석 기능의 완성도를 높이는 데 주안을 두었기 때문에 구동 관련 기능은 매우 미흡하다. 따라서 양자 시스템 코드 구동을 위한 기능들을 구현하는 것이 다음연차의 매우 중요한 목표라 할 수 있다.

(3) surface code 적용

- 현재의 블록 타입 양자 오류 정정 부호는 양자 컴퓨팅 장치의 오류에 매우 민감하다. 즉, 블록 타입의 양자 오류 정정 부호기반으로 양자 컴퓨팅을 수행하기 위해서 양자 컴퓨팅 장치의 허용 오류율이 매우 낮다(10^{-4}~10^{-6}). 이에 반해 surface code 라는 양자 오류 정정 부호는 0.1 %(10^{-3}) 정도의 오류율을 갖는 양자 컴퓨팅 장

치에서도 잘 동작할 수 있어서 현재 양자 컴퓨터 구현 연구개발에서 가장 각광을 받고 있다. 본 시스템 합성 툴은 지금까지는 블록 타입의 양자 오류 정정 부호를 대상으로 연구 개발되어 왔으나, 앞으로는 surface code 기반의 양자 컴퓨팅 체계를 지원할 수 있도록 연구 개발을 진행할 것이다.

◇ 성능 측면

(1) 시스템 빌더 동작 효율성 향상

- 시스템 빌더의 구현에 적용된 알고리즘을 개선하고 최적화 등을 수행하여 현재의 구현보다 동작의 효율성(구동 시간 및 메모리 사용량)을 향상 시키도록 한다.

- 임의빌딩블럭 부분

◇ 기능 측면

(1) 적용 가능한 Device의 확장 : 현재 Quantum dot -> 향후 Quantum dot, Ion trap, Superconducting qubit, Optical qubit

(2) 적용 가능한 QEC의 확장 : 현재 Steane code -> 향후 Steane code, Bacon-Shor code

(3) 임의빌딩블럭의 실질적인 구동 : 시스템 합성 부분에서 내려오는 명령에 따라 해당 명령을 임의빌딩블럭 명령어로 분해하여 실질적으로 구동을 할 수 있도록 개발한다.

◇ 성능 측면

(1) 현재 Measurement의 성능을 1로 가정하고 성능 평가를 하였으나 이를 확률적으로 반영하여 현실적인 성능 분석이 가능하도록 한다.

(2) 현재 사용하고 있는 산출식 $p_e = 1/_{kq}C_2$이 아닌 해당 기능에 있어서 오류 발생 시 영향을 미치는 모든 부분을 추적하여 정확한 성능 분석이 가능하도록 한다.

3.2 비용효율성 향상에 대한 연구개발

- 전체적 비용 감속 목표
  - 현재까지 제안된 비용효율성 향상 방법외에 별도로 컴파일, 시스템합성과정, 빌딩블럭 합성과정에서 각각 10배 이상씩의 성능향상을 목표로 한다.

Platform Part	Parameters	Final Goal (Phase 1)	2016.10	2017.12
Compiler	Resource Reduction	**10 times over 2016.10 version**	Set the base	Better Compile
System Builder	Resource Reduction	**10 times over 2016.10 version**	Set the base	Better System Architecture Dual-Code FTQC Model
Arbitrary Building Block Builder	Resource Reduction	**10 times over 2016.10 version**	Set the base	Better ABBBs Hetero Level Hetero Codes
Basic Building Block Builder	Resource Reduction	**10 times over 2016.10 version**	Set the base	Better BBBBs Machine Learning

- 컴파일 수준에서의 비용 감소 목표 및 적용 대상 기술들

◇ 비용 감소 목표

(1) 양자 어셈블리어 수준에서의 크기, 정확도, 시간이 포함된 비용 산출 방법 확정

현재 Scaffold 컴파일을 거치면 양자 어셈블리어를 얻을 수 있다. 즉, 전체 큐빗 수와 전체 게이트 수를 얻을 수 있다. 전체 게이트 수에 영향을 미치는 주요 요인으로는 컴파일 과정에서 나타나는 유니터리 행렬의 작은 크기의 유니터리 행렬들(일반적으로 T, H, S 게이트들로 구성되어 있다.)로의 근사과정에서 나타나는 오차범위에 대한 파라미터인 d가 있다. 이러한, 오차로 인하여 전체 알고리즘이 어느 정도 정확도로 구현되며, 만족할 만한 정확도를 갖기 위해서는 어느 정도 비용이 필요한지를 좀 더 세밀한 방법으로 확립하고, 그 비용을 산출하고자 한다.

(2) 2016년 10월 대비 10배 향상 기술 확보

- GSE 양자 알고리즘의 경우, M=40, b=3, d=10이라는 구체적인 파라미터에 대하여 필요한 자원들을 산출하였다. 이러한 산출 결과를 근거로 하여 그 결과보다 그 작은 자원을 필요로 하는 결과를 얻을 수 있도록 컴파일 엔진의 핵심 부분들을 개선해 나가고자 한다.

◇ 연구개발기술들 및 예상 효과

(1) 양자 컴퓨팅의 연산인 유니터리 행렬의 효율적인 분해 방법 분석 및 설계

단일 큐빗 게이트의 분해 방법에 대해서는 이론적으로 많은 연구가 되어 있고 프로그램화되어 있다. 하지만, 멀티 큐빗 게이트의 경우, 이론적인 연구들은 되어 있으나, 아직 컴파일 엔진이 거의 설계되어 있지 않다. 따라서 멀티 큐빗 게이트에 대한 컴파일 엔진의 프로그래밍이 필요하며, 이를 개선해 나가는 연구가 필요하다.

(2) 컴파일 엔진의 핵심 라이브러리 구축

- 현재 ScaffCC 컴파일 프로그램에는 임의의 회전 각도에 대한 단일 큐빗 게이트의 컴파일이 설계되어 있지만, 2 큐빗으로만 확장되어도 컴파일 엔진이 내장되어 있지 않으며, 현재 Scaffold 소스 파일에서 보이는 컴파일 엔진은 작성자마다 조금씩 다른 부분이 있으며, 에러가 포함된 부분도 있다. 이에 대한 부분들을 개선하여 라이브러리 형태로 구축해 나갈 예정이다. 이는 나중에 여러 양자 알고리즘의 성능을 개선하는 데에 있어서 중요한 역할을 할 것으로 예상된다.

- 시스템 합성 부분

◇ 비용 감소 목표

- 모듈내 큐빗의 배치를 최적화 하고, 모듈간의 배치의 최적화 하여 양자 컴퓨팅에 있어서의 통신의 비용(SWAP 게이트의 수) 을 감소시킨다(5배). 모듈들의 배치 방법(현재는 Nested modules)을 개선하여 필요한 큐빗의 수를 감소시킨다(2배).

◇ 연구개발기술들 및 예상 효과

- 레이아웃 최적화를 위해 MIM 같은 최적화 방법론 적용하여 최소의 큐빗 이동을 발생시키는 큐빗 배치 방법을 찾는다.

- 임의빌딩블럭 부분

◇ 비용 감소 목표

- s(임의빌딩블럭의 크기), t(각 연산 별 시간), p(성능)이라 할 때 $s \times t \times p$에서 최소 10배를 감소시키는 것을 목표로 한다.

◇ 연구개발기술들 및 예상 효과

(1) 이종 배열 사용 : 현재 이미 최상위 레벨과 하위 레벨의 임의빌딩블럭이 서로 다른 배열을 사용하여 단일 배열을 사용할 때 보다 $(25/30)^{l-1}$배의 기본빌딩블럭 수 감소의 효과를 보이고 있다. 더 나아가 임의빌딩블럭을 쓰지 않는 기능들을 제외하는 방향으로 세분화하여 다양한 배열을 사용할 경우 임의빌딩블럭의 크기와 사용하는 기본빌딩블럭 수 차원에서 비용 감소가 가능할 것으로 예상된다.

성능	오류율
CNOT	1.7363e-15
H	1.1926e-43
Z	1.1926e-43
X	1.1926e-43
Y	1.1926e-43
S	1.4226e-38
T	2.9608e-38

표 6. 레벨 3 임의빌딩블럭의 게이트 별 성능

(2) 이종 레벨 사용 : 최상의 레벨의 임의빌딩블럭의 기능별 성능을 확인해보면 1-큐빗 게이트와 2-큐빗 게이트의 성능이 지나치게 간극이 큰 것을 확인할 수 있다. 상기된 표에서 확인할 수 있듯이 2-큐빗 게이트 성능에 맞춰서 레벨이 측정된 임의빌딩블럭을 사용하여 1-큐빗 게이트 연산만 사용할 경우 비용 측면에 있어서 큰 낭비가 발생한다. 양자 알고리즘에서 임의의 rotation 게이트가 존재할 경우 이를 다수의 1-큐빗 게이트로 분해, 근사하여 사용하기 때문에 해당 경우에 있어서는 최상의 레벨의 임의빌딩블럭을 적당한 1-큐빗 게이트 성능을 가지는 하위 레벨로 하향하여 사용할 경우 비용 측면에서 큰 이득을 볼 수 있을 것으로 예상된다. 레벨을 한 단계만 낮추어 사용해도 동일한 게이트 열을 수행하는데 있어 시간 측면의 경우 지수적인 측면에서 두단계 정도 감소하는 이득을 볼 수 있다.

기능＼레벨	1	2	3
T 게이트	9.164 sec	1883.460 sec	388068.201 sec
CNOT 게이트	4.957 sec	1021.569 sec	210476.603 sec

표 7. T, CNOT 게이트 레벨 별 동작 시간

제4장 향후 활용 방안

- 다양한 알고리즘, 다양한 소자, 다양한 오류보정코드의 성능 분석

- 본 연구에서 개발된 양자컴퓨팅 플랫폼을 이용하여 설계되는 양자컴퓨팅 시스템의 예상성능을 분석하는데 활용한다. 이러한 분석을 통해서, 양자컴퓨팅의 기대 성능을 향상시키기 위해서는 어떠한 부분에서 실질적인 개선이 되어야 하는지를 명확히 파악한다.

현재까지 구현된 플랫폼의 우수성 및 비용효율성 향상 연구에서의 활용 가치

- **(세계최고 수준의 정교한 평가 가능) Microsoft, IARPA 보다도 우수한 수준**
  - ※ 프로그래밍, 컴파일과정, 시스템합성과정, 빌딩블럭 합성과정, 소자기술정보 등을 모두 포함
  - ※ 각 구성요소별 제한점이 최대한 반영됨
  - ※ 슈퍼컴 대비 임계 입력값 수준에서의 양자컴퓨팅 예상 성능 도출 가능
- **(양자컴퓨팅 운영체제의 중요성) 양자컴퓨팅의 구동 시스템으로 활용, 비용효율성 향상 기법의 분석에 활용**
  - ※ [운영체제로의 즉각적 활용가능] 최종 결과인 양자컴퓨터 운영체제는 다수의 양자정보하드웨어에 적용가능
  - ※ [플랫폼을 통한 계산성능 정교분석 가능] 다양한 조합에 대한 동적인 측면까지 포함한 계산성능 분석 가능
  - ※ [비용효율성 향상기법 검증 가능] 비용효율성 향상 기법들에 대한 플랫폼 수준에서의 정교한 평가 가능

입력 항목	값 및 의미
Ground State Estimation	• 평가 대상 양자알고리즘 • 계산과학분야에서 많이 사용되는 ground state를 계산하는 알고리즘 • 슈퍼컴에서는 m=40 정도주순에서 분석 가능
Quantum Dot Qubit Technology	• 평가 대상 양자소자기술 • 반도체형 큐빗으로써 크기가 작고 집적하기 용이함
Steane Code	• 양자오류보정코드와 해당 결함허용방식기술 • 코드가 유효하기 위한 임계 오류값은 10^-4로 추산
2D Computer Architecture	• 물리적 큐빗과 논리적 큐빗들의 공간상의 배치구조
SWAP Communication Method	• 물리적 큐빗간 혹은 논리적 큐빗간 상호 연결을 위해서 필요한 통신 기술

입력 항목	값 및 의미
Ground State Estimation	• 어셈블리코드수준에서의 큐빗과 게이트가 허용하는 최대 오류율값(Max Tolerable Error Rate): 2.4067 * 10^-11

Quantum Dot Qubit Technology	• 크기 4 nm
	• 게이트별동작시간 및 에러율
	  • QEC, 0.009 s, 0.00000000337
	  • Prep, 0.0008 s, 0.00000000563
	  • SWAP, 0.0000000168 s, 0.00000000152
	  • CNOT, 0.00000105 s, 0.00000000551
	  • Meas., 0.00000315 s, 1

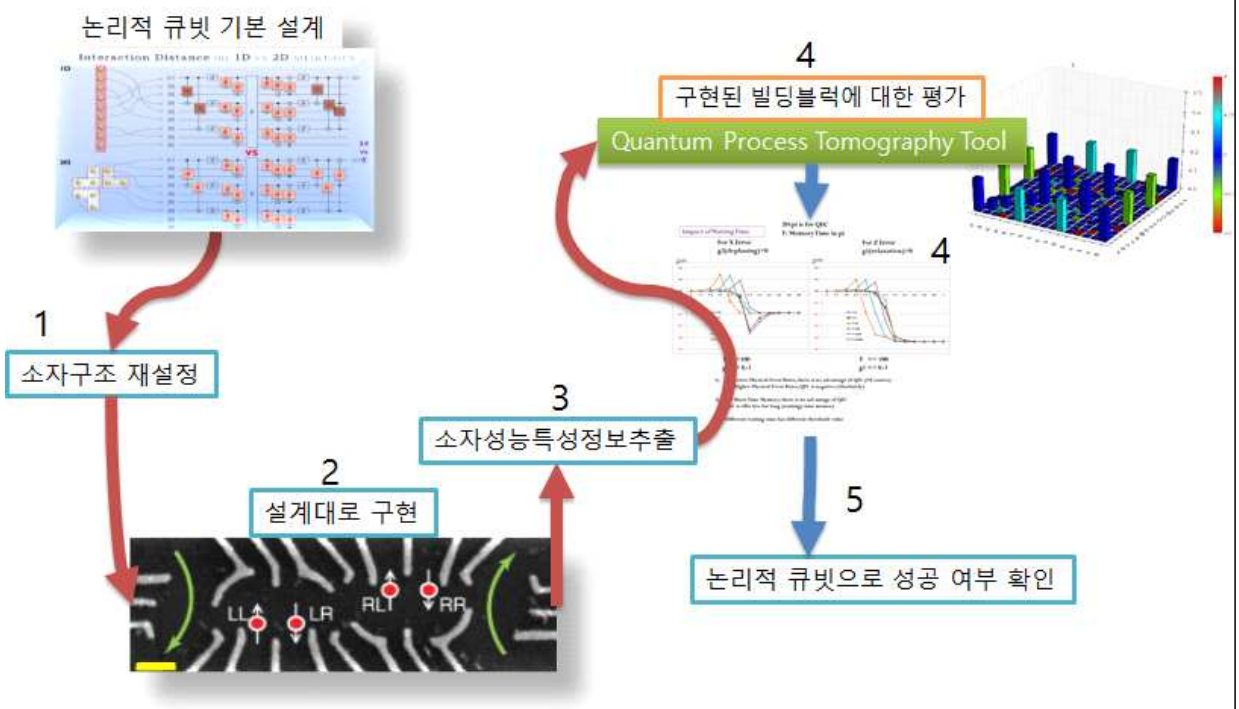
Steane Code	• BBB, ABB1, ABB2, ABB3 까지 사용되어야 함

출력 항목	값 및 의미
1회 계산 총 시간	• 407047258 초(대략 12년) [ABB0, BBB급
연산 정확도	• 0.99999999996717 (ABB3의 정확도) • KQ: 549063085925 (error의 경우의 수) • 0.000000015 (정확도)
평균 계산 시간	• ~ 12년 * 10^7 (반복횟수)
시스템 크기	• 43 * 27000 (micro meter square)

- 논리적 빌딩블럭 구현에 활용

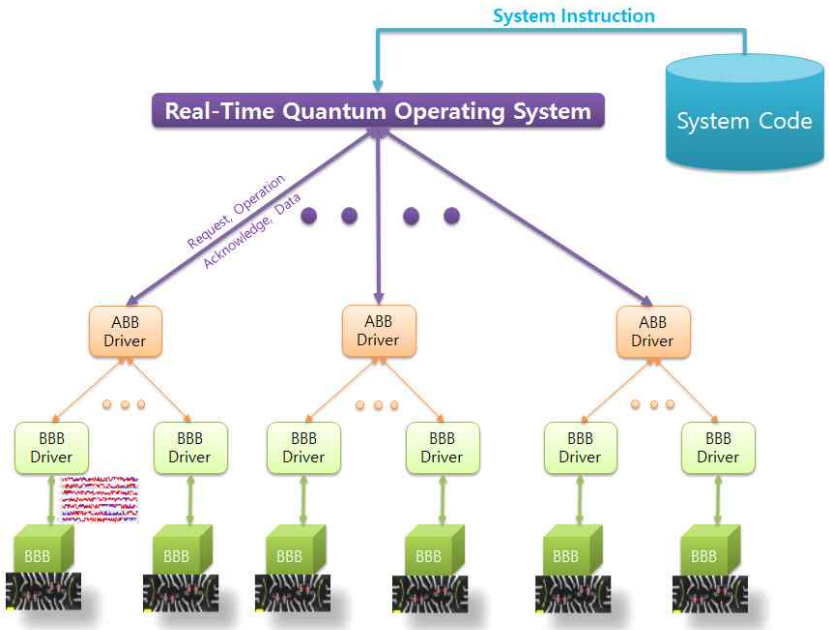
- 기본빌딩블럭 합성 기능을 통해서 주어진 양자정보소자를 대상으로 하여 실제로 오류보정을 통한 논리적 큐빗 기능의 구현을 검증하는데 활용한다.

- 아래 그림은 논리적 큐빗 블록의 설계와 이를 실제로 구현하여 검증하는 과정을 도식화하고 있다. 본 연구에서 개발된 BBB를 통하여 논리적 큐빗의 설계와 이에 따른 평가 기능을 제공하며, 이를 토대로 논리적 큐빗블럭 구현이 가능할 것으로 예상된다.



- 양자컴퓨팅 운영체제로 확장

- 본 연구에서의 플랫폼은 현재까지 성능분석의 목적으로 개발되었다. 다음 단계로는 이러한 플랫폼을 확장하여 양자정보소자에 대한 실시간 제어신호 생성등이 가능한 양자컴퓨팅 운영체제로의 확장을 고려하고 있다.
- 다음 그림은 현재 연구개발 과정에서 고려하는 다음 단계에서의 운영체제 구축에 대한 개념도이다.



제5장 결론 및 제언

본 연구개발에서는 양자정보기반 양자컴퓨팅 시스템의 구현과 구동을 위해서 필요한 플랫폼 구축과 양자컴퓨팅의 기대성능 향상을 위한 핵심기술은 비용효율성 향상기술과 관련한 연구개발을 목표로 하고 있다. 현재까지의 연구개발 결과는 사용자가 활용 가능한 수준에서의 양자컴퓨팅 플랫폼을 구축하였으며, 이를 토대로 양자컴퓨팅의 예상성능을 분석하는 것이 가능하다. 또한, 이 과정에서 결함허용과정에서 비용효율성을 향상시키는 방법과 각 부분별 비용특성을 분석하였다.

이러한 연구는 실제 양자정보소자 기술 구현과 더불어, 최종적인 목표인 양자컴퓨팅 시스템을 구축하는데 필요한 시스템구조와 이를 구동하기 위한 운영체제를 주축으로 하고 있다. 이러한 운영체제는 양자정보HW와 함께 양자컴퓨팅 시스템의 핵심기술로 활용될 예정입니다. 한편, 양자정보HW는 양자컴퓨팅 시스템의 측면에서 만족되어야 하는 기능과 성능등을 이러한 플랫폼을 통해서 명확히 파악할 수 있다. 이에 따라서, 본 연구개발 목표인 양자컴퓨팅 플랫폼 구축은 실질적인 양자컴퓨팅 구현에 매우 중요하다. 또한, 비용효율성 향상 기술은 향후 핵심기술로써 그 기술적 가치가 매우 높다.

다음 단계로는 이러한 양자컴퓨팅 플랫폼 자체의 성능을 지속적으로 향상시키는 것과 더불러 양자정보HW에서의 논리적 큐빗의 구현이 요구된다. 이를 위해서는 우선적으로 오류보정과정이 효과적으로 동작하는 것을 확인할 필요가 있는데, 이를 위해서는 양자정보 HW관련 연구가 수반되어야 할 것으로 예상된다. 따라서, 본 연구를 지속적으로 확장하여 실질적인 양자컴퓨팅 시스템을 구현할 수 있도록 HW부분에서의 연구개발도 요구된다.

연구개발 실적

[실적 1, 첨부 3]게재저널: Physical Review A (SCI, IF 2.765, 표준상위 25%)
제목: Analysis of physical requirements for simple three-qubit and nine-qubit quantum error correction on quantum-dot and superconductor qubits
저자: IlKwon Sohn, Seigo Tarucha, and Byung-Soo Choi
현재상태: 온라인출판 (12/12)

[실적 2, 첨부 2]제목: Preparation of a heterogeneously encoded logical Bell state
출원인: 황용수, 최병수
출원국가: 대한민국(국내) 미국, 일본, 중국 (3극 특허)

[실적 3, 첨부 1] 현재까지 개발된 양자컴퓨팅 플랫폼

[기타실적, 첨부 4] 게재저널: Physical Review A (SCI, IF 2.765, 표준상위 25%)
제목: Interoperability in Encoded Quantum Repeater Networks
저자: Shota Nagayama, Byung-Soo Choi, Simon Devitt, Shigeya Suzuki, Rodney Van Meter
현재상태: Phys. Rev. A 93, 042338 (2016)

[기타실적, 첨부 5] 책: Principles and Methods of Quantum Information Technologies (Lecture Notes in Physics)
저자: Yoshihisa Yamamoto, Kouichi Semba
ISSN: 978-4-431-55755-5
챕터제목: Spin Qubits with Semiconductor Quantum Dots
챕터저자: Seigo Tarucha, Michihisa Yamamoto, Akira Oiwa, Byung-Soo Choi, Yasuhiro Tokura
[기타실적, 첨부 6] 게제지: 전자통신동향분석
제목: 양자컴퓨팅시스템 개발 및 활용 동향
저자: 최병수
현재상태: 전자통신동향분석 31권 2호 (통권 158), 84-94 (2016)

소요비용

비 목 별	1세부					1세부 합계		
	ETRI	고려대		한양대		ETRI	공동	합계
	현금	현금	현물	현금	현물			
1. 인건비	52,900	-	-	-	-	52,900	-	52,900
o 내부인건비(정)	52,900	-	-	-	-	52,900	-	52,900
2. 직 접 비	392,600	-	5,250	-	5,000	422,600	10,250	432,850
가. 외부인건비	221,996	-	-	-	-	221,996	-	227,246
1) 내부인건비(계)	157,000	-	-	30,000	-	157,000	30,000	187,000
2) 외부인건비	34,996	-	5,250	-	-	34,996	5,250	40,246
나. 연구장비·재료비	28,939	-	-	-	-	28,939	-	33,939
1) 연구장비 및 시설비	24,400	-	-	-	-	24,400	-	24,400
2) 재료비 및 전산처리비	4,539	-	-	-	5,000	4,539	5,000	9,539
3) 시작품 제작비	0	-	-	-	-	0	-	0
다. 연구활동비	76,861	-	-	-	-	76,861	-	76,861
1) 국외여비	27,362	-	-	-	-	27,362	-	27,362
2) 수용비 및 수수료	9,834	-	-	-	-	9,834	-	9,834
3) 기술정보 활동비	39,665	-	-	-	-	39,665	-	39,665
라. 연구과제추진비	20,808	-	-	-	-	20,808	-	20,808
1) 국내여비	6,408	-	-	-	-	6,408	-	6,408
2) 회의-야근식대 다과비	8,400	-	-	-	-	8,400	-	8,400
3) 사무용품비	6,000	-	-	-	-	6,000	-	6,000
마. 연구수당	54,979	-	-	-	-	54,979	-	54,979
바. 위 탁 연 구 개 발 비	0	-	-	-	-	0	-	0
사. 연구지원비	2,749	-	-	-	-	2,749	-	2,749
아. 성과활용지원비	15,000	-	-	-	-	15,000	-	15,000
자. 평가·관리비	1,268	-	-	-	-	1,268	-	1,268
3. 간 접 비	60,219	-	-	-	-	60,219	-	60,219
4. 장비구입비(직접비)	0	-	-	-	-	0	-	0
합 계 장비비 포함	505,719	-	5,250	-	35,000	505,719	40,250	545,969
합 계 장비비 미포함	505,719	-	5,250	-	35,000	505,719	40,250	545,969

비 목 별	1차년도 (2015 년)		2차년도 (2016 년)		3차년도 (2017 년)		합계	
	금 액	구성비율	금 액	구성비율	금 액	구성비율	금 액	구성비율
1. 인 건 비	104,370	20.9	52,900	9.7	52,900	9.6	210,170	13.2
◦내부인건비(정규직)	104,370	20.9	52,900	9.7	52,900	9.6	210,170	13.2
2. 직 접 비	395,630	79.1	432,850	79.3	436,100	79.4	1,264,580	79.3
2.1. 외부인건비	39,000	7.8	227,246	41.6	223,496	40.7	489,742	30.7
1) 내부인건비(계약직)	33,000	6.6	187,000	34.3	187,000	34.0	407,000	25.5
2) 외부인건비	6,000	1.2	40,246	7.4	36,496	6.6	82,742	5.2
2.2. 연구장비·재료비	194,170	38.8	33,939	6.2	40,939	7.5	269,048	16.9
1) 연구장비 및 시설비	76,000	15.2	24,400	4.5	24,400	4.4	124,800	7.8
2) 재료비 및 전산처리비	118,170	23.6	9,539	1.7	16,539	3.0	144,248	9.0
3) 시 작 품 제 작 비	0	0	0	0.0	0	0.0	0	0.0
2.3. 연구활동비	107,189	21.4	76,861	14.1	76,861	14.0	260,911	16.4
1) 국 외 여 비	35,041	7.0	27,362	5.0	27,362	5.0	89,765	5.6
2) 수용비 및 수수료	8,909	1.8	9,834	1.8	9,834	1.8	28,577	1.8
3) 기술정보 활동비	63,240	12.6	39,665	7.3	39,665	7.2	142,570	8.9
2.4. 연구과제추진비	21,460	4.3	20,808	3.8	20,808	3.8	63,076	4.0
1) 국내여비	3,560	0.7	6,408	1.2	6,408	1.2	16,376	1.0
2) 회의-야근식대 다과비	15,500	3.1	8,400	1.5	8,400	1.5	32,300	2.0
3) 사무용품비	2,400	0.5	6,000	1.1	6,000	1.1	14,400	0.9
2.5. 연구수당	28,674	5.7	54,979	10.1	54,979	10.0	138,632	8.7
2.6. 위 탁 연 구 개 발 비	0	0	0	0.0	0	0.0	0	0.0
2.7. 연구지원비	2,300	0.5	2,749	0.5	2,749	0.5	7,798	0.5
2.8. 성과활용지원비	1,520	0.3	15,000	2.7	15,000	2.7	31,520	2.0
2.9. 평가·관리비	1,316	0.3	1,268	0.2	1,268	0.2	3,852	0.2
3. 간 접 비	-	-	60,219	11.0	60,219	11.0	120,438	7.6
4. 장비구입비(직접비)	0	0	0	0.0	0	0.0	0	0.0
총 액 장비비 미포함	500,000	100	545,969	100	549,219	100	1,595,188	100
총 액 장비비 포함	500,000	100	545,969	100	549,219	100	1,595,188	100

참고문헌

[MS] https://www.microsoft.com/en-us/research/project/language-integrated-quantum-operations-liqui/

[IBM] http://www.research.ibm.com/quantum/

[Metodi11] T. Metodi et al., Quantum computing for computer architects, Morgan & Claypool, 2011.

[Metodi08] T. Metodi et al., High-level Interconnect model for the quantum logic array architecture, ACM journal of emerging technologies for computing, 2008

[Metodi05b] T. Metodi et al., A quantum logic array microarchitecture: scalable quantum data movement and computation, International symposium on microarchitecture (MICRO-38), 2005

[Metodi05a] T. Metodi et al., A general purpose architectural layout for arbitrary quantum computations, SPIE defense & security symposium, 2005.

[Jonse12] N Cody Jonse et al., Layered Architecture for quantum computing, PRX, vol. 12 (3), pp. 031007 (2012)

[Weinstein13] Yaakov S. Weinstein, Non-fault-tolerant T gates for the [7,1,3] quantum error-correction code, PHYSICAL REVIEW A 87, 032320 (2013)

[Weinstein15a] Yaakov S. Weinstein, How Often Must We Apply Syndrome Measurements?, Quantum Information Science Group, MITRE (2015)

[Goto16] Hayato Goto, Minimizing resource overheads for fault-tolerant preparation of encoded states of the Steane code, Scientific Reports 5:19578, (2016)

[Weinstein12] Yaakov S. Weinstein, Use of Shor states for the [7,1,3] quantum error-correcting code, PHYSICAL REVIEW A 86, 052336 (2012)

[Weinstein15b] Yaakov S. Weinstein, Syndrome Measurement Strategies for the [[7,1,3]] Code, Quantum Information Processing (2015)

약어표	
CNOT	Controlled-NOT
DFT	Discrete Fourier Transform
FCFS	First Come First Served
FFT	Fast Fourier Transform
GSE	Ground State Estimation
MIM	Mixed Integer Programming
QD	Quantum Dot
QEC	Quantum Error Correction
QFT	Quantum Fourier Transform
QuArC	Quantum Architectures and Computation group
SC	Superconductor
ScaffCC	Scaffold Compiler Collection
SQCT	Single Qubit Circuit Toolkit