Araştırma Makalesi
Adıyaman Üniversitesi Mühendislik Bilimleri Dergisi
21 (2023) 227-241

KUANTUM PROGRAMLAMA AÇISINDAN KUANTUM DERLEYİCİLERİN KARŞILAŞTIRMALI ANALİZİ VE IBMQ UYGULAMASI

Mehmet KARAKÖSE[1], Hasan YETİŞ[2*], Osman Furkan KÜÇÜK[3], Çağatay Umut ÖĞDÜ[4], Orhan YAMAN[5]

[1,2,3,4] Fırat Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, Elazığ, 23119, Türkiye

[5] Fırat Üniversitesi, Mühendislik Fakültesi, Adli Bilişim Mühendisliği Bölümü, Elazığ, 23119, Türkiye

Geliş Tarihi/Received Date: 31.07.2023 Kabul Tarihi/Accepted Date: 17.11.2023 DOI: 10.54365/adyumbd.1334196

ÖZET

Kuantum hesaplama, geleneksel bilgisayarların yapamayacağı kadar karmaşık hesaplamaları çok daha hızlı ve daha verimli gerçekleştirmeye olanak tanıyan bir teknolojidir. Ancak kuantum bilgisayarların çalıştırılması için özel olarak tasarlanmış kuantum algoritmalara ihtiyaç duyulmaktadır. Bu algoritmaların kuantum bilgisayarlarda verimli bir şekilde çalıştırabilmek için uygun derleyici ve kuantum bilgisayar seçimi kritik öneme sahiptir. Bu çalışmada kauntum programlama ve derleyicileri hakkında bilgiler verilerek, literatürdeki kuantum derleyicilerin karşılaştırmaları gerçekleştirilmiştir. Örnek bir soyut kuantum devre 5 kübtlik ibmq_belem, ibmq_quito ve ibmq_manila kuantum bilgisayarlarında çalıştırılarak, kuantum devrelerin çalışma mantığı uygulamalı olarak açıklanmıştır. Yapılan analizlerler sonucu L tipi kübit bağlantısına sahip ibmq_manila bilgisayarının ortalama %86 ile daha başarılı sonuçlar ürettiği gözlemlenmiştir. Diğer taraftan T tipi kübit bağlantılarına sahip ibmq_quito ve ibmq_belem bilgisayarlarının ürettikleri sonuçların başarısı ortaalama %82 ve %48 ile sınırlı kalmaktadır. Aynı kübit bağlantısına sahip bu bilgisayarların başarımları arasındaki gözle görülür farkın sebebi kübit ve bağlantılardaki hata oranlarının olduğu sonucuna varılmıştır.

Anahtar Kelimeler: *Uygulamalı kuantum hesaplama, Kuantum bilgisayarlar, Kuantum derleyiciler, IBMQ.*

COMPARATIVE ANALYSIS OF QUANTUM COMPILERS IN TERMS OF QUANTUM PROGRAMMING AND IBMQ IMPLEMENTATION

ABSTRACT

Quantum computing is a technology that allows performing complex calculations much faster and more efficiently than conventional computers. However, specifically designed quantum algorithms are needed to run quantum computers. Appropriate compiler and quantum computer selection is critical in order to run these algorithms efficiently on quantum computers. In this study, information about quantum programming and compilers is given and the quantum compilers in the literature are compared. An example quantum circuit is run on 5 qubit ibmq_belem, ibmq_quito and ibmq_manila quantum computers, and the working logic of quantum circuits is explained practically. As a result of the analysis, it is observed that the ibmq_manila computer with L-type qubit connection produced more successful results with an average of 86%. On the other hand, the success of the results produced by ibmq_quito and ibmq_belem computers with T-type qubit connections is limited to 82% and 48% on average. It has been concluded that the reason for the noticeable difference between the performances of these computers with the same qubit connection is the error rates in the qubits and connections.

Keywords: *Applied quantum computing, Quantum computers, Quantum compilers, IBMQ.*

e-posta[1] : mkarakose@firat.edu.tr ORCID ID: https://orcid.org/0000-0002-3276-3788
e-posta[2] : h.yetis@firat.edu.tr ORCID ID: https://orcid.org/0000-0001-7608-3293 (Sorumlu Yazar)
e-posta[3] : osmanfurkankucuk2003@gmail.com ORCID ID: https://orcid.org/0009-0002-5594-3009
e-posta[4] : cagatayogdu@gmail.com ORCID ID: https://orcid.org/0009-0004-1697-4392
e-posta[5] : orhanyaman@firat.edu.tr ORCID ID: https://orcid.org/0000-0001-9623-2284

1. Giriş

Kuantum mekaniğini kullanarak bir fiziksel hesaplamanın nasıl yapılacağı üzerinde fikirler kuran bilim insanları, çeşitli girişimlerde bulunarak kuantum bilgisayar düşüncesini geliştirmişlerdir.1980 ve 1990'lı yıllarda Richard Feynman ve Paul Benioff yapmış oldukları çalışmalar ile kuantum bilgisayarların temelini literatüre kazandırmışlardır ve kuantum bilgisayarlar günümüzde de gelişimine devam etmektedir [1–3]. Henüz pratikte kullanımı kısıtlı olan bu teknoloji, yakın gelecekte siber güvenlik, malzeme, ilaç, bankacılık, finans sektörlerinde ve ileri imalat teknolojilerinde ciddi katkı sağlayacağı öngörülmektedir [4].

Klasik bilgisayarlardan farklı olarak klasik bitler yerine kuantum bitleri (kübit) kullanır. Bu kübitler herhangi bir foton, elektron, iyon veya atom çekirdeklerinin uygun koşullarda muhafaza edilmesi ile oluşturulabilir. Kuantum bitlerinin kendine has özellikleri sayesinde problem çözümlerinde çok yüksek bir hız kazanılmıştır. Öyle ki 53 kübite sahip, Google Yapay Zekâ Topluluğu tarafından geliştirilen Sycamore kuantum işlemcisinin, klasik süper bilgisayarın 10.000 yılda yapabileceği hesaplanan işlemi yaklaşık 200 saniyede tamamlayacağı gösterilmiştir [5]. Yapılan çalışmalarda, kombinatoryal optimizasyon problemlerinde, doğrusal cebir temelli problemlerde, diferansiyel denklem problemlerinde, faktörizasyon sürecinde klasik bilgisayarlara nazaran üstün performans sergilediği de bilinmektedir [6]. Üstelik bir süper bilgisayar olan Tianhe-2 çalışması esnasında 17.6 MW güç tüketirken Sycamore 26 kW güç tüketmekte ve hesaplamayı saniyeler içinde tamamlamaktadır.

Kuantumun üstünlüğü kendini göstermiş olsa da ne yazık ki günümüz kuantum bilgisayarları gürültülü sonuçlar ürettiğinden bu üstünlüğü tam kapasite ile kullanamamaktayız. Gürültü kuantum bilgisayarların özelliğinden kaynaklanan bir durumdur. Bir kuantum hesaplamanın yapılabilmesi için kübitlerin birbirleriyle kuvvetli bir bağ içinde olması ve dışarıdan son derece izole halde tutulmaları gerekir. Kuantum bitler yalnızca diğer kübitlerle dolanık hale gelmez, manyetik ve elektriksel gürültüler nedeniyle sistem, yani çevre ile de dolanık hale gelebilmekte ve etkilenebilmektedir. Bu etkiye kuantum uyumsuzluğu (*quantum decoherence*) denir [7]. Günümüzde kuantum bilgisayarlar mutlak 0 sıcaklığına yakın değerde ve herhangi manyetik etkiden uzak tutulmaktadır. Sadece ölçüm yapılacağı zaman ise sisteme dışarıdan etki edilir. Bulunduğumuz noktada çağımız Gürültülü Orta Ölçekli Kuantum (NISQ) çağı olarak adlandırılmaktadır [8]. Yapılan ölçümün hata oranının düşük sayılabilmesi için çeşitli donanım sınırlamaları getirilir. Örneğin bir kuantum işlemcide kübitler sınırlı bağlantılarla 2 boyutlu bir harita üzerinde düzenlenir ve yalnızca bitişik komşusu olan kübitler ile etkileşime girmesine izin verilir. Bu yüzden NISQ çağında kuantum devrenin en iyi şekilde optimizasyonunu sağlayacak uygun kuantum derleyicinin seçimi önemlidir [9].

Bu makalede kuantum programlama temellerine, kuantum derleyicinin görevlerine, çalışma mekanizmalarına, mevcut olan kuantum derleyicilerin karşılaştırmalarına yer verilerek IBMQ composer ortamında yarım toplayıcı devre uygulaması gerçekleştirilmektedir. Farklı kübit bağlantılarına sahip IBMQ bilgisayarların üretmiş oldukları sonuçlar analiz edilmiştir.

1.1. Motivasyon

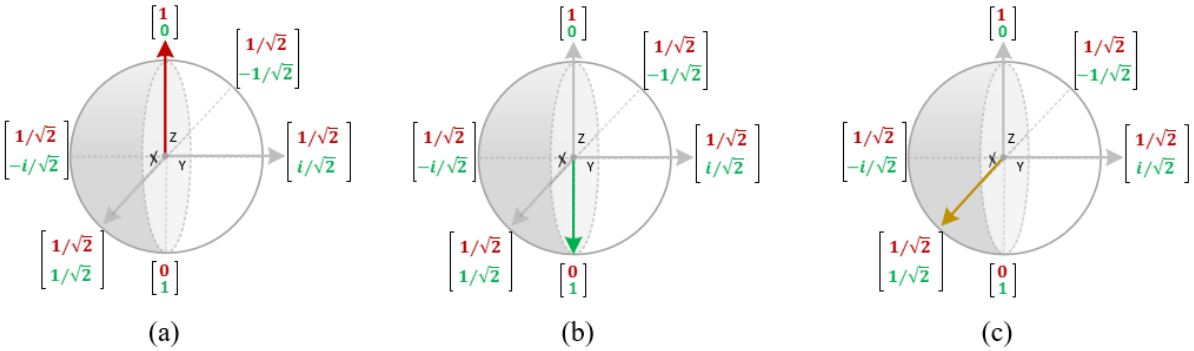
Kuantum hesaplama ve kuantum bilgisayarlar yüksek işlem gücü ve düşük güç tüketimleri ile büyük potansiyel arz etmektedir. NISQ çağında değerlendirilen günümüz kuantum bilgisayarlarında, kuantum devrelerin çalıştırılması sırasında gürültü meydana gelmektedir. Gürültülerin en aza indirgenmesi için kuantum derleyiciler ve uygun kuantum bilgisayar seçimi önemli bir konudur. Bu kapsamda gerçekleştirilen çalışma için temel motivasyon kaynaklarımız aşağıda verilmiştir:

- Literatürde kuantum derleyiciler ve uygun kuantum bilgisayarların seçilmesi ile ilgili yapılan çalışmaların yetersizliği ve özellikle Türkçe kaynakların eksikliği. Uygulamalı kuantum hesaplama alanında çalışmak isteyen araştırmacılar için kaynak teşkil edilmesi.

- Kuantum donanımların sahip oldukları kübit/bağlantı hata değerlerinin ve derleyiciler aracılığıyla elde edilen dönüştürülmüş devrelerin kuantum algoritmaların çalıştırılması üzerindeki etkilerinin incelenmesi.
- NISQ dönemindeki kuantum bilgisayarlarda gürültülerin en aza indirgenmesi gerekliliği ve gürültüleri azaltmak için uygun derleyici/donanım seçiminin öneminin vurgulanması

2. Kuantum Bilgisayarlarda Programlama

Kuantum programlama belirli bir sonuç elde etmek amacıyla kuantum donanımında çalışabilir bir dizi işlemin oluşturulması için kodun tasarlanması ve oluşturulmasıdır [10]. Kuantum programlamada kübit kavramları kullanılmaktadır. Kübitler 0-1 arasında sonsuz değerde bulunabilirler [11] Kübitlerin aynı anda 1 ve 0 değerlerini alabilme ilkesine süperpozisyon ilkesi denir. Kübit durumlarının daha iyi anlaşılabilmesi adına görsel olarak Bloch küreleri ile temsil edilir [12]. Bloch küresi örneği Şekil 1'de verilmiştir. Şekil 1.a, 1.b ve 1.c sırası ile $|0\rangle, |1\rangle$ ve $|+\rangle$ durumlarını göstermektedir.



Şekil 1. Bloch küresi üzerinde kübit durumları. a) $|0\rangle$, b) $|1\rangle$, c) $|+\rangle$ [13].

Bir kuantum bilgisayarında programın/algoritmanın çalıştırılması istenildiğinde genel olarak dikkat edilmesi gereken hususlar şunlardır [14]:

I. Kullanıcı programı hangi kapı setiyle ifade edebilir.
II. Gerçekte hangi fiziksel kapılar kullanılmaktadır.
III. Kübit bağlantılar nelerdir.
IV. Gürültü kaynakları nelerdir.

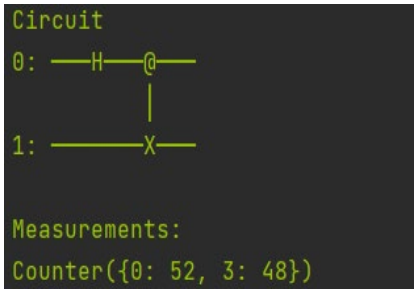
Kullanıcının kullandığı kapı seti ile donanımın sunduğu kapı seti farklılık gösterebilir. Uyumsuzluğu gidermek için kapı işlemleri daha farklı alt işlemlere ayrılması gerekebilir, bu da algoritmanın uzamasına sebep olabilir. Aynı şekilde kuantum donanımı yazılan algoritmanın gerektirdiği fiziksel kübit bağlantılarını sağlamayabilir. Son maddede değinilen gürültü kaynağı, kullanılan kapıların tam olarak kuantum bilgisayarın kullandığı kapı setine eşlenememesinden kaynaklanır. Doğru eşlenememe durumu kübitlerin dolanıklığını bozar ve algoritmanın performansını önemli ölçüde azaltır [15].

Mevcut durumda kuantum programlama için Cirq, Forest, Ocean, ProjectQ, Qiskit, IBM-Q, PyZX, Pytket gibi kütüphaneler; OpenQL [16] gibi yük seviyeli framework'ler, IBM Quantum gibi bulut tabanlı donanım erişimi sağlayıcıları bulunmaktadır [17]. Ayrıca diğer kütüphanelerin aksine Quantum Development Kit (QDK), kendi dili olan Q# ile kullanıcılara kuantum programlama imkân sunar. Çizelge 1'de temel bir Bell Durumu'nu oluşturmak için Python dilinde Cirq Kütüphanesi kullanılarak yazılmış kod verilmiştir [18].

Çizelge 1. Cirq kütüphanesi kullanılarak Python ortamında yazılmış Bell durumu kodları.

1	import cirq
2	from qiskit.visualization import plot_histogram
3	# Devreyi ve kübitleri çağırma
4	qreg = [cirq.LineQubit(x) for x in range(2)]
5	circ = cirq.Circuit()
6	# Devreyi bell durumuna hazırlama
7	circ.append([cirq.H(qreg[0])])
8	cirq.CNOT(qreg[0], qreg[1])])
9	print(circ) #devreyi çizme
10	circ.append(cirq.measure(*qreg, key=”z”)) # Ölçüm ekleme
11	# Devreyi simüle etme
12	sim = cirq.Simulator()
13	res = sim.run(circ, repetitions=100)
14	# Çıktıları yazdırma
15	print(“Olcumler:”)
16	print(res.histogram(key=”z”))

Bell durumu 2 kübitin birbiri ile maksimum korelasyon ve bağlantı içerdiği halidir. Yani kübitlerin dolanık hale gelmesidir [19]. İkiden fazla kübitin dolanık hale gelmesi için farklı durumlar bulunsa de Bell durumu yalnızca 2 dolanık kübitin ölçümlerinin eş olduğunu gösterir [18]. Şekil 2'de verilen programın çıktılarına bakıldığında, 100 defa çalıştırılan simülasyonda sonucun $|00\rangle$ (0) ya da $|11\rangle$ (3) olarak üretildiği görülmektedir. Bu durum ilk kübitin ölçümünün yapılması ile ikinci kübitin de de aynı değeri alması anlamına gelmektedir.



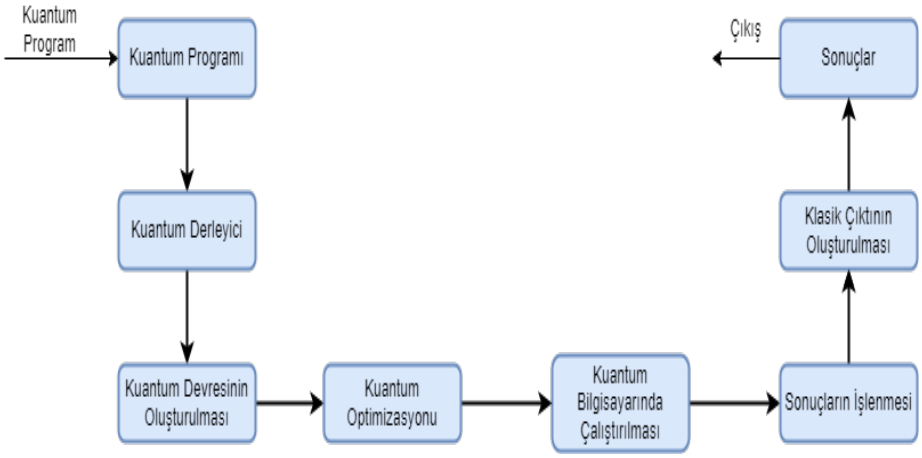
Şekil 2. Python Cirq kütüphanesi Bell durumu çıktısı

Kuantum devreleri genellikle kullanıcılar tarafından devrenin donanım bağımsız (soyut) tasarlanır. Bu soyut devre uygun bir derleyici aracılığıyla belirli bir kuantum donanımında çalıştırılacak bir devreye dönüştürülür. Fakat devre dönüşümü ihmal edilemez bir maliyeti de bulunmaktadır. Bu nedenle, derleyici bir optimizasyon sorunuyla karşı karşıyadır. Yükü en aza indirirken uygulanabilir bir dönüşüm bulmak tek bir işlemci için bile NP-zor olduğu bilinmektedir [20,21]. Bu nedenle, kuantum derleyicisinin optimizasyon yapması gerekmektedir [22].

3. Kuantum Derleyiciler

Kuantum programlama, kuantum bilgisayarlarının gücünden faydalanarak yeni ve gelişmiş algoritmaların geliştirilmesini sağlayan bir alandır. Kuantum bilgisayarları, geleneksel bilgisayarlara kıyasla benzersiz hesaplama yetenekleri sunar ancak bu yetenekleri tam anlamıyla kullanabilmek için özel olarak tasarlanmış derleyici yazılımlarına ihtiyaç duyarlar. Kuantum derleyiciler, kuantum programlarını kuantum bilgisayarlarında yürütülebilir hale getirmek için kaynak kodunu optimize eden, çeviren ve düzenleyen araçlardır. Ayrıca, kuantum cihazlarındaki sınırlamalara, hatalara uyum sağlamak için optimizasyon teknikleri ve hata düzeltme yöntemleri uygular. Kuantum derleyiciler, kuantum

programcılarına daha etkili, hata toleranslı ve performans açısından optimize edilmiş kuantum programlar oluşturma imkânı sunar, böylece kuantum bilgisayarların potansiyelini tam anlamıyla kullanabilmelerini sağlar. Şekil 3'te temel olarak kuantum programının çalışması için gerçekleştirilen aşamalar verilmiştir. Devamında her bir aşama maddeler halinde açıklanmıştır.



Şekil 3. Kuantum programların çalıştırılarak sonuçların elde edilmesi süreci

1) Kuantum Programı: Kuantum derleyicinin başlangıç noktasıdır. Kuantum programı, kuantum bilgisayarda çalıştırılacak olan algoritmanın veya işlemin bir tanımını içerir. Genellikle yüksek seviye kuantum programlama dilleriyle yazılır.
2) Kuantum Derleyici: Kuantum programını geleneksel bilgisayarların anlayabileceği dile dönüştüren bir yazılım aracıdır. Kuantum derleyici, kuantum programını analiz eder, doğruluk kontrolleri yapar ve optimize eder. Derleyici, kuantum programını kuantum devresi olarak temsil eden düşük seviyeli bir temsile dönüştürür.
3) Kuantum Devresinin Oluşturulması: Kuantum derleyicinin çıktısı olarak elde edilen kuantum devresi, kuantum programını fiziksel olarak gerçekleştirmek için kullanılan bir dizi kuantum kapısının ve kuantum hallerinin düzenlenmesini içerir. Kuantum devresi, kuantum kapıları, ölçümler ve kuantum durumları arasındaki ilişkileri tanımlar.
4) Kuantum Optimizasyonu: Kuantum devresi oluşturulduktan sonra, kuantum optimizasyon aşaması gerçekleştirilir. Bu aşamada, kuantum devresi üzerinde çeşitli optimizasyon teknikleri uygulanır. Optimizasyon, devrenin performansını iyileştirmek, hata (gürültü) etkisini azaltmak veya kullanılan kaynakları azaltmak gibi amaçlarla yapılabilir.
5) Kuantum Bilgisayarında Çalıştırılması: Kuantum devresi, kuantum bilgisayarda gerçekleştirilmek üzere fiziksel kuantum cihazlarına veya kuantum simülasyonlarına aktarılır. Gerçekleme aşamasında, kuantum kapıları ve ölçümler, kuantum cihazının özelliklerine ve kısıtlamalarına uygun şekilde uygulanır.
6) Sonuçların İşlenmesi: Kuantum bilgisayarında çalıştırılma işlemi tamamlandıktan sonra elde edilen sonuçlar işlenir. Bu aşamada ölçümlerden elde edilen kuantum durumları veya istatistiksel sonuçlar analiz edilir ve yorumlanır.
7) Klasik Çıktının Oluşturulması: Kuantum sonuçları, klasik bir çıktı formatına dönüştürülür. Bu çıktı, kuantum bilgisayardan gelen sonuçların klasik bilgisayarlar veya diğer sistemler tarafından işlenebilmesini sağlar.
8) Sonuçlar: Kuantum derleme sürecinin sonunda elde edilen sonuçları ifade eder. Bu aşamada, çıktılar kullanıcının elde etmek istediği bilgilere veya çözümlere dönüştürülür.

Kuantum hesaplama ve programlama için literatürde birçok derleyici tasarımı bulunmaktadır. Her derleyici farklı avantajlara sahiptir. Bu nedenle, en uygun derleyiciyi seçmek için önemli bir konudur. Kuantum algoritmaları genellikle Python gibi yüksek seviyeli dillerin sunmuş olduğu kütüphaneler aracılığıyla yazılarak QASM diline çevrilir. Bu sebeple, kuantum algoritmalarının devreye dönüştürülmesi için kullanılan programlama diline dikkat edilmesi ve derleyiciye uygun hale getirilmesi için teknik bilgiye ihtiayaç duyulduğu bildirilmektedir [23]. Bu kapsamda kuantum derleyicinin, gerçek kuantum bilgisayarının kapı seti ile en uygun nasıl eşleyebileceği önemli bir konudur [24]. Uygun kapı seti ile eşlenmemesi durumunda kuantum programların yürütme süresi uzamakta ve hata toleransı artmaktadır [25]. Ayrıca kübitlerin uygun fiziksel kübitlerle eşlenmesi gerektiği ve bu eşleme işleminin, kübitlerin çözünürlük süresi veya bağlantı özellikleri gibi farklı karakteristikleri nedeniyle kuantum algoritmasının yürütülmesini doğrudan etkilediği belirtmiştir [26].

Literatürde kuantum derleyicilerin kübit eşleştirmesi ve kullanılan optimizasyon algoritmaları hakkında yapılan çalışmalar yer almaktadır [23,25–27]. Bir kuantum algoritmanın optimizasyonu donanımdan bağımsız ve donanıma bağımlı olmak üzere iki aşamada optimize edilebilir [28]. Donanımdan bağımsız optimizasyon, devre boyutu veya devre derinliği gibi bir maliyet fonksiyonuna göre kuantum devre düzeyinde genel optimizasyonlar yapılır [22,29]. Donanıma bağımlı optimizasyonda ise hedef kuantum bilgisayarın mevcut kapı seti, kübitlerin hata oranları, kübitlerin bağlantı özellikleri veya farklı kübitlerin süperpozisyon halinden klasik davranışa dönüşme süresi (*decoherence*) gibi donanıma özgü özellikleri dikkate alınır [30].

Kuantum algoritmalarının optimize etmek için çeşitli çalışmalar mevcuttur. Bu çalışmalar arasında Heyfron ve Campbell, Clifford+T kapı setini kullanarak T kapılarının sayısını azaltan bir kuantum derleyicisi önermektedir. T kapısının maliyetinin diğer Clifford kapılarından çok daha yüksek olduğunu ve T kapılarının sayısını azaltarak devre maliyetini düşürdüklerini raporlamışlardır [31]. Itoko, derleme aşamasında kübitler ile kapıların eşlenmesini geliştirmek için bir yaklaşım sunmuştur. Bu yaklaşıma göre donanım özelliklerine bağlı olarak en uygun eşlemeyi sağlayan bir algoritma kullanılmıştır. Bu algoritma, belirli bir kuantum bilgisayarın donanım özelliklerini dikkate alarak, kuantum algoritmalarının uygun bir şekilde derlenmesi için gerekli olan kübitlerin ve kapıların belirlenmesine yardımcı olmaktadır [30]. Bu yaklaşım sayesinde kuantum algoritması daha verimli bir şekilde derlenebilir ve donanımın mevcut özellikleri en iyi şekilde kullanılabilir. Maslov, devre derinliğini yani, kübitlerde ardışık olarak yürütülen kapıların sayısını azaltmak için şablon tabanlı bir yaklaşımda bulunmuştur. Bu yöntem, devre derinliğinin azaltılması gibi çeşitli avantajlar getirmekte ve kuantum algoritmalarının daha verimli bir şekilde derlenmesini sağlamaktadır [32].

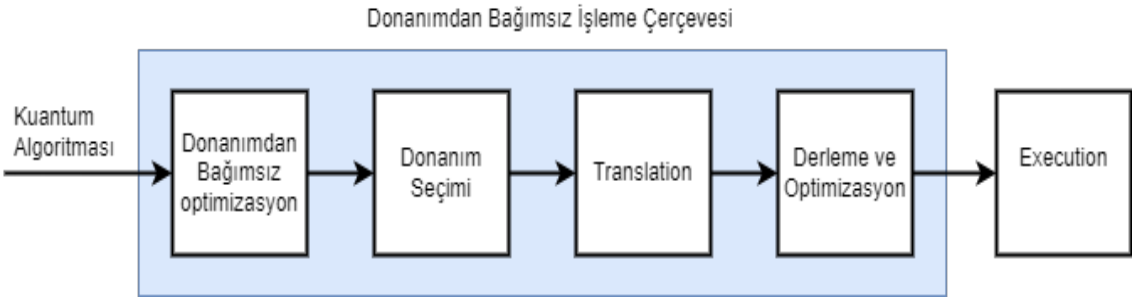
Verimli bir derleyici tasarlamak zor bir görevdir çünkü kuantum bilgisayarlarının karmaşıklığı ve donanım kısıtlamalarıyla başa çıkmak gerekmektedir. Bu nedenle literatürde sorunu ele alan birçok öneri ortaya çıkmıştır. Farklı öneriler bir araya getirilerek daha etkili ve optimize edilmiş derleyiciler geliştirilebilir. Bu konuda IBMQ çerçevesindeki kuantum derleyicisi bir örnektir [17]. IBMQ derleyicisi, farklı yönlerden sorunu ele alan ve çeşitli optimizasyon katmanlarına sahip bir yapıya sahiptir. Bu katmanlar, derleme işlemini daha etkili hale getirmek için farklı yöntemler ve teknikler kullanmaktadır.

4. Kuantum Derleyicilerin Mimari Analizi ve IBMQ Uygulaması

Derleyici tasarımlarının etkili bir şekilde tasarlanması zor bir görevdir. Bu nedenle, literatürde bu sorunu ele alan birçok öneri bulunmaktadır. Bu bölümde kuantum literatürde kuantum derleyiciler ile gerçekleştirilen çalışmalar analiz edilerek, yarım toplama devresinin IBM composer ortamında uygulaması gerçekleştirilmiştir. Farklı kübit bağlantılarına ve hata oranlarına sahip kuantum bilgisayarlar üzerinde gerçekleştirilen uygulamalar sonucu elde edilen sonuçlar yorumlanmıştır.

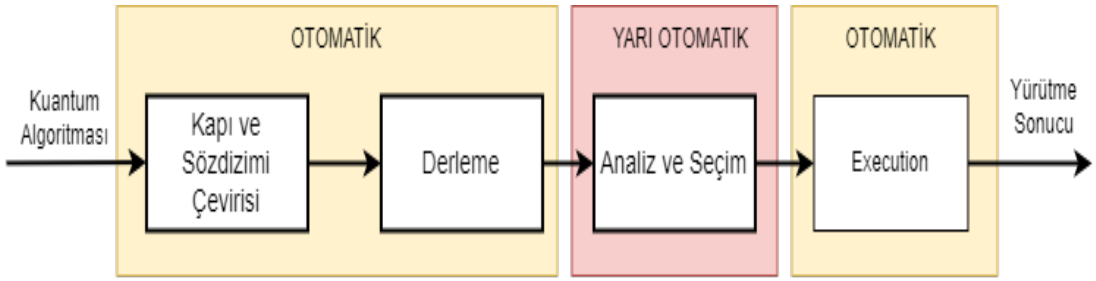
Bu alanda gerçekleştirilen bir çalışmada Frank Leymann, donanım bağımsız kuantum algoritmalarının işlenmesi için bir yöntem sunmuştur [24]. Gerekli işlem adımları Şekil 4'te verilmiştir. Yönteme göre öncelikle bağımsız bir kuantum programlama dili kullanarak kuantum algoritması tanımlanır. Kuantum algoritması daha önce açıklanan optimize yöntemlerle optimize edilir ve kuantum bilgisayarı için uygun hale getirilir. Kuantum algoritmasını karakterize eden önemli özelliklerden,

örneğin kübit sayısı veya kullanılan kapı seti gibi yararlanarak uygun kuantum donanımı seçilir [33]. Seçilen kuantum donanımında çalıştırılabilmesi için, destekleyen bir kuantum bilgisayar (IBM, Google, Honeywell vb.) [34] kullanılarak kuantum diline dönüştürülür ve seçilen kuantum donanımı için yürütülebilir bir forma derlenir.



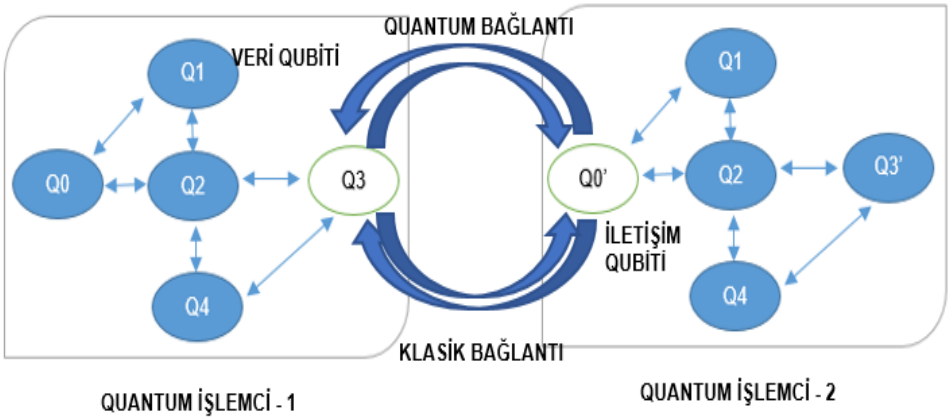
Şekil 4. Donanımdan bağımsız kuantum algoritmaları için gerekli işleme adımları [24].

Hangi kuantum derleyicinin daha uygun olduğunu tespit etmek yazılan algoritmanın farklı derleyicilerde ayrı ayrı derlenmesini ve derleme sonuçlarının karşılaştırılmasını gerektirmektedir. Bu işlem oldukça zaman almaktadır. Bu zorlukların üstesinden gelmek için Marie Salm, seçilen kuantum algoritma ve kuantum bilgisayar için birden fazla kuantum derleyicinin otomatik olarak denenmesini sağlayan bir çerçeve önermiştir [9]. Genel olarak bu yaklaşım Şekil 5'te gösterilmiştir. Yaklaşıma göre öncelikle, verilen kuantum devresi seçilen derleyicilerin SDK'ları tarafından desteklenen programlama diline ve uygun kapı setine çevrilir. Çevrilen devreler ve seçilen kuantum bilgisayarı hakkındaki bilgiler ikinci adımda derleme için girdi olarak kullanılır. Derleme işleminden sonra, oluşan devrelerin genişlikleri ve derinlikleri NISQ Analyzer kullanılarak analiz edilir ve kullanıcıya uygun sonuçlar önerilir [35]. Kullanıcı daha sonra en uygun derlemeyi manuel olarak seçer ve seçilen devre yürütülür.



Şekil 5. Çoklu derleyicilerle devrelerin otomatik derlemesi ve karşılaştırılması [9].

Ferrari ve arkadaşları Şekil 6'da mimari modeli gösterilen dağıtık bir kuantum hesaplama önerisinde bulunmuşlardır [36]. Dağıtık model ile NISQ çağındaki kuantum bilgisayarların sınırlamalararına çözüm olmayı hedeflemişlerdir. Sistem, birden fazla küçük ölçekli kuantum bilgisayarın birbiriyle iletişim halinde birleşerek büyük ölçekli kuantum bilgisayarını gerçekleştirmektedir. Klasik bilgileri iletmek amacıyla klasik bağlantı kurulurken, kuantum işlemleri gerçekleştirmek için kuantum bağlantı kullanılır. Her iki işlemcide de en az bir iletişim kübiti ayrılır ve kuantum bağlantısında Bell durumları iletilerek dolanık hale getirilir. Dolanık hale gelen kübitlere CNOT işlemini uygulayarak veri iletimini sağlamaktadır. Devredeki her CNOT işleminde uzaktan bir CNOT uygulanması gerektiğinden çalışmalarında, iki farklı dolaşma (*entanglement*) stratejisi kullanmışlardır. Önerilen model konsept model olup, yazılan kuantum algoritmaların bu mimaride çalışabilecek şekilde uygun derlenmesi için farklı derleyici tasarımlarının geliştirilmesi bu alandaki yeni çalışma konularını oluşturmaktadır.



Şekil 6. Dağıtık kuantum hesaplama için önerilen model [36].

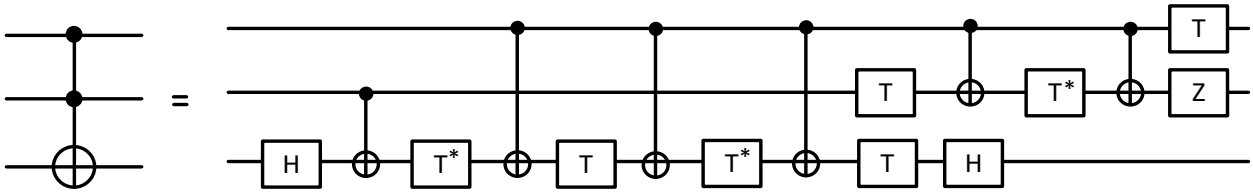
Kuantum hesaplama alanında konu ile ilgili diğer çalışmalar destekledikleri mimari, kullandıkları optimizasyon yöntemleri ve uygulama alanlarına göre Çizelge 2'de karşılaştırmalı olarak verilmiştir.

Çizelge 2. Derleyicilerin mimari, optimizasyon yöntemi ve uygulama alanı bakamından analizi

Çalışma	Desteklenen Mimari	Optimizasyon Yöntemi	Uygulama Alanı
[9]	Genişletilmiş HTTP REST API ve NISQ Analyzer UI	-	Kuantum Algoritma-Derleyici Analizi
[37]	Pauli dizisi merkezli mimari	VQE (varyasyonel kuantum özçözücü)	Kimya Simülasyonları
[38]	NFC tabanlı iletişim ağı mimarisi	BB84 protokolü	Kriptografi
[39]	IBM Q16 Melbourne kuantum mimarisi	Makine öğrenimi	Kuantum devre çıktısının değerlendirilmesi
[40]	Genel NISQ (Gürültülü Orta Ölçekli Kuantum) ve FT (Hata Tolereli) Kuantum Mimarileri	SQUARE (Stratejik Kuantum Ancilla Bit Yeniden Kullanımı)	Modüler kuantum program optimizasyonu
[41]	FT (Hata Tolereli) Kuantum Mimarileri	Code Teleportation (Kod Teleportasyonu)	Kuantum hata düzeltme kodlarının entegrasyonu
[42]	QuTech	-	Kuantum veritabanı
[36]	Distributed (Dağıtık), IBM Yorktown ve IBM Melbourne	Eşlenme Takası Temelli Strateji, Veri Kübiti Takası Temelli Strateji	Genel Hesaplamalar

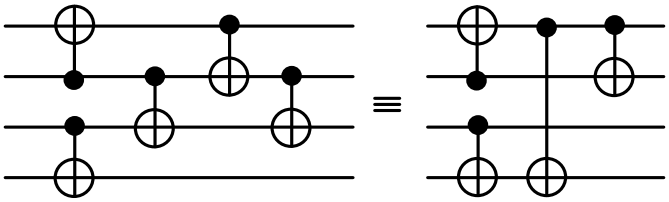
Kuantum derleyiciler yüksek seviyeli programlama dilini QASM diline optimum bir şekilde çevirmeyi gerçekleştirmenin yanında bu dilin uygun bilgisayar mimarisinde çalışması için gerekli işlemleri de gerçekleştirmektedir. Burada gerçekleşen temel adım aşağıda açıklanmıştır:

- Kuantum devrenin kuantum bilgisayarın çalıştırabileceği kapılara ayrıştırılması (Decomposition): Kuantum algoritmaların kuantum bilgisayarın deteklediği kapılar cinsinden yazılması gerekmektedir. Örneğin Toffoli kapısı doğrudan kuantum bilgisayarlarda çalışamaz. Bunun için ilgili kuantum bilgisayarın çalıştırabileceği kapılar cinsinden dönüşümü gerekmektedir. Şekil 7'de Toffoli kapısının H, T, Z ve CNOT kapıları cinsinden yazımı gösterilmiştir.



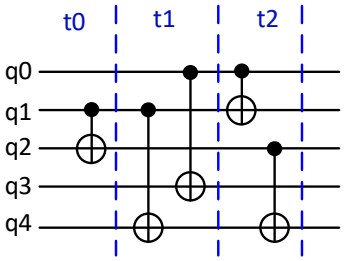
Şekil 7. Toffoli kapısının H, T, Z ve CNOT kapıları cinsinden oluşturulması [43].

- Devre elemanlarının optimizasyonu (Optimization): Kuantum devrelerin daha az kapı ile temsil edilmesi, kübitlerin daha kısa süre saklanması gerekliliğini doğuracağından ortaya çıkabilecek gürültü miktarının azalmasında olumlu etki göstermektedir. Bu optimizasyon işlemi tekrar eden kapıların birbirini sadeleştirmesi şeklinde yapılabileceği gibi aynı işlemi yapan kuantum devre karşılıklarının elde edilmesi ile de gerçekleşebilir. Şekil 8'de örnek bir optimizasyon işlemi gösterilmiştir.



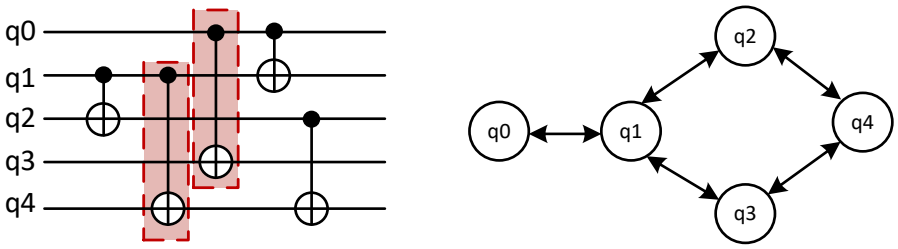
Şekil 8. Devre üzerinde optimizasyon işlemi

- Aynı anda çalışabilecek kapıların belirlenmesi ile paralelleştirirlmesi (Scheduling): Kuantum devrelerde farklı kübitler üzerinde gerçekleşen işlemler aynı t anında yapılabilmektedir. Aynı anda çalışabilecek kapıların belirlenmesi örneği Şekil 9'da verilmiştir. Şekilde CNOT(q1,q4) ve CNOT(q0,q3) kapıları farklı kübitler üzerinden işlem yaptığından aynı t1 anında gerçekleştirilmektedir.



Şekil 9. Aynı anda çalışabilecek kapıların belirlenmesi

- Donanım kübit bağlantısına bağlı olarak kübit eşleştirmelerinin gerçekleştirilmesi (Mapping): Kuantum bilgisayarlar farklı kübit bağlantılarına sahip yapıda olabilir ve genellikle tüm kübitlerin birbiri ile fiziksel bağlantıları bulunmaz. Şekil 10'da örnek bir kuantum devre ve bu devrenin çalıştırılacağı kuantum bilgisayarın kübit bağlantısı verilmiştir. Şekilde kırmızı ile gösterilen CNOT kapılarının uygulanması için q1-q4 ve q0-q3 arasında donanım üzerinde doğrudan bağlantı olmadığı görülmektedir. Bu durumda kübit değiştirme (*swapping*) işlemi uygulanarak eşleştirme işlemi gerçekleştirilebilir.



Şekil 10. Kübit eşleştirmesinin gerçekleştirilmesi

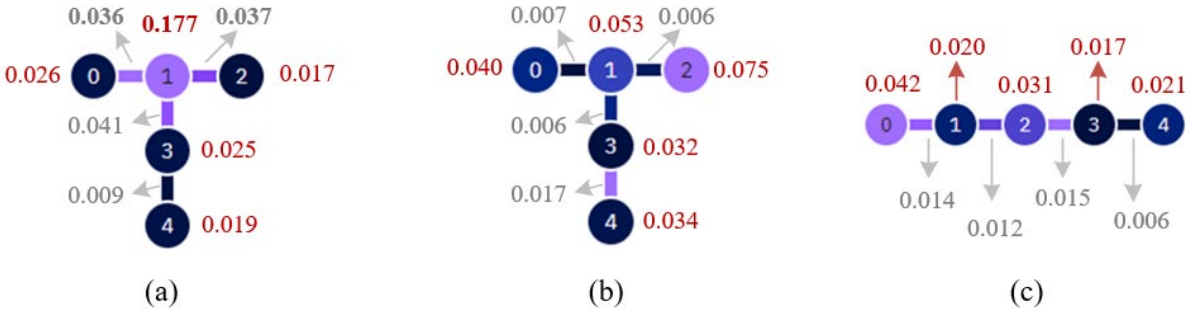
- Hata töleransını en aza indirgemek için iyileştirmeler yapılması (Fault Tolerance): Kuantum bilgisayarlarda kübitlerin ve bağlantıların hata değerleri bulunmaktadır. Bir kuantum devre bu hatalardan minimum etkilenecek şekilde ayarlanmalıdır.

Yukarıda sayılan adımlar IBM kuantum devre oluşturucu (*IBMQ Composer*) tarafından oluşturulan devrelerin uygun kuantum bilgisayara göre dönüşümde de (*transpiling*) uygulanmaktadır. Bu çalışmada yarım toplayıcı devresinin 5 kübitlik ibm_belem, ibm_quito ve ibm_manila bilgisayarlarda uygulaması gerçekleştirilmiştir. Yarım toplayıcı devresi ve karşılık gelen QASM kodu Şekil 11'de verilmiştir. Burada q[0] ve q[1] girişleri 1 olarak ayarlanmıştır. Sonuç q[2]q[1] kübitlerine yansıtılmıştır. Dolayısı ile sonucun $|10\rangle$ (2) olması beklenmektedir.

(a) (b)

Şekil 11. Yarım toplayıcı kuantum devrenin IBM composer ortamında oluşturulması ve karşılık gelen QASM kodu. a) Yarım toplayıcı kuantum devre, b) QASM kod karşılığı

Örnek uygulama ibm_belem, ibm_quito ve ibm_manila kuantum bilgisayarları için test edilmiştir. Kuantum bilgisayarların sonuçlarının güvenilirliğini belirlemek üzere *Quantum Volume* (QV) metriği kullanılır. Bu bilgisayarlar sırası ile 16, 16 ve 32 QV değerlerine sahiptir. Bu kuantum bilgisayarlardan ibmq_belem ve ibmq_quito T tipi bağlantıya sahipken, ibmq_manila L tipi bağlantıdadır. Bilgisayarların kübit bağlantıları Şekil 12'de verilmiştir. Şekilde kübitlerin renkleri çalışmanın gerçekleştiği sıradaki okuma hatalarını, bağlantı renkleri ise CNOT uygulama hatalarını temsil etmektedir. Burada koyu renkler hatanın daha düşük olduğunu göstermektedir. Her renklendirme kendi içindeki kübitler üzerinde yapıldığından bu renkler karşılaştırma için yeterli olmamaktadır. Bu sebeple kübit okuma hataları kırmızı renkle, CNOT bağlantı hataları ile gri renkte gösterilmiştir. Şekil 12.a'da ibmq_belem'in 1. Kübitindeki okuma hatasının ve CNOT bağlantı hatalarının yüksek olması dikkat çekmektedir.

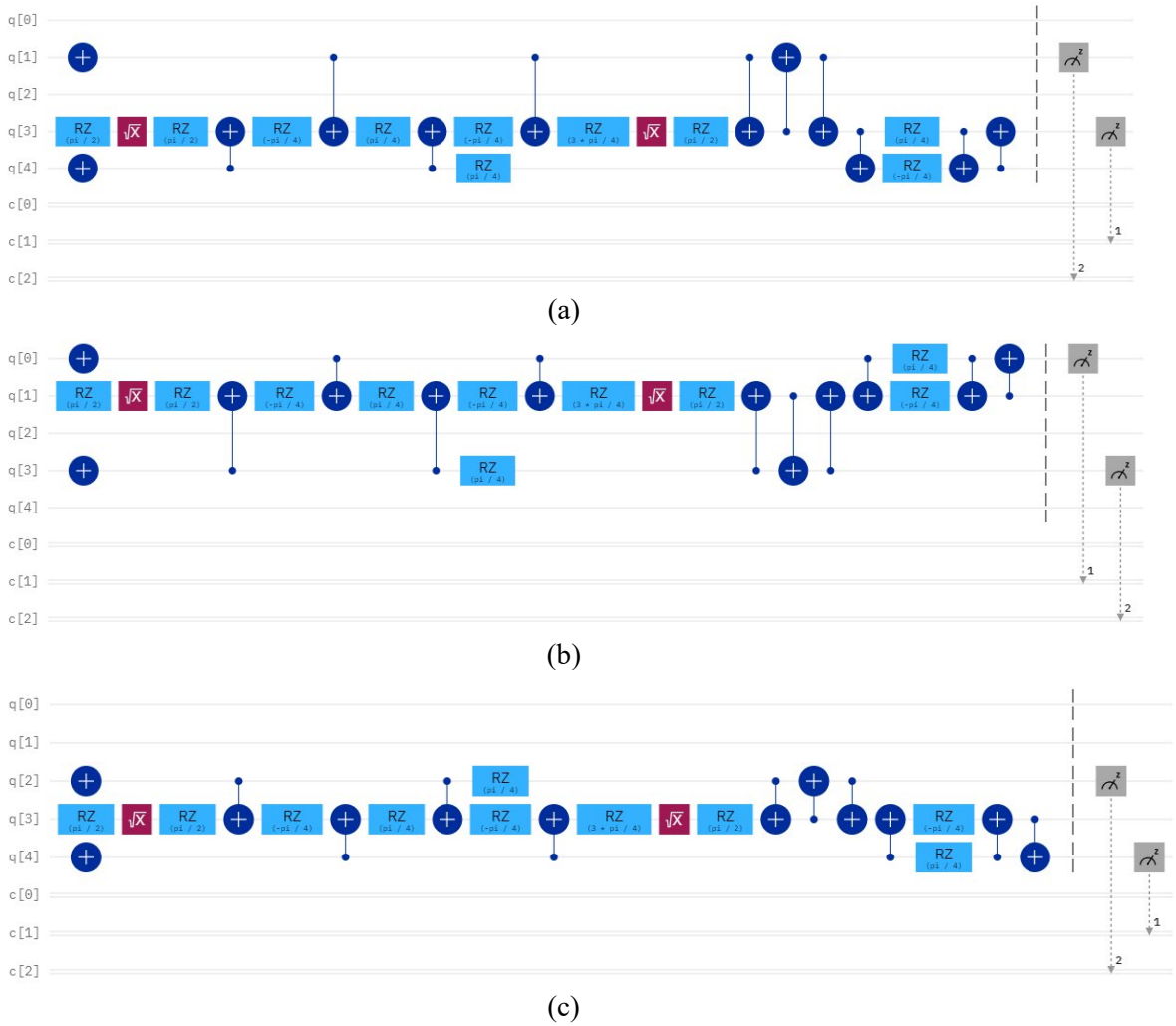


(a) (b) (c)

Şekil 12. IBM kuantum bilgisayarlarının kübit bağlantıları ve hata değerleri. Kırmızılar kübit hataları, griler CNOT bağlantı hatalarını göstermektedir. a) ibmq_belem, b) ibmq_quito, c) ibmq_manila

Orijinal devresi Şekil 11'de verilen kuantum devrenin, kuantum bilgisayarların çalıştırabileceği kapı kümesi cinsinden dönüşümü Şekil 13'te verilmiştir. Her üç dönüşüm devresi de aynı işlemi gerçekleştirmekte olup farklı kübitleri kullanmaktadır. Sonuçta gözlem işlemi yapılırken ilgili kübitler

c[2] ve c[1] bitlerine yazdırılır. Bu sayede algoritma kuantum bilgisayarın mimarisine uygun hale getirilirken sonuç değeri değişmez.



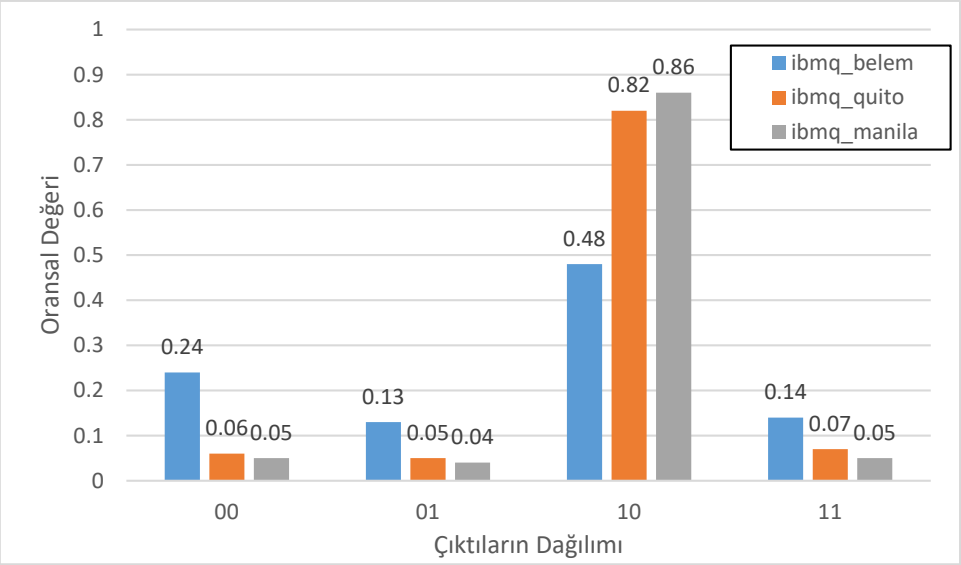
(a)

(b)

(c)

Şekil 13. Şekil 11'deki orijinal devrenin IBMQ bilgisayarlarda çalıştırılabilir hale dönüştürülmesi. a) ibmq_belem, b) ibmq_quito, c) ibmq_manila

Aynı kuantum devrenin uygun kuantum bilgisayarlarına göre dönüşümlerinin gerçekleştirilmesi sonucu çalıştırılması ile ürettiği sonuçlar Şekil 14'te verilmiştir. Şekilde ibmq_belem bilgisayarının performansının diğer kuantum bilgisayarlarına göre düşük kaldığı görülmektedir. Bu durum, dönüştürülmüş devresinin 1, 3 ve 4 nolu kübitleri kullanmış olması ve 1 nolu kübitin okuma hatasının kabul edilebilir seviyenin üstünde olmasına bağlanmaktadır.



Şekil 14. Örnek uygulamanın gerçek kuantum bilgisayarlarda 1024 *shots* çalıştırılması ile ürettikleri sonuçların oransal dağılımı.

5. Sonuçlar

Kuantum hesaplama ve programlamanın klasik bilgisayarlarımıza getirdiği üstünlüğün kabul edilmesi bu alanda yapılan çalışmaları hızlandırmış ve siber güvenlik, kimya, finans gibi çeşitli sektörlerde çığır açacak gelişmeler sağlayacağı öngörülmüştür. Ancak günümüz teknoloji sınırlamaları nedeniyle bu bilgisayarlarda donanımsal hatalar meydana gelmektedir. Kuantum gürültüsü yaşanan gelişmelerin bir noktada sınırlanmasına, yavaşlamasına, çeşitli problemlerin ortaya çıkmasına neden olmuştur. Kuantum bilgisayarların çalıştırılması için özel olarak tasarlanmış kuantum algoritmalarına ihtiyaçları vardır ve bu algoritmaların oluşturulması için uygun bir derleyiciye ihtiyaç duyulmaktadır. Derleyiciler, programlama dili seviyesinde yazılmış kuantum algoritmalarını kuantum devrelerine çevirir ve kuantum bilgisayarlarda çalıştırılabilir duruma getirir. Derleyiciler, kuantum bilgisayarın özelliklerine ve sınırlamalarına dikkat eder. Farklı kuantum bilgisayarlar ve mimariler için algoritmalarını kuantum devrelerine dönüştürürken kullanılacak en uygun kapı seti kümesini seçerek, daha az sayıda kuantum biti kullanarak daha büyük ve karmaşık problemleri çözmeyi mümkün kılar.

Kuantum derleyiciler, kuantum programlarımızı/algoritmalarımızı kuantum bilgisayarında işlenebilen kübit talimatlarını çevirmeyi sağlayan, bunu yaparken hata önlemeye ve kuantum hesaplamayı hızlandırcak çeşitli iyileştirmeler yapmaya çalışır. Klasik programlamaya benzer olarak kuantum programlama şu aşamaları barındırır: Algoritmanın yazılması, yüksek seviye derleme, düşük seviye derleme-kuantum devrenin elde edilmesi, kuantum optimizayon-kuantum hata düzeltmesi, donanımda çalıştırılması, sonuçların ölçülmesi. Uygun derleyici ve kuantum bilgisayar seçimi, kuantum devresinin en uygun şekilde derlenmesini sağlayarak devrenin çalıştırılma süresinin düşmesini ve dolayısıyla hata oranını düşürebilir. Gürültülü Orta Ölçekli Kuantum (NISQ) çağındaki kuantum donanımından tam verim alabilmek için devrelerimizi algoritmamıza en uygun derleyiciyi elde etmemiz son derece önemlidir.

Günümüzde kullanıcı bağımsız veya kullanıcı bağımlı birçok derleyici, programlama kütüphaneleri, yazılım çerçeveleri bulunmakla beraber genelde kuantum devreler soyut olarak (donanımdan bağımsız) oluşturulur. Çalışmada soyut devrenin gerçek bir devreye dönüşüm aşamaları bakımından, kullandığı mimari, kullanım alanı, tasarımı açısından birbirinden farklı derleyicilere yer verilerek IBM composer ortamında gerçek kuantum bilgisayarlarda soyut devrenin uygulanarak analiz edilmesi sağlanmıştır. Yapılan çalışmada 5 kübtlik ibmq_belem, ibmq_quito ve ibmq_manila bilgisayarlarına yer verilmiştir. Bu bilgisayarlardan ibmq_belem ve ibmq_quito T tipi kübit bağlantılarına sahip iken, ibmq_manila L tipi kübit bağlantısına sahiptir. Yarım toplayıcı örneğinin bu

bilgisayarlarda çalıştırılması ile elde edilen dönüştürülmüş devreler gösterilerek, ürettiği sonuçlar analiz edilmiştir. Gerçekleştirilen analizler sonucunda L tipi bağlantıya sahip ibmq_manila bilgisayarının problemimiz için daha iyi sonuçlar ürettiği gözlemlenmiştir. Diğer taraftan aynı tip kübit bağlantı yapısına sahip olmasına rağmen ibmq_belem bilgisayarının ürettiği sonuçlar gözle görülür bir şekilde yetersiz kalmaktadır. Bunun sebebi ibmq_belem bilgisayarnının kullandığı kübitlerdeki, kübit okuma ve CNOT kapısı uygulama gibi hata oranlarının yüksek olması olarak yorumlanmaktadır. Gerçekleştirilen çalışma, geliştirilen kuantum programının daha doğru çalışabilmesi için uygun dönüşümleri gerçekleştiren derleyicilerin ve yapısına uygun kuantum bilgisayarın seçilmesi gerektiğini gözler önüne sermektedir. Yapılan çalışma ile kauntum programlama ve kuantum derleyicilerin yapıları hakkında bilgi verilmiş, uygulamalı kuantum hesaplamada daha başarılı sonuçlar elde etmek için uygun derleyici/donanım seçiminin önemi vurgulanmıştır.

Teşekkür

Bu çalışma; Türkiye Bilimsel ve Teknolojik Araştırma Kurumu tarafından TÜBİTAK 1001 programı kapsamında 121E439 nolu proje ile desteklenmiştir.

Çıkar Çatışması Beyanı

Yazarların bir kurum, kuruluş veya kişi ile mali çıkar çatışması yoktur ve yazarlar arasında çıkar çatışması bulunmamaktadır.

Kaynaklar

[1] Feynman RP. Simulating physics with computers. Int j Theor phys. 2018;21(6/7).

[2] Benioff P. The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines. Journal of statistical physics. 1980;22:563-91.

[3] Yetiş H, Karaköse M. A New Framework Containing Convolution and Pooling Circuits for Image Processing and Deep Learning Applications with Quantum Computing Implementation. Traitement du Signal. Nisan 2022;39(2):501-12.

[4] Bova F, Goldfarb A, Melko RG. Commercial applications of quantum computing. EPJ quantum technology. 2021;8(1):2.

[5] Arute F, Arya K, Babbush R, Bacon D, Bardin JC, Barends R, vd. Quantum supremacy using a programmable superconducting processor. Nature. 2019;574(7779):505-10.

[6] Hassija V, Chamola V, Saxena V, Chanana V, Parashari P, Mumtaz S, vd. Present landscape of quantum computing. IET Quantum Communication. 2020;1(2):42-8.

[7] Salm M, Barzen J, Leymann F, Weder B. Prioritization of compiled quantum circuits for different quantum computers. İçinde: 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER). IEEE; 2022. s. 1258-65.

[8] Preskill J. Quantum Computing in the NISQ era and beyond. Quantum. 06 Ağustos 2018;2:79.

[9] Salm M, Barzen J, Leymann F, Weder B, Wild K. Automating the comparison of quantum compilers for quantum circuits. İçinde: Symposium and Summer School on Service-Oriented Computing. Springer; 2021. s. 64-80.

[10] Miszczak J. High Level Structures for Quantum Computing. Springer Nature; 2022.

[11] Yetiş H, Karaköse M. Kuantum Uyarlamalı Genetik Algoritmalar için Çözüm Kalitesini Artıracak Yeni Bir Yaklaşım. Fırat Üniversitesi Mühendislik Bilimleri Dergisi. 2021;33:71-9.

[12] Mukai T. Completely scrambled memory for quantum superposition. Scientific reports. 2019;9(1):1147.

[13] SoniaLopezBravo. The qubit in quantum computing - Azure Quantum [Internet]. 2023 [a.yer 27 Temmuz 2023]. Erişim adresi: https://learn.microsoft.com/en-us/azure/quantum/concepts-the-qubit

[14] Coles PJ, Eidenbenz S, Pakin S, Adedoyin A, Ambrosiano J, Anisimov P, vd. Quantum Algorithm Implementations for Beginners. :77.

[15] Yetis H, Karaköse M. Investigation of Noise Effects for Different Quantum Computing Architectures in IBM-Q at NISQ Level. Içinde: 2021 25th International Conference on Information Technology (IT). Zabljak, Montenegro: IEEE; 2021

[16] Khammassi N, Ashraf I, Someren JV, Nane R, Krol AM, Rol MA, vd. Openql: A portable quantum programming framework for quantum accelerators. ACM Journal on Emerging Technologies in Computing Systems (JETC). 2021;18(1):1-24.

[17] BM Quantum [Internet]. [a.yer 25 Nisan 2022]. IBM Quantum. Erişim adresi: https://quantum-computing.ibm.com/

[18] Hidary JD, Hidary JD. Quantum computing: an applied approach. C. 1. Springer; 2019.

[19] Bar NF, Yetis H, Karakose M. An Approach Based on Quantum Reinforcement Learning for Navigation Problems. İçinde: 2022 International Conference on Data Analytics for Business and Industry (ICDABI). IEEE; 2022. s. 593-7.

[20] Botea A, Kishimoto A, Marinescu R. On the complexity of quantum circuit compilation. İçinde: Proceedings of the International Symposium on Combinatorial Search. 2018. s. 138-42.

[21] Soeken M, Meuli G, Schmitt B, Mozafari F, Riener H, De Micheli G. Boolean satisfiability in quantum compilation. Philosophical Transactions of the Royal Society A. 2020;378(2164):20190161.

[22] Yetiş H, Karaköse M. An improved and cost reduced quantum circuit generator approach for image encoding applications. Quantum Information Processing. 01 Haziran 2022;21:203.

[23] Bishop LS, Bravyi S, Cross A, Gambetta JM, Smolin J. Quantum volume. Quantum Volume Technical Report. 2017;

[24] Leymann F, Barzen J, Falkenthal M, Vietz D, Weder B, Wild K. Quantum in the cloud: application potentials and research opportunities. arXiv preprint arXiv:200306256. 2020;

[25] Steiger DS, Häner T, Troyer M. ProjectQ: an open source software framework for quantum computing. Quantum. 2018;2:49.

[26] Zhang Y, Deng H, Li Q, Song H, Nie L. Optimizing quantum programs against decoherence: Delaying qubits into quantum superposition. İçinde: 2019 International Symposium on Theoretical Aspects of Software Engineering (TASE). IEEE; 2019. s. 184-91.

[27] Salm M, Barzen J, Leymann F, Weder B. About a criterion of successfully executing a circuit in the NISQ era: what wd$\ll 1/\epsilon$ eff really means. İçinde: Proceedings of the 1st ACM SIGSOFT International Workshop on Architectures and Paradigms for Engineering Quantum Software. 2020. s. 10-3.

[28] Häner T, Steiger DS, Svore K, Troyer M. A software methodology for compiling quantum programs. Quantum Science and Technology. 2018;3(2):020501.

[29] Svore KM, Aho AV, Cross AW, Chuang I, Markov IL. A layered software architecture for quantum computing design tools. Computer. 2006;39(1):74-83.

[30] Itoko T, Raymond R, Imamichi T, Matsuo A. Optimization of quantum circuit mapping using gate transformation and commutation. Integration. 2020;70:43-50.

[31] Heyfron LE, Campbell ET. An efficient quantum compiler that reduces T count. Quantum Science and Technology. 2018;4(1):015004.

[32] Maslov D, Dueck GW, Miller DM, Negrevergne C. Quantum Circuit Simplification and Level Compaction. IEEE Trans Comput-Aided Des Integr Circuits Syst. Mart 2008;27(3):436-44.

[33] Suchara M, Kubiatowicz J, Faruque A, Chong FT, Lai CY, Paz G. Qure: The quantum resource estimator toolbox. İçinde: 2013 IEEE 31st International Conference on Computer Design (ICCD). IEEE; 2013. s. 419-26.

[34] McCaskey AJ, Lyakh DI, Dumitrescu EF, Powers SS, Humble TS. XACC: a system-level software infrastructure for heterogeneous quantum–classical computing. Quantum Science and Technology. 2020;5(2):024002.

[35] Salm M, Barzen J, Breitenbücher U, Leymann F, Weder B, Wild K. The NISQ analyzer: automating the selection of quantum computers for quantum algorithms. İçinde: Symposium and Summer School on Service-Oriented Computing. Springer; 2020. s. 66-85.

[36] Ferrari D, Cacciapuoti AS, Amoretti M, Caleffi M. Compiler design for distributed quantum computing. IEEE Transactions on Quantum Engineering. 2021;2:1-20.

[37] Li G, Shi Y, Javadi-Abhari A. Software-hardware co-optimization for computational chemistry on superconducting quantum processors. İçinde: 2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA). IEEE; 2021. s. 832-45.

[38] Han J, Liu Y, Sun X, Song L. Enhancing data and privacy security in mobile cloud computing through quantum cryptography. İçinde: 2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS). IEEE; 2016. s. 398-401.

[39] Saravanan V, Saeed SM. Test data-driven machine learning models for reliable quantum circuit output. İçinde: 2021 IEEE European Test Symposium (ETS). IEEE; 2021. s. 1-6.

[40] Ding Y, Wu XC, Holmes A, Wiseth A, Franklin D, Martonosi M, vd. Square: Strategic quantum ancilla reuse for modular quantum programs via cost-effective uncomputation. İçinde: 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA). IEEE; 2020. s. 570-83.

[41] Oskin M, Chong FT, Chuang IL. A practical architecture for reliable quantum computers. Computer. 2002;35(1):79-87.

[42] Chakraborty S. A Prototype For Quantum Database In Hybrid Quantum. 2022;

[43] Gerdt VP, Kragler R, Prokopenya AN. A mathematica package for simulation of quantum computation. İçinde: International Workshop on Computer Algebra in Scientific Computing. Springer; 2009. s. 106-17.