COMPUTERSCIENCE

CS@UH

Timing Analysis and Optimization of Quantum Computing Systems

Albert M. K. Cheng

Professor of Computer Science and Electrical & Computer Engineering

- Outline
  - Introduction
  - Elementary Introduction to Quantum Computing
  - Preliminaries of Functional Reactive Programming
  - Response Time Analysis
  - Hybrid Quantum-Classical Computing

3rd Workshop on OPtimization for Embedded and ReAl-time systems (OPERA) at RTSS

December 2, 2025

UNIVERSITY of HOUSTON

Introduction

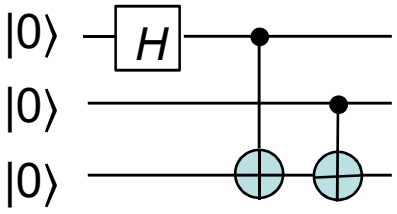


Fig. 1. Quantum Circuit

Quantum computing *[6] is a research and development priority, critical in maintaining national security and scientific leadership.*

- Quantum computing promises an **astronomical increase in computing capabilities**, providing super-fast response in calculating the solutions to some challenging and even intractable problems.
- However, **fast is not real-time; instead, "real-time" means "to be on-time every time"** [8]. Every instance of all computational and physical tasks must meet application-specific deadline, delay, periodicity, and other timing requirements. **Pre-deployment response time analysis is thus required.**
- **Can quantum computing machines and algorithms satisfy the specified timing constraints in real-time applications?**

Introduction

- There exist timing analysis techniques for classical real-time systems, **but will these hold for real-time quantum computing systems running quantum programs?**
- Current research in quantum computing focuses on constructing a universal gate set, making qubits initializable and easily readable, controlling/avoiding decoherance, and building scalable machines by increasing the number of qubits [12].
- However, the **timing analysis and verification of quantum computing machines and quantum programs has never been systematically explored**.
- Our project is therefore to pursue answers to **determining whether a given quantum computing machine (such as the first circuit-based IBM Q [19]) and quantum programs satisfy the timing constraints** imposed by a given real-time application.

Introduction

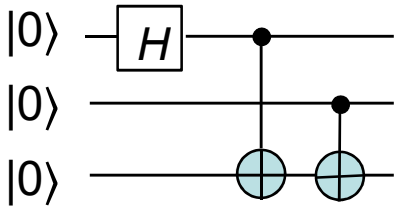


Fig. 1. Quantum Circuit

Quantum circuit: most popular quantum computing model.

Based on the *quantum bit* **qubit**, analogous to the *binary bit* in classical computing systems.

Quantum computers and classical computers are equivalent in terms of computability since both comply with the Church–Turing Thesis [26].

However, **quantum algorithms have several-fold lower runtime complexities than those of corresponding known classical algorithms** for important problems including:

cryptography, search, real-time machine learning, computational biology, and computer-aided drug design, resulting in "**quantum supremacy**."

Introduction

- A challenge to overcome is that quantum computers suffer from **quantum decoherence and state fidelity**, making it difficult to maintain qubits' quantum states.
- Therefore, quantum computers incur **errors resulting in wasted execution times**. **Error correction [16], [27] is needed** to isolate the system from its environment since interactions with the external world result in decoherence.
- This project has several threads of exploration -- one is detailed here.
- We **leverage our framework and extensive results in the timing analysis and optimization of functional reactive programming (FRP) systems** developed since 2009 [30].

Introduction

- In FRP systems, **a currently running tasks is terminated** (and not preempted for its execution to resume later at the task's preempted point) **when a more urgent task arrives and thus the completed execution steps of the former (lower priority) task are wasted**.
- The **processing time incurred in running these terminated tasks must be included in the worst-case response time (WCRT)** of the system when determining whether it satisfies the specified timing constraints in a given real-time application.
- Leverage the analysis of the FRP model to predict the WCRT of fault-tolerant classical computing systems since:
- **Timing analysis of re-executions for fault recovery plus transient-faults-induced wasted execution times is similar to determining the response time of tasks with terminated and incomplete executions in the FRP model. Accounting for wasted execution times due to errors in quantum computers can be treated similarly.**

Elementary Introduction to Quantum Computing

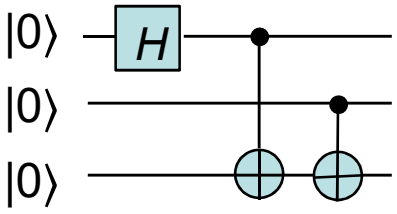


Fig. 1. Quantum Circuit

The most popular model of quantum computation specifies computation performed on a **network of quantum logic gates**, analogous to logic gates in conventional digital circuits.

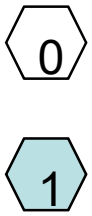
This model is an **abstract linear-algebraic generalization of a classical digital circuit** and **complies with the laws of quantum mechanics**.

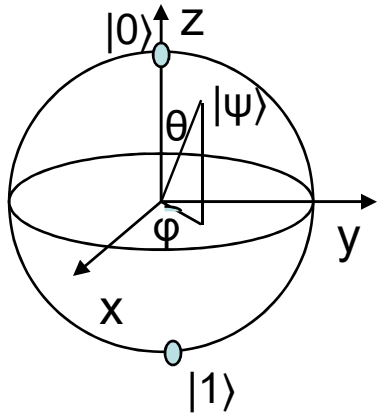
Quantum programs **execute over quantum states, which comprise one or more quantum bits (or qubits).**

Using Dirac's notation, a state denoted as $|\psi\rangle$ is called a **ket**.
The most elementary computing space in quantum computing is formed by a set of two orthogonal vectors represented by two kets: $|0\rangle$ and $|1\rangle$.

Elementary Introduction to Quantum Computing

- A **qubit** is a **normalized linear combination of these two vectors**.
- Therefore, while a regular bit in classical computers can only take two values, 0 or 1, a **qubit is a continuum of values in the Bloch sphere of radius 1**.





Bloch sphere is a geometrical representation of a two-level quantum mechanical system's pure state space.

Fig. 2. Classical bit. **Fig. 3. Qubit and Block sphere.**

- More precisely, a **single qubit is characterized by a vector of complex numbers** $\langle \alpha, \beta \rangle$ such that $|\alpha|^2 + |\beta|^2 = 1$. A complex number is represented by $p+qi$, where p and q are real numbers, and i is the imaginary unit satisfying $i^2 = -1$.
- Vector $\langle 1, 0 \rangle$ represents state $|0\rangle$ while the vector $\langle 0, 1 \rangle$ represents the state $|1\rangle$. In general, any computing space is a tensor product of several qubits.

Elementary Introduction to Quantum Computing

- **If the values of both α and β are non-zero, then a qubit is in a superposition of |0⟩ and |1⟩.**
- **A qubit is only in superposition until it is measured**, when the outcome will be 0 with probability $|\alpha|^2$ and 1 with probability $|\beta|^2$,
- as illustrated by Schrodinger's cat which is both dead and alive in a room until this room is opened.
- **The measurement has the effect of collapsing the state to match the measured outcome**, i.e., either |0⟩ or |1⟩,
- **and thus this measurement is not passive**. Consequently, all measurements afterwards return the same value.

Elementary Introduction to Quantum Computing

- An **operation** in any part of a quantum algorithm can execute on one or several of the total number of qubits in the quantum computing system.
- There are two basic types of quantum operations: **(1) reversible operations with unitary operators, and (2) irreversible operations called "measurements" or "observations,"** which are projections onto Eigen-vectors of the observation operator.
- In operation (2), **the module of the projection onto the observation space specifies the probability to observe a given state (|0⟩ or |1⟩ for a single qubit).**
- An Eigen-vector of the operator will be the subsequently updated state.
- Therefore, **observation is not a linear operation in the general case**, though it can be linear in specific configurations or when the normalization factor is unused.
- **The dimension of the computing space in the case of several qubits is the tensor product of the spaces, so it is exponentially larger than the Cartesian product in the current approach**, but the Cartesian product can be seen as a subset of the tensor product.

Elementary Introduction to Quantum Computing – *Quantum Decoherence*

- **Quantum computers suffer from quantum decoherence**, or loss of quantum coherence, which must be controlled or eliminated for computing to be valid.
- **The system is coherent if there exists a definite phase relation between different states**, necessary to perform quantum computing on quantum information encoded in quantum states.
- Since interactions with the external world cause the quantum computing system to decohere, isolating the system from its environment is required.
- The quantum gates and the background thermonuclear spin and lattice vibrations of the physical mechanism for implementing the qubits are also sources of decoherence.
- **Since decoherence is effectively non-unitary, it is irreversible, and thus it should be highly controlled if it cannot be avoided.**

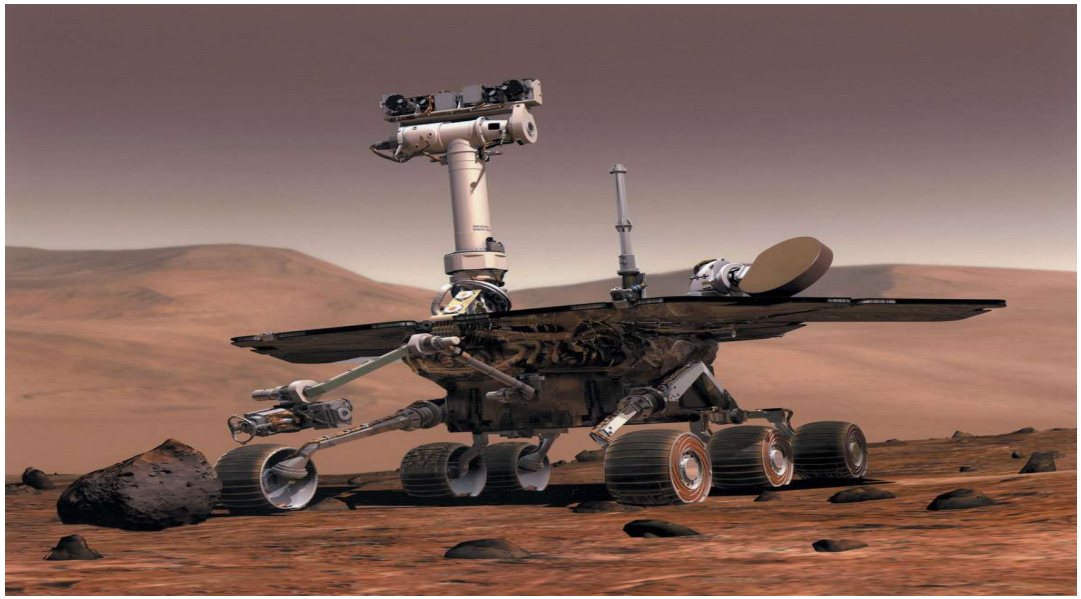
Elementary Introduction to Quantum Computing – *Quantum Decoherence*

- **Decoherence times** vary from nanoseconds to seconds at low temperature [11], [35].
- Cooling the qubits to 20 millikelvin can avoid significant decoherence [20]. Thus, **long tasks may render some quantum programs inoperable since keeping the state of qubits for a sufficiently long interval will eventually corrupt the superpositions [1].**
- **Response time analysis is therefore required to account for decoherence and error recovery times** in order to provide real-time performance guarantees.
- **If the error rate is sufficiently low, quantum error correction can be used to eliminate errors and decoherence** according to the quantum threshold theorem [26].
- Thus, the overall computation time can be longer than the decoherence time **if an error correction scheme can be implemented to recover from errors at a faster rate than the error rate caused by decoherence.**

Preliminaries of Functional Reactive Programming (FRP)

- Systems that **react to the environment being monitored and controlled in a timely fashion using functional reactive programming (FRP)** are known as Functional Reactive Systems (FRS).
- These systems can range from small devices (which are not a CPS) to distributed and complex components (a CPS).
- FRP is a style of functional programming where programs are inherently stateful, but automatically react to changes in state.
- Therefore, the program remains an algebraic description of system state, with the task of keeping the stated (unidirectional) relationships in sync left to the *language*.
- **FRP allows intuitive specification and formal verification of safety-critical behaviors**, thus reducing the number of defects during the design phase, and the **stateless nature of execution avoids the need for complex programming involving synchronization primitives**.
- More resistant to faults since there are no intermediate states. **FRP-programmed components are mathematical functions which can be composed more easily**.

☆ Motivation for FRP



Pathfinder mission to Mars: best known Priority Inversion problem.

Failure to turn on priority Inheritance (PI) - Most PI schemes complicate and slow down the locking code, and often are used to compensate for poor application designs.

http://research.microsoft.com/en-us/um/people/mbj/mars_pathfinder/mars_pathfinder.html

Functional Reactive Programming

- Priority-based Functional Reactive Programming (P-FRP)
- P-FRP provides real-time guarantees using static priority assignment
- Higher-priority tasks preempt lower-priority ones; preempted tasks are aborted
- Multi-version commit model of execution
- Atomic execution – “all or nothing” proposition
- Execution different from ‘standard’ models

Other Examples of Functional Programming (FP) Languages:

- Haskell
- Atom - Domain Specific Language in Haskell
- Erlang - Developed at Ericsson for programming telecommunication equipment
- Esterel - Designed for reactive programming
- F# - Developed by Microsoft; available as a commercial platform

Functional Reactive Programming (FRP)

CS@UH

- Type-safe programming
- Discrete and Continuous aspects
- Transactional model prevents priority inversion
- Synchronization primitives not required
- Ideal for parallel execution

Basic Abstractions

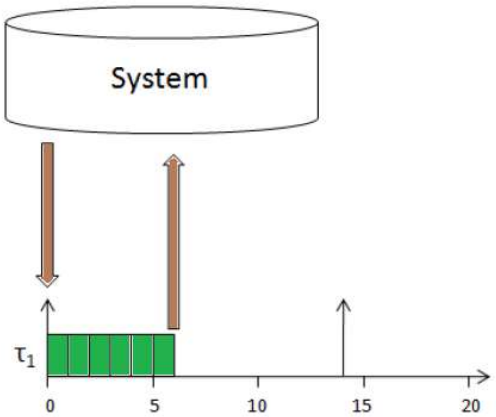
- FRP divides inputs into two basic classes:
  - Behaviors or signals: Functions of time.
  - Events: Temporal sequences of discrete values.
- An FRP language must include a means of altering or replacing a program based on event occurrences - this is the basis of FRP's reactivity.
- These abstractions may be reified in an FRP language or may form the basis of other abstractions, but they must be present.

Priority-based FRP (P-FRP)

- **In P-FRP, the scheduling model is called** Abort-and-Restart **(ANR)**
  - **Copy and restore operations**
- **To allow for correct restarting of handlers, compilation is extended to generate statements that store variables modified in an event handler into fresh *temporary* (or *scratch*) variables in the beginning of the handler while interrupts are turned off, and to restore variables from the temporary variables at the end of the handler while interrupts are turned off.**

Priority-based FRP (P-FRP)

τ_1 starts at 0 and it copies a set of data from the system. After six ticks, its work is done and then it restore the updated data into the system.



Copy-and-Restore Operation

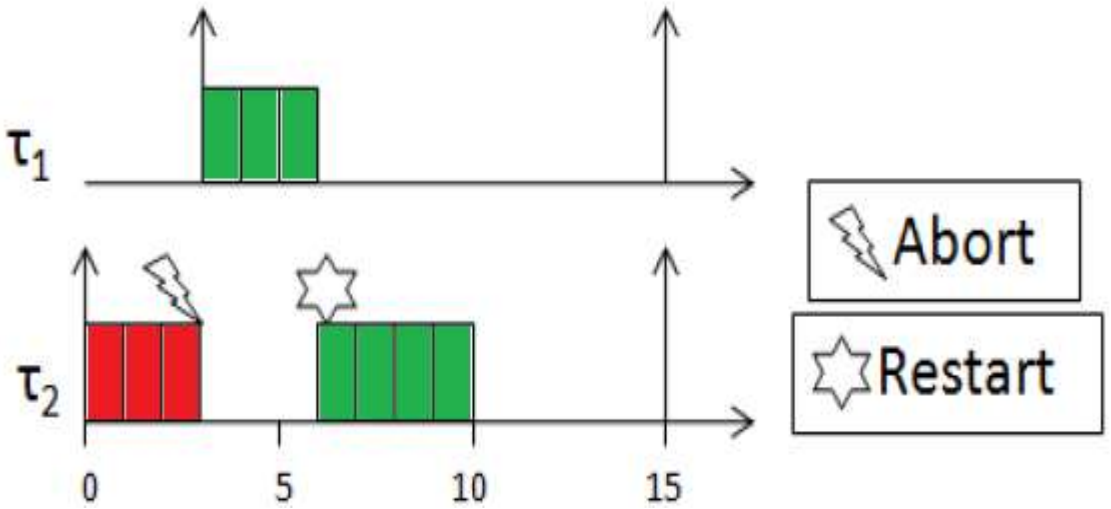
The Abort-and-Restart (ANR) Scheduling Model

The idea of the ANR model is that a lower-priority task is aborted when a higher priority task arrives into the system. Once the higher-priority task is done, the lower priority task restarts as new.

Priority-based FRP (P-FRP)
Example: Wasted execution times

Task	Period	WCET
τ_1	12	3
τ_2	15	4

(τ_1 has the highest priority)



Priority-based FRP (P-FRP) Example

Example: Automobile Anti-Lock Braking (ABS) Controller

- Activities of an ABS control system
  1. C = worst case execution time
  2. T = (sampling) period = D (deadline)

- (A) Car speed measurement: C= 1 ms, T= 5 ms
- (B) Wheel speed measurement: C= 2 ms,T=8 ms
- (C) Analysis and computation task : C= 3 ms,T=20 ms
- (D) Brakes (Abort (release) /Retry (pressure)) : C= 1 ms,T=25 ms

Kaleb R. Christoffersen and Albert M. K. Cheng, Model-Based Design: Anti-lock Brake System with Priority-Based Functional Reactive Programming, RTSS WIP 2013.

Priority-based FRP (P-FRP)

Example: ABS Controller - with RM scheduling

- The shortest repeating cycle / LCM = 200 ms

A's response time = 1 (Same as its own Computation Time)

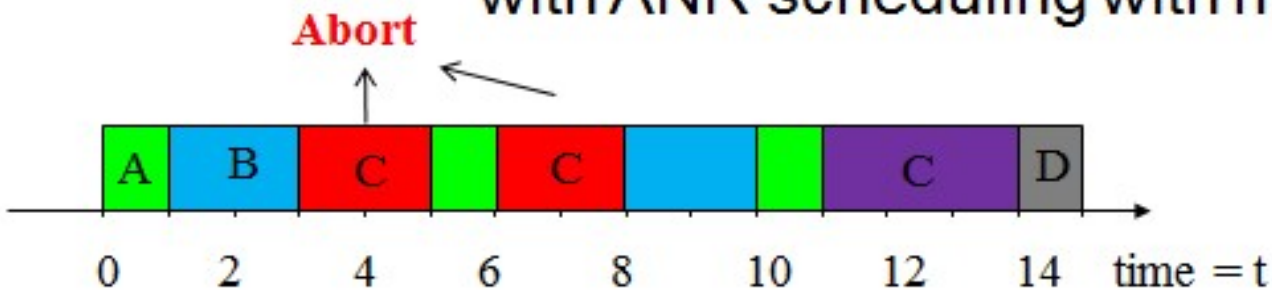
B's = 2 + 1 (time to execute task A) = 3

C's = 3 + 1 (A's) + 2 (B's) = 6

Priority-based FRP (P-FRP)

Example: ABS Controller –

with ANR scheduling with max 2 aborts.



A's response time = 1 (Same as its Computation Time)

B's = 2 + 1 (time to execute task A) = 3

C's = 3 + 1 (A's) + 2 (B's) = 14 < Deadline

Priority-based FRP (P-FRP)

- **Advantages of Abort-and-Restart (ANR)**
  - A simpler programming model
  - Tasks execute atomically so no task is blocked by another task
    - The priority inversion problem is removed
    - No overheads caused by priority inheritance
    - Closer adherence to priority scheduling
- The worst-case response time of a task is the length of the longest interval from a release of that task till its completion.
- With ANR, interference from higher-priority tasks induces both an interference cost and an abort cost on the response time of the preempted lower-priority task.
- **We have developed a comprehensive framework for response time analysis since 2009. Initial analysis abstracts memory and I/O access times. Recent work accounts for precise memory and I/O access times.**

Response Time Analysis of Priority-based FRP (P-FRP)

- Response time analysis is an **exact** schedulability test to calculate the worst-case response time of a task which includes the time of interference from other higher priority tasks and blocking from lower priority tasks.

- RTA is **not** exact unless **blocking** is exact - which it is not. If the worst-case response time of a task is longer than its deadline (***D***), it means the task will not meet its deadline. The opposite situation is that if the worst-case response time of the task is less than or equal to its deadline, the task will meet its deadline.

- The analysis can be applied for ***D* = *T*** (task's period), ***D* < *T***, or ***D* > *T***.

Priority-based FRP (P-FRP)

Response time Analysis for ANR

•For the highest-priority task, its worst response time will be equal to its own computation time, that is $R = C$.

- If task *j* has the highest arrival rate, then the execution time of a task *i* cannot exceed $T_j - C_j$ or task *i* will suffer **interference** (*I*) and **aborts** (**α**). So for a general task *i* :

$$R_i = C_i + I_i + \alpha_i$$

Priority-based FRP (P-FRP)

Interference Cost

•If the execution time of some task ***i*** exceeds $T_j - C_j$, then task ***i*** will **never** be able to complete execution.

•A simple expression for obtaining this Interference Cost is using the ceiling function:

$$I_i = \left\lceil \frac{R_i}{T_j} \right\rceil \cdot C_j$$

Priority-based FRP (P-FRP)

Maximum Interference

- Each task of higher-priority is interfering with task ***i***, *and so:*

$$I_i = \left\lceil \frac{R_i}{T_{i+1}} \right\rceil \cdot C_{i+1} + \left\lceil \frac{R_i}{T_{i+2}} \right\rceil \cdot C_{i+2} + \ldots + \left\lceil \frac{R_i}{T_n} \right\rceil \cdot C_n$$

- This gives us the following equation:

$$I_i = \sum_{j=i+1}^{n} \left\lceil \frac{R_i}{T_j} \right\rceil \cdot C_j$$

Priority-based FRP (P-FRP)

Maximum Abort Costs

•Each higher-priority task is interfering with task ***i***, so the maximum Abort Costs are as follows:

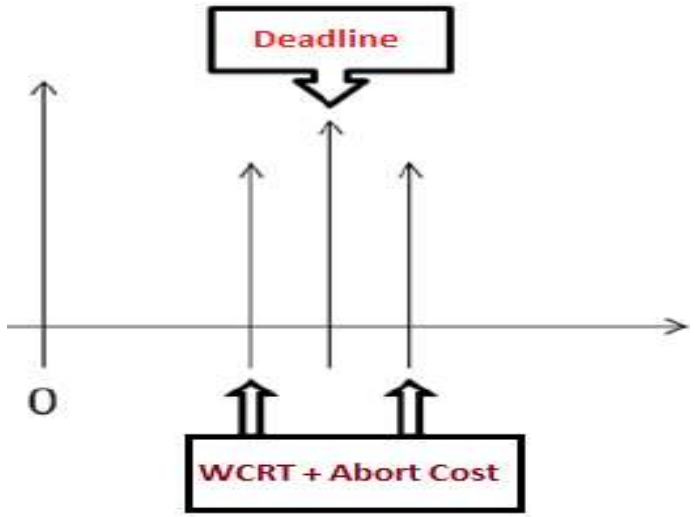
$$\alpha_i = \sum_{j=i+1}^{N} \left\lceil \frac{R_i}{T_j} \right\rceil \cdot \max_{k=i}^{j-1} C_k$$

•C_k is the maximum execution time between ***i*** and the highest-priority task.

Priority-based FRP (P-FRP)

Maximum Abort Costs

•The maximum abort cost equation is sensible and simple but overly pessimistic. Therefore, the test is said to be **sufficient** but not **necessary**.



Priority-based FRP (P-FRP)

- Abort-and-Restart with a limit on the number of aborts

$$R_i \leq C_i + I_i + B_i + \alpha_i$$

$$I_i = \sum_{j=i+1}^{n} min(MaxNumberAborts, \left\lceil \frac{R_i}{T_j} \right\rceil) \cdot C_j$$

$$B_i = \max_{j \in LowerPriority(i))} C_j \quad (Blocking\ Costs)$$

$$\alpha_i = MaxNumberAborts \cdot \sum_{j=i}^{n-1} C_j$$

Response Time Analysis of Priority-based FRP (P-FRP)

Algorithm 1 P-FRP Exact Schedulability Test Algorithm

Input: $\Gamma_n, [0, LCM)$
Output: True/False, (schedulable or not)

1: $\sigma_n([0, LCM)) \leftarrow \{[0, LCM)\}$
2: **for** $\tau_i = n \rightarrow 2$ **do**
3: $\quad \sigma_{i-1}([0, LCM)) \leftarrow \lambda(\sigma_i([0, LCM)), \Gamma_n)$
4: $\quad$ **if** $|\sigma_{i-1}([0, LCM))| = 0$ **then**
5: $\quad\quad$ **return** false
6: $\quad$ **end if**
7: **end for**
8: **return** $\leftarrow \mu(\sigma_1([0, LCM)), C_1)$

- On-line Schedulability Test returns the gap (the amount of execution time available) for the next lower-priority task.
- Precise (tight) timing characterization of the embedded controller software execution leads to faster physical system response compared with one designed without accurate controller timing analysis (and thus requires more tolerance of execution time variations).

Response Time Analysis of Quantum Computing Systems

- There is limited work in one direction: **using quantum computing or formalism inspired by quantum computing [24] to determine the Worst-Case Execution time (WCET) of computer programs running on non-quantum computers** accounting for cache and task preemptions while avoiding the underlying NP-hard analysis problems.
- However, we are not aware of work in the other direction: **leveraging the timing analysis of functional reactive systems to model and determine the Worst-Case Response Time (WCRT) of quantum programs running on quantum computers.**
- There is work such as a **fully verified optimizer for quantum circuits, written with the Coq proof assistant [18], but it does not consider timing constraints.**

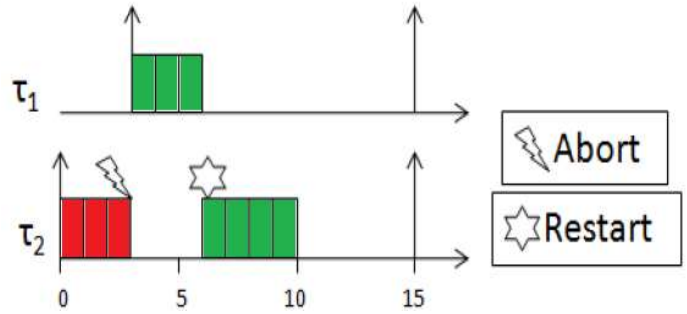
Response Time Analysis of Quantum Computing Systems

- With appropriate transformations, the **behavior of quantum programs resembles that of P-FRP programs and thus a corresponding P-FRP analysis strategy [39] can be applied.**
- As in [24], we model any interaction of the computing space with the hardware component as a Finite State Machine (FSM).
- We next associate a component of the vector state with each state of the FSM and then use simple transition matrices to represent operations for transitioning from one state to another.
- The linear operation is a matrix product with a state vector.
- Since a simple matrix product is used to update the states, the **framework can be considered as a Markov chain [4], [25], which is also deterministic for a deterministic FSM.**

Response Time Analysis of Quantum Computing Systems

- **Optional elements of the state space can keep track of the relevant parts of the state history such as the number of hits and misses in the cache, which is the memory component contributing to the exponential-time complexity of response time analysis of non-quantum programs.**
- It is updated by the state vector before state update. Using Dirac's notation, $|x\rangle$ denotes the state vector associated with state x and $|h\rangle$ denotes the history state vector.
- Then an update of state vectors is expressed as follows: $|h'\rangle = |h\rangle + P|x\rangle$ and $|x'\rangle = O|x\rangle$, where following the update, primed state vectors become state vectors, P is typically a projector for the state's interesting parts for the history state vector, and O is the state-transition matrix.
- **Since our approach is to utilize efficient FRP-based timing analysis [39] to determine the WCRT of quantum programs, we perform the inverse of the above steps and then apply our P-FRP analysis strategy [39].**

Response Time Analysis of Quantum Computing Systems



- Our approach to account for quantum decoherence times is motivated by the observation that the **quantum decoherence times can be modeled as wasted execution times due to task termination resulting from the arrival of higher-priority tasks in P-FRP.**

- We **first leverage the existing work on analyzing P-FRP programs to predict their WCRT of fault-tolerant classical computing systems** [9], [22], [33], [34]

- since the **timing analysis of re-executions for fault recovery plus transient-faults-induced wasted execution times is similar to determining the response time of tasks with terminated and incomplete executions in the P-FRP model.**

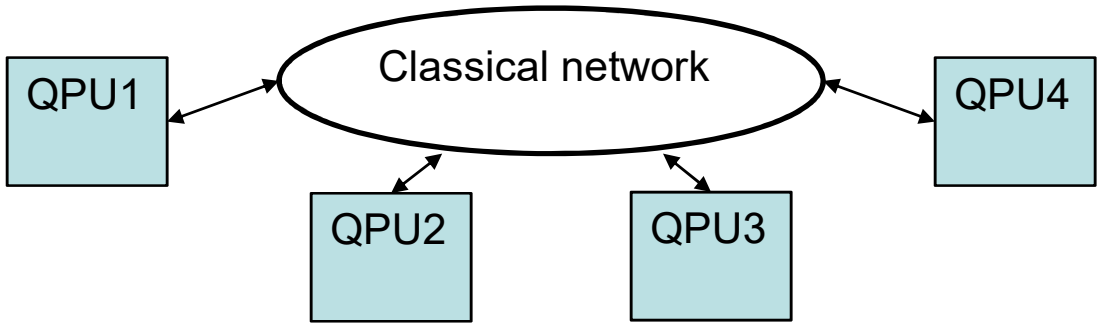
Response Time Analysis of Quantum Computing Systems

- These studies assume that faults occur at a **Poisson rate**.
- We need to determine the occurrence rate for quantum decoherence in a given quantum computer based on its characteristics.
- We then show that **accounting for wasted execution times due to errors and recovery schemes in quantum computers resulting from quantum decoherence and state fidelity can be treated as faults in non-quantum computers.**
- Ongoing work is **developing a mapping between occurrences of decoherence and arrivals of higher-priority tasks in P-FRP** so that the polynomial-time approximate timing analysis method [39] can be applied.
- **More details in:** Albert M. K. Cheng, "Response Time Analysis of Real-Time Quantum Computing Systems," 29th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS) BP, CPS-IoT Week, San Antonio, Texas USA, May 9-12, 2023.

* Supported in part by UH GEAR and Equipment Grants (Nos. 67250 and 68121).

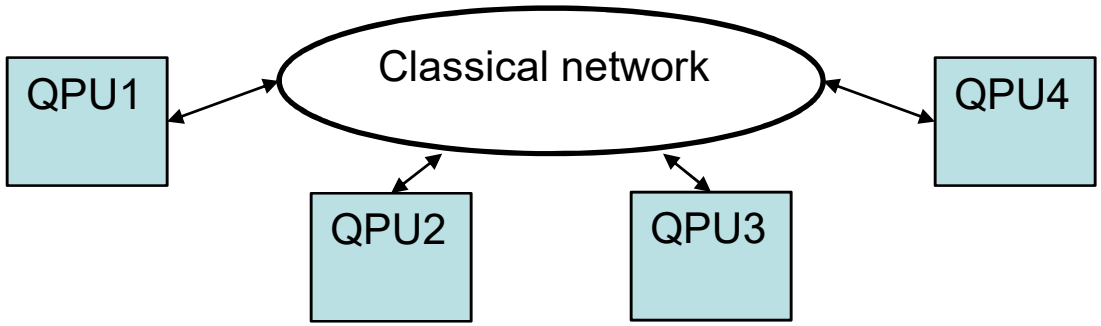
Hybrid Quantum-Classical Computing

- Current quantum computing hardware is noisy due to decoherence and can only store information for a short period of time.
- Limited to small number of qubits which are planar-connected.
- Practical applications of quantum computing require higher connectivity and far more qubits than current quantum processing units (QPUs) can provide.
- One way to make quantum computing practical in the near term is to connect the QPUs using classical computer networks which has not been systematically investigated.



Hybrid Quantum-Classical Computing

- Networked quantum processors require classical control signals to be exchanged within microseconds, aligned with qubit coherence times, to facilitate distributed operations.
- We introduce an adaptive scheduling framework that adjusts task decisions at runtime based on execution-time variations changes while maintaining provable WCRT guarantees for quantum and classical hardware.



References

[1] Amy, M., Matteo, O., Gheorghiu, V., Mosca, M., Parent, A., Schanck, J.: Estimating the cost of generic quantum pre-image attacks on SHA-2 and SHA-3. (2016)

[2] Anderson, J. ,Ramamurthy, S., Jeffay, K.: Real-time computing with Lock-free Shared Objects. ACM TOCS 5(6), pp. 388–395 (1997)

[3] Audsley, N., Burns, A., Richardson, M., Tindell, K., Wellings, A.: Applying new scheduling theory to static priority preemptive scheduling. Software Engineering Journal 8(5), pp. 284-–292 (1993)

[4] Basharin, G. P., Langville, A. N., Naumov. V. A.: The life and work of A. A. Markov. Linear Algebra Appl. 386, 0, pp. 3–26, (2004)

[5] Belwal, C., Cheng, A. M. K.: Determining Actual Response Time in P-FRP. Thirteenth International Symposium on Practical Aspects of Declarative Languages (PADL), Austin, Texas, January 24-25 (2011)

[6] 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 22 (5), pp. 563–591 (1980)

[7] Bini, E., Baruah, S. K.: Efficient Computation of Response Time Bounds under Fixed-priority Scheduling. Proc. 15th Conference on Real-Time and Network Systems, pp. 95—104 (2007)

References

[8] Cheng, A. M. K.: Real-Time Systems: Scheduling, Analysis and Verification. Wiley-Interscience (2002)

[9] Cheng, A. M. K., Dai G., Paluri, P. K., Ansari, M., Li, Y., Knape, D. B.: Fault-Tolerant Regularity-Based Real-Time Virtual Resources. 25th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), Hangzhou, China, August (2019)

[10] Davis, R. I., Burns, A.: Response Time Upper Bounds for Fixed Priority Real-Time Systems. Proc. Real-Time Systems Symposium (RTSS), pp. 407–418 (2008)

[11] DiVincenzo, D. P.: Quantum Computation. Science, 270 (5234) (1995)

[12] DiVincenzo, D. P.: The Physical Implementation of Quantum Computation. Fortschritte der Physik 48 (9–11), pp. 771–783, April (2000)

[13] Dyakonov, M. I., Luryi, S., Xu, J., Zaslavsky, A. (eds.): Is Fault-Tolerant Quantum Computation Really Possible? Future Trends in Microelectronics. Up the Nano Creek, pp. 4–18, October 14 (2006)

[14] Elliott, C., Hudak, P.: Functional reactive animation. ICFP (1997)

[15] Fahmy, S. F., Ravindran, B., Jensen, E. D.: Response time analysis of software transactional memory-based distributed real-time systems. ACM SAC Operating Systems (2009)

References

[16] Franklin, D., Chong, F. T.: Challenges in Reliable Quantum Computing. Nano, Quantum and Molecular Computing pp. 247–266 (2004)

[17] Haskell, http://www.haskell.org. Last accessed 5 Feb 2023

[18] Hietala, K., Rand, R., Hung, S.-H., Wu, X., Hicks, M.: A Verified Optimizer for Quantum Circuits. Proc. ACM Program. Lang. 5, Article 37, January (2021)

[19] IBM Quantum Update: Q System One Launch, New Collaborators, and QC Center Plans. HPCwire, January 10, (2019)

[20] Jones, N.: Computing: The quantum company. Nature. 498 (7454), pp. 286–288. (2013)

[21] Kaiabachev, R., Taha, W., Zhu, A.: E-FRP with Priorities. EMSOFT, pp. 221—230 (2007)

[22] Lin, J., Cheng, A. M. K.: Approximation Algorithms in Partitioning Real-Time Tasks with Replications. International Journal of Parallel, Emergent and Distributed Systems (IJPEDS), Volume 33, Number 2, pp. 211–232 (2018)

[23] Liu, C. L., Layland, L. W.: Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. Journal of the ACM 20(1), pp. 46—61 (1973)

[24] Louise, S.: A First Step Toward Using Quantum Computing for Lowlevel WCETs Estimations. ACM Transactions on Architecture and Code Optimization, Vol. 16, No. 3, Article 29, July (2019)

References

[25] Markov, A.: Rasprostranenie zakona bol'shih chisel na velichiny, zavisyaschie drug ot druga. Izvestiya Fizikomatematicheskogo obschestva pri Kazanskom universitete, 2-ya seriya 15, 94, pp. 135–156 (1906)

[26] Nielsen, M., Chuang, I: Quantum Computation and Quantum Information. Cambridge University Press (2000 and 2010).

[27] Pakkin, S., Coles, P.: The Problem with Quantum Computers. Scientific American, June (2019) [28] Peterson, J., Hager, G.D., Hudak, P.: A Language for Declarative Robotic Programming. ICRA. IEEE, Los Alamitos (1999)

[29] Peterson, J., Hudak, P., Reid, A., Hager, G.D.: FVision: A Declarative Language for Visual Tracking. Ramakrishnan, I.V. (ed.) PADL. LNCS, vol. 1990, pp. 304. Springer, Heidelberg (2001)

[30] Ras, J., Cheng, A. M. K.: Response Time Analysis for the Abortand-Restart Event Handlers of the Priority-Based Functional Reactive Programming (P-FRP) Paradigm. Best Paper Award Nominee. Proc. 15th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), Beijing, China, August (2009)

[31] Ras, J., Cheng, A. M. K.: Response Time Analysis of the Abort-andRestart Model under Symmetric Multiprocessing. Best Paper Award Nominee. Proc. 7th IEEE International Conference on Embedded Software and Systems (ICESS), Bradford, UK, June 29 - July 1 (2010)

References

[32] Swaine, M.: It's Time to Get Good at Functional Programming. Dr. Dobbs Journal, December (2008)

[33] Torre, E., Cheng, A. M. K.: Fault Tolerance in a Two-State Checkpointing Regularity-Based System. Proc. 41th IEEE Real-Time Systems Symposium (RTSS), December 1-4 (2020)

[34] Torre, E., Cheng, A. M. K., Dai, G., Paluri, P. K.: Two-State Checkpointing Regularity-Based System for Mixed-Criticality Tasks. Proc. 27th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), May (2021)

[35] Vepsalainen, A. P., Karamlou, A. H., Orrell, J. L., Dogra, A. S., Loer, B.: Impact of ionizing radiation on superconducting qubit coherence. Nature. 584 (7822): pp. 551–556, August (2020)

[36] Wan, Z., Hudak, P.: Functional reactive programming from first principles. ACM SIGPLAN Conference on Programming Language Design and Implementation, pp. 242—252 (2000)

[37] Wan, Z., Taha, W., Hudak, P.: Real - time FRP. ICFP (2001)

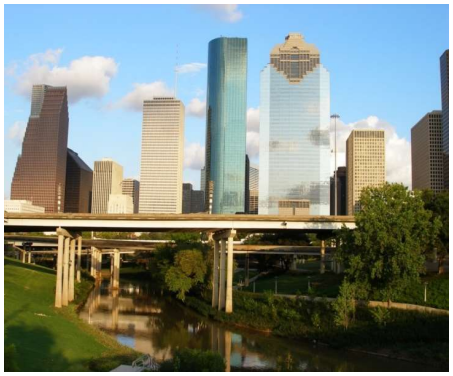
[38] Wan, Z., Taha, W., Hudak, P.: Task Driven FRP. Adsul, B., Ramakrishnan, C.R. (eds.) PADL, LNCS, vol. 2257. Springer, Heidelberg (2002) [39] Zou, X., Cheng, A. M. K., Rincon, R., Jiang, Y.: Multi-Mode PFRP Task Scheduling. Outstanding Paper Award. Proc. 20th IEEE International Symposium on Real-time Computing (ISORC), Toronto, Canada, May 16-18 (2017)

Thank you! *Albert Cheng,*

***Local Organization Chair,* RTSS 2022, Houston, Texas.**

***Invited Speaker on Quantum Computing,* RTSS 2025 OPERA, Boston, Mass.**

Prof. Albert Cheng and Ph.D. student Jiwoo Lee, who is about to present one of the three RTS team's papers at the 43rd IEEE-CS Real-Time Systems Symposium (RTSS) in Houston, Texas, USA, on December 6, 2022. This is the first in-person plus virtual conference (held at DoubleTree by Hilton Greenway Houston) since the 2018 edition.



Houston downtown.

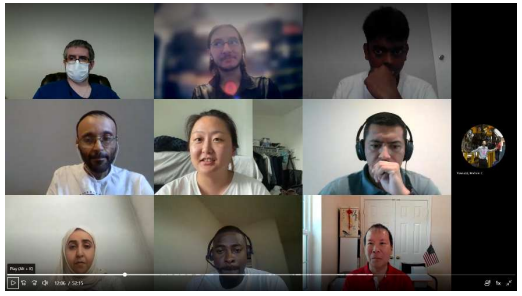
The Alamo, San Antonio.

RTSS 2022

CPS-**IoT** Week 2023

Thank you! *Albert Cheng* Real-Time Systems Group

- **Director:** Prof. Albert M. K. Cheng
- **PhD students:** Michael Yantosca, Thomas Carroll, Afrooz Abbasi, Jennifer Jiwoo Lee
- **Undergrad students:** Vinh Nguyen, Siddarth Sannabhadti, Nilesh Garg, Arham Faheem, Michael Yannuzzi, Swaroop Vedula
- **High school students:** Sarvajit Jonnalagadda, Ronit Katikaneni, Comyar Dehlavi, Ruiyang Huang
- **Scholar:** Javier Mendez (Colombian Air Force)
- **Recent graduates and their positions:** Yuanfeng Wen (MS, Microsoft, then Meta), Zeinab Kazemi (Cisco, then Microsoft), Daxiao Liu (Uber), Chaitanya Belwal (PhD, Halliburton; Visiting Assistant Prof., UHCL), Jim Ras (PhD), Jian Lin (PhD, Associate Professor, UHCL), Yu Li (PhD, Faculty, Virginia Tech), Behnaz Sanati (PhD, Mbm), Xingliang Zou (PhD, Indeed), Carlos Rincon (PhD, UH), Guangli Dai (PhD, Meta), Pavan Paluri (PhD, AMD)



Fall 2022 (9/16) group meeting

Yu Li (Best Junior PhD Student Awardee, Best PhD Student Awardee, and Friends of NSM Graduate Fellow) and Prof. Albert Cheng visit the NSF-sponsored Arecibo Observatory after their presentation at the flagship RTSS 2012 in Puerto Rico.

Real-time systems research group at Yuanfeng Wen's graduation party in May 2013. Yuanfeng is now at Facebook.

Fall 2016 (8/22) group meeting – l to r: Guillermo Rodriguez, Zhenggang Li, Tiffany Ang, Wenhui Chu, Pavani Tenneti, Carlos Rincon, Brandon Knape, Prof. Albert Cheng, Binh Doan, Nancy Lam, Yating Hou, Xingliang Zou, Elizabeth Pham. Not present: Yu Li (internship), Behnaz Sanati, Nick Troutman.