Turing Machine

- Theoretical Model Developed by Alan Turing (published May 1936)
- Abstraction of Classical Digital Computer
- Allows for Reasoning About Fundamental Computational Issues:
  - the Halting Problem
  - Computable vs. Non-computable Functions
  - Useful Model for use in Complexity Theory

Halting Problem/Decidability

Alan Turing

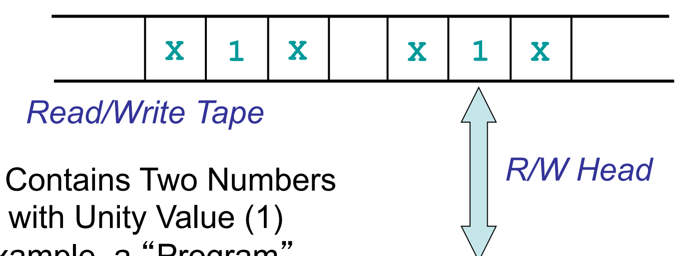
Alonzo Church

- Given a Description of a Program and Finite Input, ***Decide*** whether the Program will run forever or finishes
- One of First Problems of Decidability
- Led to Concept of Undecidability and Birth of Classical Complexity Theory
- Alonzo Church Lambda Calculus to Show the Existence of an Undecidable Problem (published April 1936)

Turing Machine Example

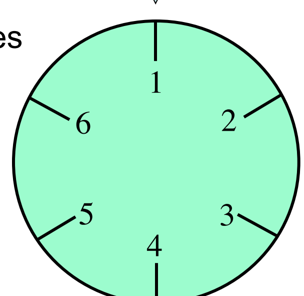
- Example Machine has 3 Parts:
  1. A Recording Tape Divided into Squares
  2. A Tape Read/Write (R/W) Head
  3. A Dial that "Chooses" Operations
- Machine can Write a Symbol **`x`** or **`1`** in Blank Square
- Machine can Erase Symbols in an Occupied Square
- A Value is Written as a Sequence of 1s, eg "4" is Written as "**`1111`**"
- Symbol **`x`** Indicates Beginning/Ending of a Number

Example Turing Machine Diagram



- Tape Contains Two Numbers Both with Unity Value (1)
- As Example, a "Program" will be Shown that Computes Sum of Two Numbers

Dial



*Berman, et al., Introduction to Quantum Computers, 1998

Programming Symbols

- Commands:
  1. `D` Write the Digit `1`
  2. `X` Write the Symbol `X`
  3. `E` Erase the Symbol
  4. `R` Move Tape One Square to Right
  5. `L` Move Tape One Square to Left
  6. `1` to `6` Change Dial Setting to this Number
- `!` Job is Completed
- `?` Mistake
- Each Command can have Number Also

*Berman, et al., Introduction to Quantum Computers, 1998

Addition Program

Current Dial Setting Column

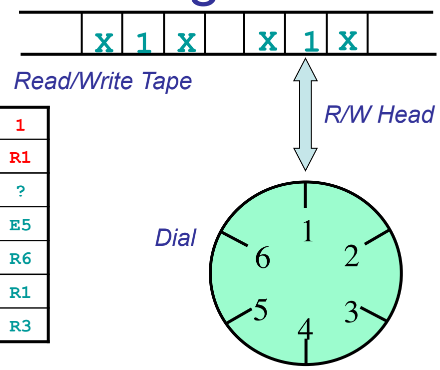
Instructions Based on Dial Setting and Current Tape Value

Tape Content (before instruc.)

	(*blank*)	X	1
1	D6	E2	R1
2	R2	E3	?
3	R3	E4	E5
4	L4	?	R6
5	L5	?	R1
6	X6	!	R3

*Berman, et al., Introduction to Quantum Computers, 1998

Addition Program Example

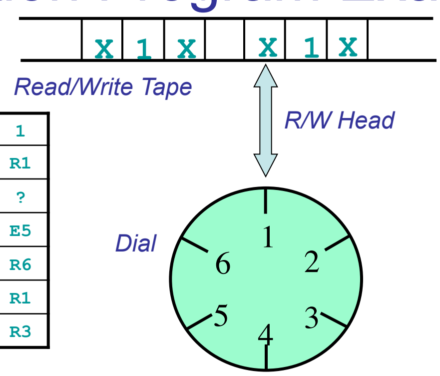


	(*blank*)	X	1
1	D6	E2	R1
2	R2	E3	?
3	R3	E4	E5
4	L4	?	R6
5	L5	?	R1
6	X6	!	R3

- R/W Head Reads a **1**
- Dial Setting is **1** → Command is **R1**
- Tape Head Moved One Position to Right
- Dial is Set to **1**

*Berman, et al., Introduction to Quantum Computers, 1998

Addition Program Example

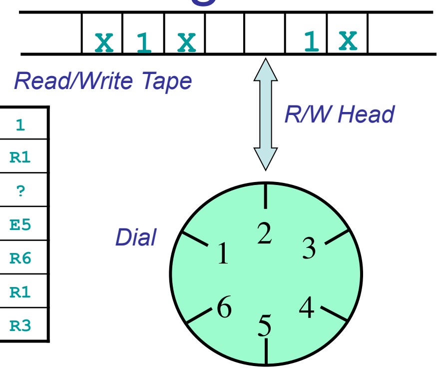


	(*blank*)	X	1
1	D6	E2	R1
2	R2	E3	?
3	R3	E4	E5
4	L4	?	R6
5	L5	?	R1
6	X6	!	R3

- R/W Head Reads a **X**
- Dial Setting is **1** → Command is **E2**
- Tape Square is Erased (set to ***blank***)
- Dial is Set to **2**

*Berman, et al., Introduction to Quantum Computers, 1998

Addition Program Example

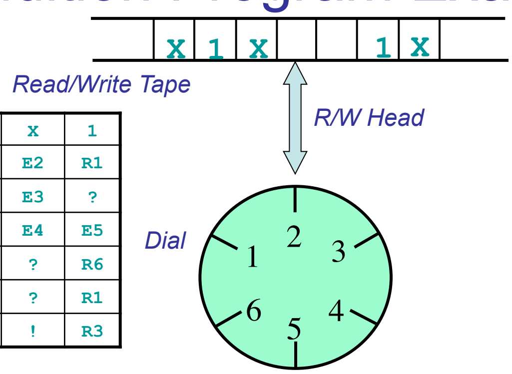


	(blank)	x	1
1	D6	E2	R1
2	R2	E3	?
3	R3	E4	E5
4	L4	?	R6
5	L5	?	R1
6	X6	!	R3

- R/W Head Reads a **blank**
- Dial Setting is **2** → Command is **R2**
- Tape Head Moved One Position to Right
- Dial is Set to **2**

*Berman, et al., Introduction to Quantum Computers, 1998

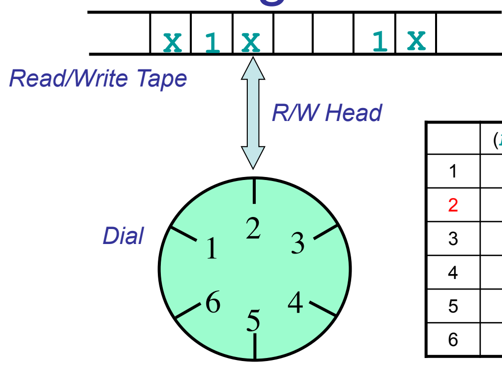
Addition Program Example



	(blank)	x	1
1	D6	E2	R1
2	R2	E3	?
3	R3	E4	E5
4	L4	?	R6
5	L5	?	R1
6	X6	!	R3

- R/W Head Reads a **blank**
- Dial Setting is **2** → Command is **R2**
- Tape Head Moved One Position to Right
- Dial is Set to **2**

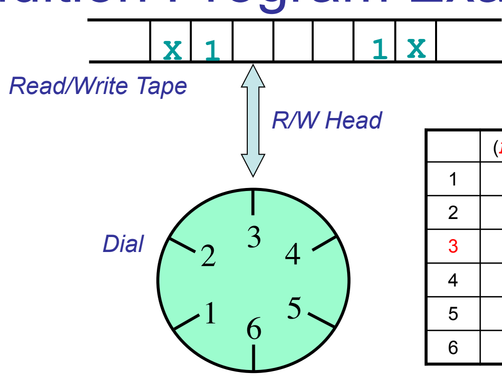
Addition Program Example



	(*blank*)	X	1
1	D6	E2	R1
2	R2	E3	?
3	R3	E4	E5
4	L4	?	R6
5	L5	?	R1
6	X6	!	R3

- R/W Head Reads a **X**
- Dial Setting is **2** → Command is **E3**
- Tape Square is Erased (set to ***blank***)
- Dial is Set to **3**

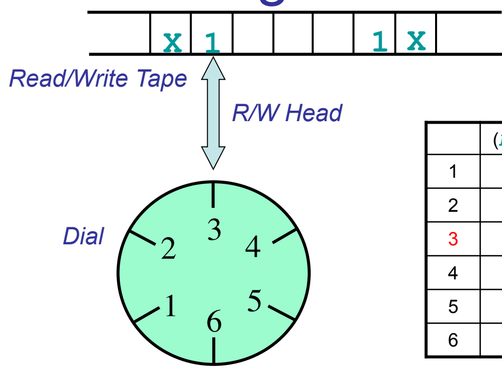
Addition Program Example



	(*blank*)	X	1
1	D6	E2	R1
2	R2	E3	?
3	R3	E4	E5
4	L4	?	R6
5	L5	?	R1
6	X6	!	R3

- R/W Head Reads a ***blank***
- Dial Setting is **3** → Command is **R3**
- Tape Head Moved One Position to Right
- Dial is Set to **3**

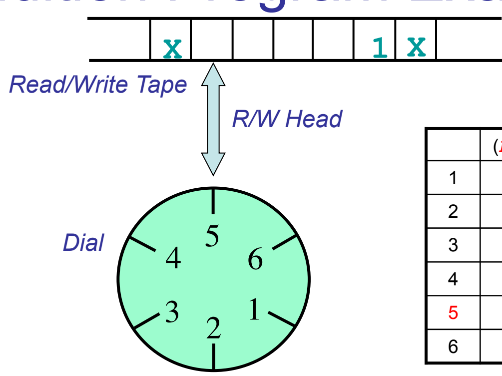
Addition Program Example



	(blank)	X	1
1	D6	E2	R1
2	R2	E3	?
3	R3	E4	E5
4	L4	?	R6
5	L5	?	R1
6	X6	!	R3

- R/W Head Reads a `1`
- Dial Setting is `3` → Command is `E3`
- Tape Square is Erased (set to *`blank`*)
- Dial is Set to `5`

Addition Program Example



	(blank)	X	1
1	D6	E2	R1
2	R2	E3	?
3	R3	E4	E5
4	L4	?	R6
5	L5	?	R1
6	X6	!	R3

- R/W Head Reads a *`blank`*
- Dial Setting is `5` → Command is `L5`
- Tape Head Moved One Position to Left
- Dial is Set to `5`

Summary of Example Program

Tape Content							Op
X	1	X		X	1	X	R1
X	1	X		X	1	X	E2
X	1	X		_	1	X	R2
X	1	X	_		1	X	R2
X	1	X			1	X	E3
X	1	_			1	X	R3
X	1				1	X	E5
X	_				1	X	L5
X		_			1	X	L5
X			_		1	X	L5
X				_	1	X	L5

Summary of Example Program (cont)

Tape Content							Op
X					1	X	R1
X				_	1	X	D6
X				1	1	X	R3
X			_	1	1	X	R3
X		_		1	1	X	R3
X	_			1	1	X	R3
X				1	1	X	E4
_				1	1	X	L4
	_			1	1	X	L4
		_		1	1	X	L4
			_	1	1	X	L4

Summary of Example Program (cont)

Tape Content							Op
				<u>1</u>	1	x	R6
			_	1	1	x	X6
			<u>x</u>	1	1	x	!

Result is:
1+1=2

- TM is Abstraction of Classical Computer
- Generalized (Programmable) Finite Automaton
- Halting Problem is Given Turing Machine and this Program, will it Terminate?

TM Analogy to Digital Computer

- Tape (Data Mem.) and Dial (Instr. Mem.) are Memory Elements
- Writing/Erasing Elements are ALU
- Programming “Actions” are the Control Unit (in Dial Also)
- Initial Tape Content is Input Data
- Tape Content after Halt is Output Data
- Sequence of Operations Represent Clocking or Sequential Behavior

TM Analogy to Digital Computer

- A Given Dial Setting Represents Inst. Pointer Register
- For Given Dial Setting, Choice Among next Operation Based on Current Tape Value Represents Decision Construct (i.e. if-then-else)
- Any Problem that can be Solved as a Program on a TM is "Turing Computable"
- All Classical Computers can Execute any Turing Computable Program Regardless of Underlying Hardware/Software Implementation

TM as a Finite Automaton

- Finite Automaton is Mathematical Term for a Finite State Machine
- in TM, the State is the Combination of the Current Tape Symbol and the Dial Setting
- The Program Defines the State Transitions and the Machine Operation
- A Program has a Halt State (hopefully)
- Example Halt State is Dial Setting, (Tape Symbol)=(6,**x**) Causing a Halt Operation **!**
- Note that Digital Logic Control Circuits Typically have no Halt State

Probabilistic Turing Machine (PTM)

- Some Transitions are Random Choices Among the Possibilities
- In Our Example, a PTM would be Modeled as a Dial Setting where the Three Operations are Chosen Randomly Instead of Deterministically Based on Tape Content
- Stochastic Performance:
  - Different Results Among Consecutive Runs with Same Program and Input Data
  - Different Runtimes, Possible for Halt State to Never be Reached

Non-deterministic Turing Machine

- Similar to PTM - More than One Next State Per Computational Step
- Probability Distribution of all Allowed States is Known
- "Guesses" the Correct Answer at Each Step
- Mathematical Model of Computation
- Non-deterministic Turing Machine not Realized
- PTM Could be Built - but Probably not very useful

Computational Complexity Theory

- Measure of Computer Runtime (time) and Memory Usage (space) for Particular Computable Function in Terms of Elementary Operations
- Elementary Operation Examples:
  - Number of Vertices of a Decision Diagram
  - Number of Clock Cycles per Digit for a Square Root Algorithm
  - Number of Clock Cycles per Vertex to Traverse a Graph
  - Number of Swaps in a Bubble Sort Algorithm

Complexity Classes

- Theoretical Computer Scientists have Defined Several Different Classes of Decidable Problems
- Class P: Those Problems Requiring Total Time or Space Resources Expressible as a Polynomial in Terms of Elementary Operations on a Turing Machine (eg. Bubble Sort)
- Class *NP*: Those Problems Requiring Total Time or Space Resources Expressible as a Polynomial in Terms of Elementary Operations on a Non-determinstic Turing Machine (*NP* does NOT stand for "not polynomial") (eg. Graph Isomorphism)
- Class Co-*NP*: The Complement of the Problems in *NP* where the yes/no Answer is Reversed
- Class *NP*-Complete: A subset of the problems in Class *NP* such that if Any One Could be solved in *P*, all Others could be REDUCED to This One and Also Solved in *P* (eg. SAT, set covering, traveling salesman)

Does P equal NP? Prove/disprove it and you will receive $1,000,000 from the Clay Institute!

Bubble Sort Example

- Given a List of n unsorted Numbers, place them in Ascending Order
- Elementary Operation is the Adjacent Number SWAP
- Begin at First Number in List and Proceed Forward in list Item by Item
- If Current Value > Next Item Perform a SWAP
- Repeat Traversal Through List Until Sorted
- First Traversal Through List Guarantees Largest Value Moved to End
- Second Traversal Guarantees Next Largest Moved to Second from End, etc.
- Spatial Worst-case Complexity: $O(n) = n$
- Temporal Worst-case Complexity: $O(n) = n^2$
- Temporal Best-case Complexity: $\Omega(n) = n$
- Theoretical Worst-case Complexity for the Sorting Problem: $O(n) = n \log_2 n$

Tractable and Intractable Problems

- Informally, a Problem Solvable in Polynomial Time is Termed a *Tractable* Problem
- Likewise, Problems Requiring More than Polynomial Time are *Intractable*
- *Intractable* Problem Example is One that Requires 2^n (exponential) Amount of time
- Some <u>*Intractable* Problems on TMs</u> are <u>*Tractable* on a Quantum Computer</u> as Defined by Deutsch!!!!
- These Problems are in Complexity Class QP
- Complexity Class QP: The Set of Problems Solvable in Polynomial Time on a Quantum Computer for Which the Best Known TM Algorithm is Intractable

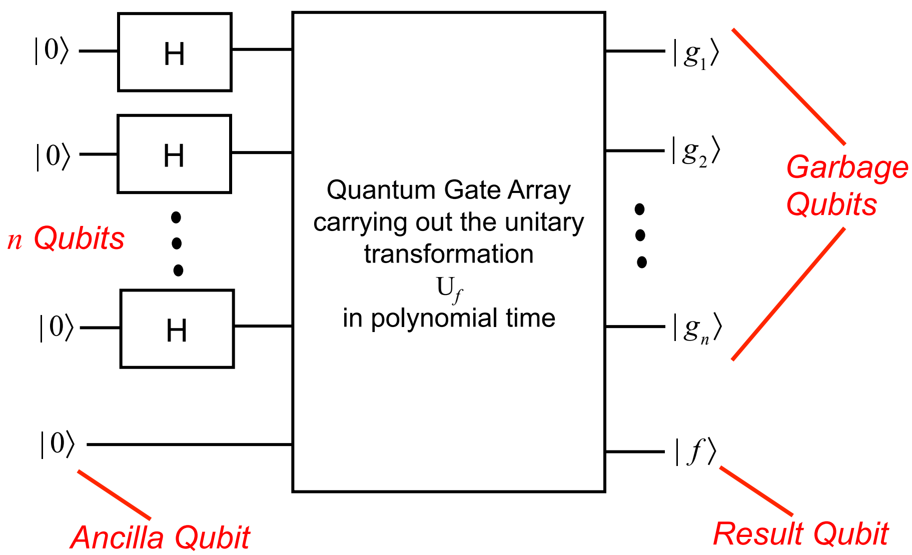
Quantum Parallelism

- This Notion may Explain why the Class QP Exists
- Qubits can Exist in a Superposition of Basis States
- This Allows Parallelism at the Information Level
- Classical Computer Parallelism Exists
  - Temporal Level (aka Pipelining)
  - Multiple CPUs

Quantum Parallelism Example

- Assume we Wish to Compute $f(j)$ in Polynomial Time
- Where j is a Binary String of Length n
- Need Single Copy of Quantum Circuit that Evaluates f
- Can Compute Superposition of all 2^n Values of f
- Polynomial Time with Single Copy of Quantum Circuit
- Classical Computer Requires:
  1. 2^n Runs on Single Processor
  2. 2^n Parallel Processors

Example Quantum Computation



First Stage in Example

- Initialize n Qubits to $|0\rangle$
- Perform Hadamard Transform on n Qubits

$$\mathbf{H} \otimes \mathbf{H} \otimes \cdots \otimes \mathbf{H}(|0\rangle|0\rangle\cdots|0\rangle) = \frac{1}{2^{n/2}}(|0\rangle + |1\rangle)^n = \frac{1}{2^{n/2}}\sum_{k=0}^{2^n-1}|k\rangle$$

$$\mathbf{H} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Second Stage in Example

- Quantum Gate Array Implements Function f

$$f : \{0,1\}^n \rightarrow \{0,1\}$$

- Maps n-bit Input String $j=(j_0,j_1,\ldots,j_{n-1})$ to 0 or 1 in Polynomial Time
- Quantum Gate Array Transformation:

$$\mathbf{U}_f : |j_0\rangle |j_1\rangle \ldots |j_{n-1}\rangle |0\rangle$$
$$\mapsto |j_0\rangle |j_1\rangle \ldots |j_{n-1}\rangle |f(j_0, j_1, \ldots, j_{n-1})\rangle$$

Result in Example

- Input to Function Evaluation is in Superposition State Created by n Hadamard Gates
- Result is Superposition of all Values of Function f

$$|f\rangle = \sum_{j=0}^{2^n-1} |j\rangle |f(j)\rangle$$

Quantum Algorithm Classes

- 3 Broad Catagories (Shor' 03)
  1. Finding Periodicity of a Function (using Fourier transforms)
     - Shor' s Algorithms-factoring/discrete log
     - Hallgren' s Algorithm-solve' s Bell' s equation
  2. Search N items in $N^{1/2}$ time
     - Grover' s Algorithm
  3. Simulation of Quantum Systems
     - Potentially large class suggested by Feynman
     - Not many of these Presently developed

Quantum Algorithm Development

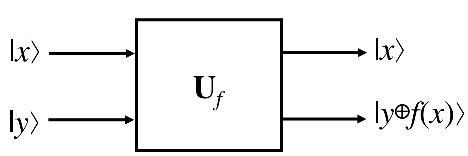
- QAs Offer Substantial Speedup over Classical but Limited in Applicability
- Concentrate on Problems NOT in Class P
- Common Conjecture is QAs do NOT Solve NP Problems in Polynomial Time
- If Conjecture is True, Then Class of Problems Applicable for QA Speedup is Neither NP-hard nor P
- Remaining Population of Problems is Relatively Small

Parallelism

- Classical Computation
  - Single Copy of Circuit/Algorithm Requires to Compute Function 2^n Times
  - 2^n Copies of Circuit/Algorithm Allows to Obtain all Values of Function in Single Time Step
- Quantum Circuit/Algorithm
  - Requires Single Copy of Circuit /Algorithm
  - Obtain all Values in Single Time Step
- Parallelism is at Information Level
  - More than Superposition
  - Entanglement also Plays a Role
- Power Requirements Differ Exponentially

Superposition Example

- Consider Qubit $|x\rangle$ with Two Possible Values $|0\rangle$ or $|1\rangle$
- Consider a Function $f(x)$ With Two Possible Values $|f(x)\rangle = |0\rangle$ or $|f(x)\rangle = |1\rangle$
- We Wish to Construct a Circuit Whose Output is a Superposition of $|f(0)\rangle$ and $|f(1)\rangle$
- Possible to Construct Circuit to Transform 2 Input Qubits $|x\rangle$ and $|y\rangle$ into Results $|x\rangle$ and $|y \oplus f(x)\rangle$ Using only Fredkin Gates (Marinescu, p.189)
- Function $f(x)$ is “Hardwired” into Circuit/Algorithm



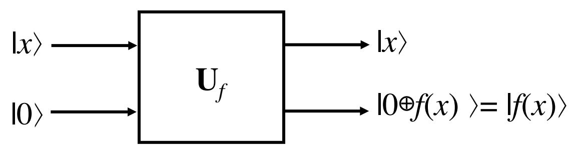
Superposition Example (cont)

- Setting $|y\rangle = |0\rangle$ Results in the Following Transformation:

$$|x\,0\rangle \mapsto |x\ f(x)\rangle$$

- Abbreviated Notation Implies Tensor Product:

$$|x\rangle|0\rangle \mapsto |x\rangle|f(x)\rangle \qquad |x\rangle \otimes |0\rangle \mapsto |x\rangle \otimes |f(x)\rangle$$



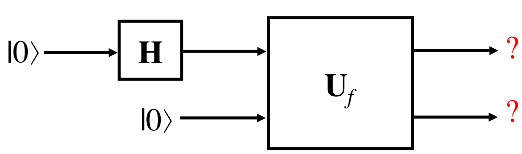
Superposition Example (cont)

- Instead of $|x\rangle$, Consider Replacing it With Qubit $|0\rangle$ Applied to a Hadamard Gate Initially:

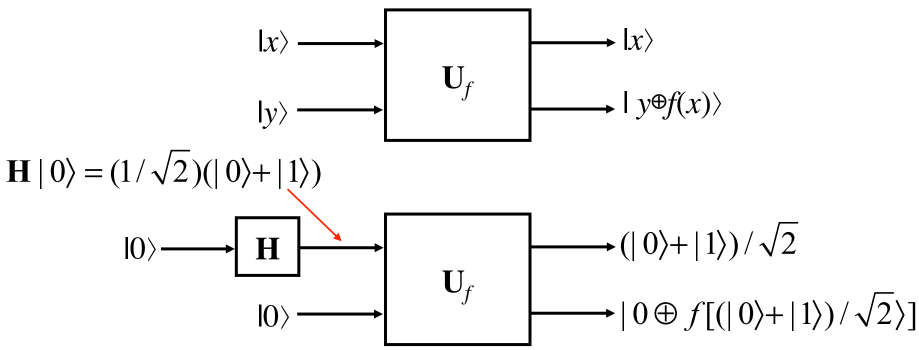
$$\mathbf{H}|0\rangle = (1/\sqrt{2})(|0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 0| - |1\rangle\langle 1|)|0\rangle$$

$$\mathbf{H}|0\rangle = (1/\sqrt{2})(|0\rangle\langle 0|0\rangle + |0\rangle\langle 1|0\rangle + |1\rangle\langle 0|0\rangle - |1\rangle\langle 1|0\rangle)$$

- Results in the Following Superposition State Applied to $\mathbf{U}_f$: $\quad \mathbf{H}|0\rangle = (1/\sqrt{2})(|0\rangle + |1\rangle)$



Superposition Example (cont)



$$|\, f[(|\,0\rangle + |\,1\rangle)/\sqrt{2}]\rangle = \underline{\underline{[|\, f(|\,0\rangle) + f(|\,1\rangle)\rangle]/\sqrt{2}}}$$

Superposition Example (cont)

- Output State of Quantum System is Tensor Product of the Two Output Vectors:

  $$[|\,0 f(|\,0\rangle)\rangle + |\,0 f(|\,1\rangle)\rangle + |\,1 f(|\,0\rangle)\rangle + |\,1 f(|\,1\rangle)\rangle]/\sqrt{2}$$

- Output State Contains Information About Both Possible Evaluations $|f(0)\rangle$, $|f(1)\rangle$ in One Run/ Execution Period – ***Quantum Parallelism***
- Measurement will Yield one of the Basis States (Eigenvectors) of the Observable
- Complete Algorithm/Circuit Must Allow for "Cancelling Out" Undesirable Results
- Can be Applied to Systems of m Qubits Yielding 2^m Superimposed Results!