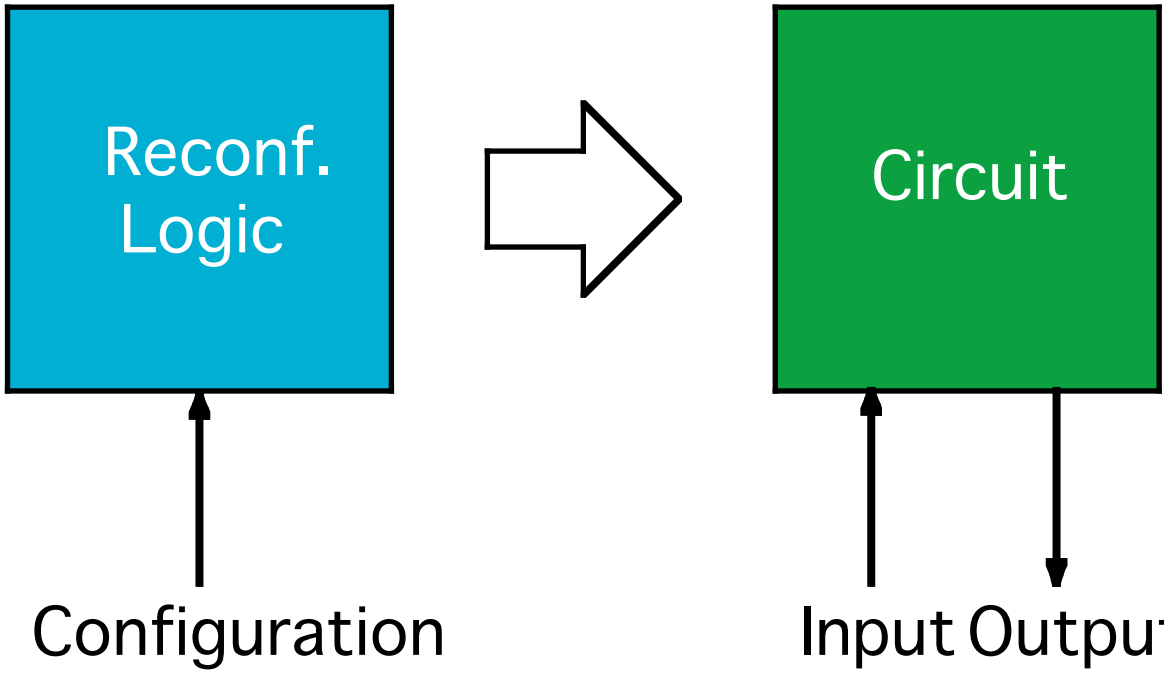
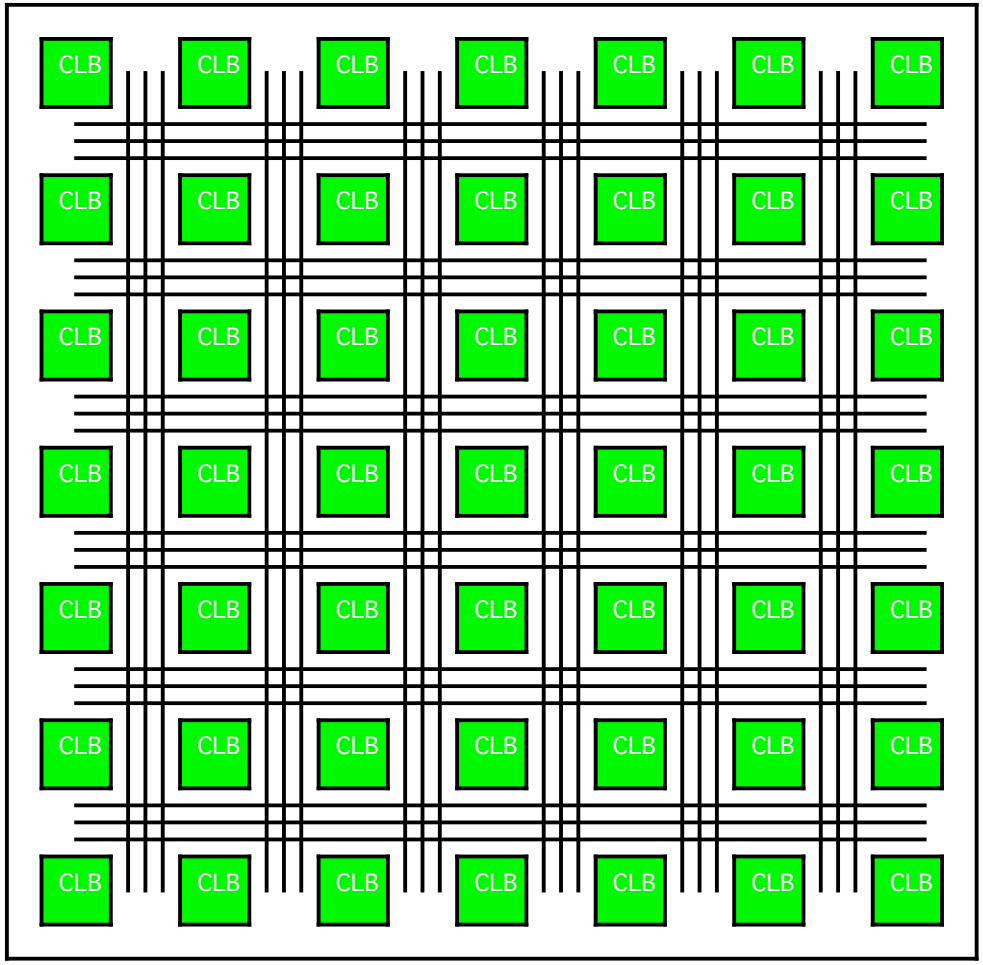
ECE 497NC:
Unconventional Computer Architecture

Lecture 6:
Reconfigurable Computing 1:
Motivation and Concepts

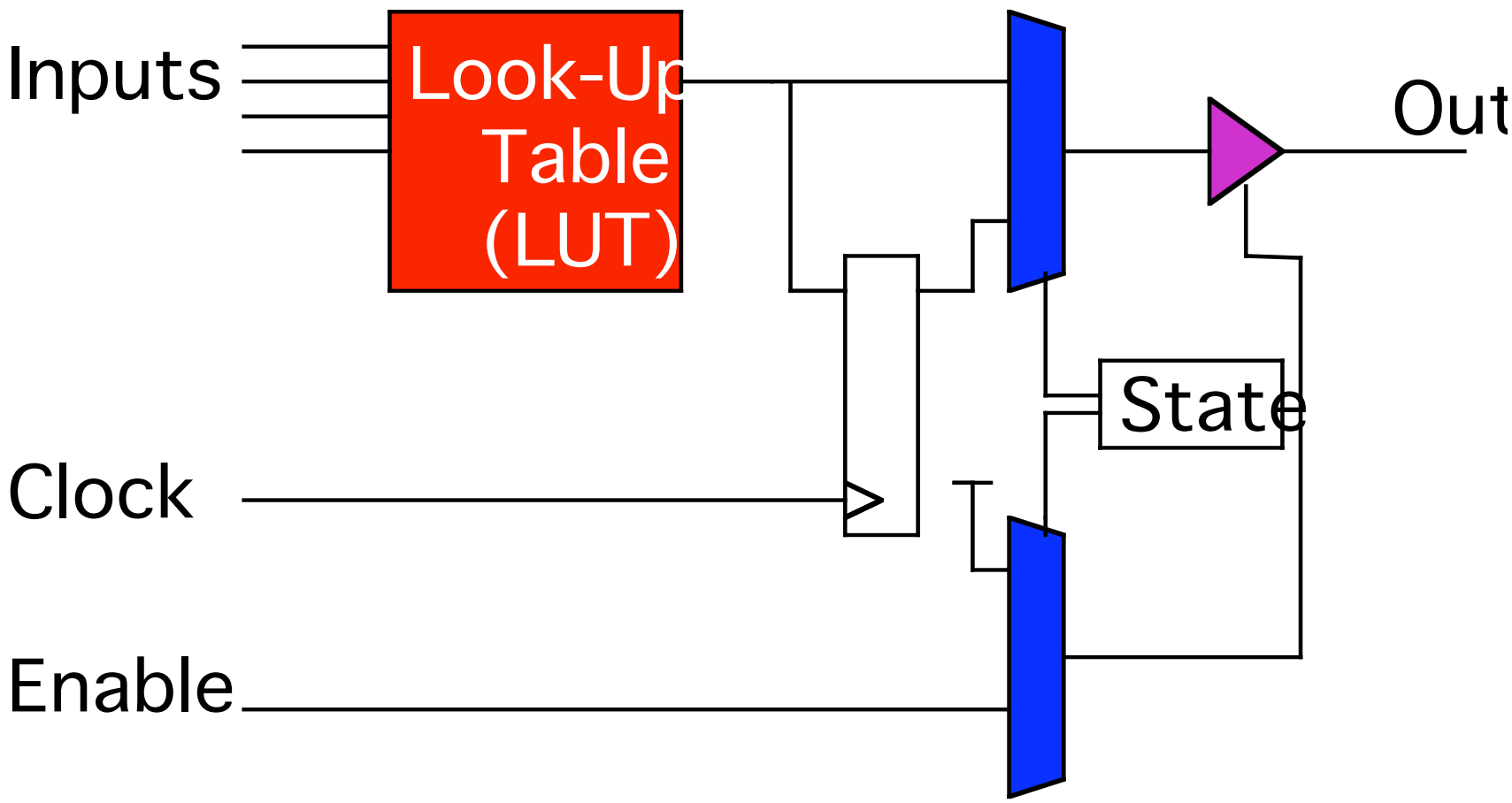
Reconfigurable Logic



Field-Programmable Gate Arrays



Combinational Logic Blocks



FPGA Design Parameters

- # of inputs to LUT
  - Trade-off number of CLBs required against size of CLB and routing area
- How is logic implemented
  - LUT vs. programmable and-or-invert vs. other
  - Technology used to hold state (program) of CLB
- Flip-flop in CLB?
- Additional Functionality
  - Carry chains

Results of Studies

- Wiring dominates area
  - Size of CLB less critical
    - Would seem to argue for more complex CLBs
  - # of inputs to CLB has strong impact on area required
- Latches seem to be a big win
- 4-Input CLBs best for area
- 5-6 Input CLBs best for speed

Note: analysis 10 years old – assumptions may be incorrect

Why Would you Want to Compute With FPGAs?

- Huge performance gap between software and hand-designed hardware systems
  - Often 100-to-1 ratio of performance or performance/area
- Hardware systems not so good for general computing
  - Big design, cost barriers to implementation
  - Not practical to buy a new machine every time you want to run a different program
- Reconfigurable systems offer best-of-both-worlds
  - Run-time programmability
  - Hardware-level performance

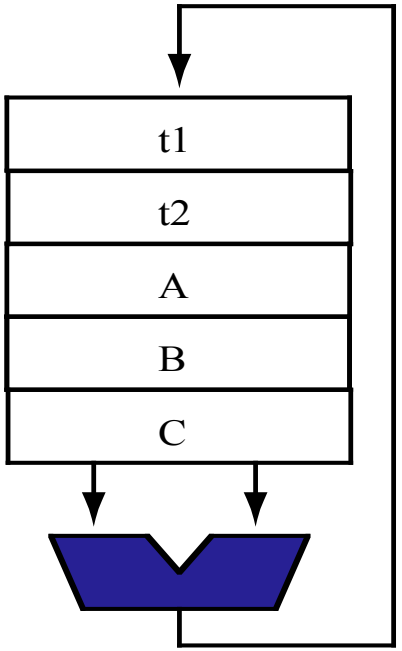
Why is Custom Logic Faster Than Software?

- Spatial vs. Temporal Computation
  - Processors divide computation across time, dedicated logic divides across space

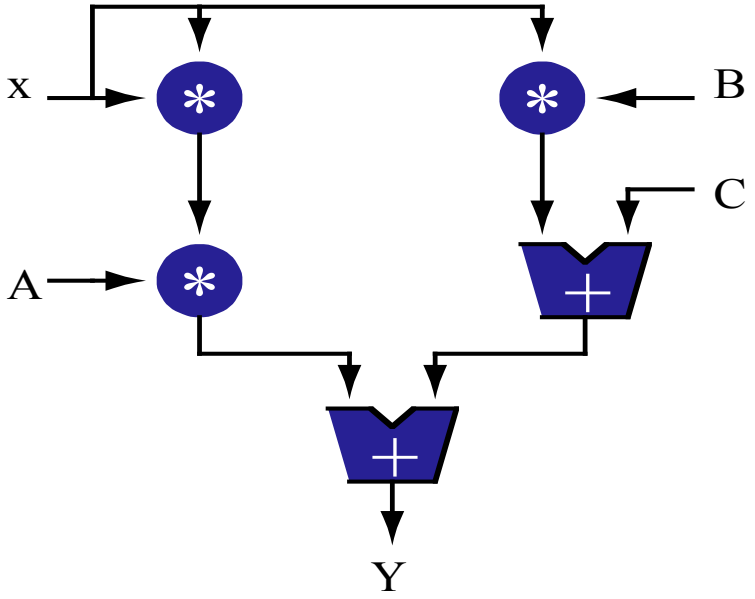
$$y = Ax^2 + Bx + C$$

Temporal Computation

```
t1 = x
t2 = t1 * A
t2 = t2 + B
t2 = t2 * t1
y  = t2 + C
```



Spatial Computation



Why is Custom Logic Faster Than Software?

- Specialization
  - Instruction set may not provide the operations your program needs
  - Processors provide hardware that may not be useful in every program or in every cycle of a given program
    - Multipliers
    - Dividers
- Instruction Memory
  - Processors need lots of memory to hold the instructions that make up a program and to hold intermediate results.
- Bit Width Mismatches
  - In general, processors have a fixed bit width, and all computations are performed on that many bits
    - Multimedia vector instructions (MMX) a response to this

Good Applications for Reconfigurable Computing

- Relatively small application graph
  - FPGAs have limited capacity
  - Simple control flow helps a lot
- Data Parallelism
  - Execute same computations on many independent data elements
  - Pipeline computations through the hardware
- Small and/or varying bit widths
  - Take advantage of the ability to customize the size of operators

Reconfigurable Computing Successes

- RSA Decryption
  - Programmable-Active-Memory machine set record for decryption of RSA-encrypted data
- DNA Sequence Matching
  - Reconfigurable hardware has achieved 100x better performance than contemporary supercomputers
- Signal Processing
  - FPGA-based filters often get 10x better performance than DSP chips
  - Benefit from customization of hardware to the application
- Emulation
  - Use reconfigurable logic to simulate new processors at high speeds
- Cryptographic Attacks
  - High-performance low-cost implementations for breaking encryption algorithms