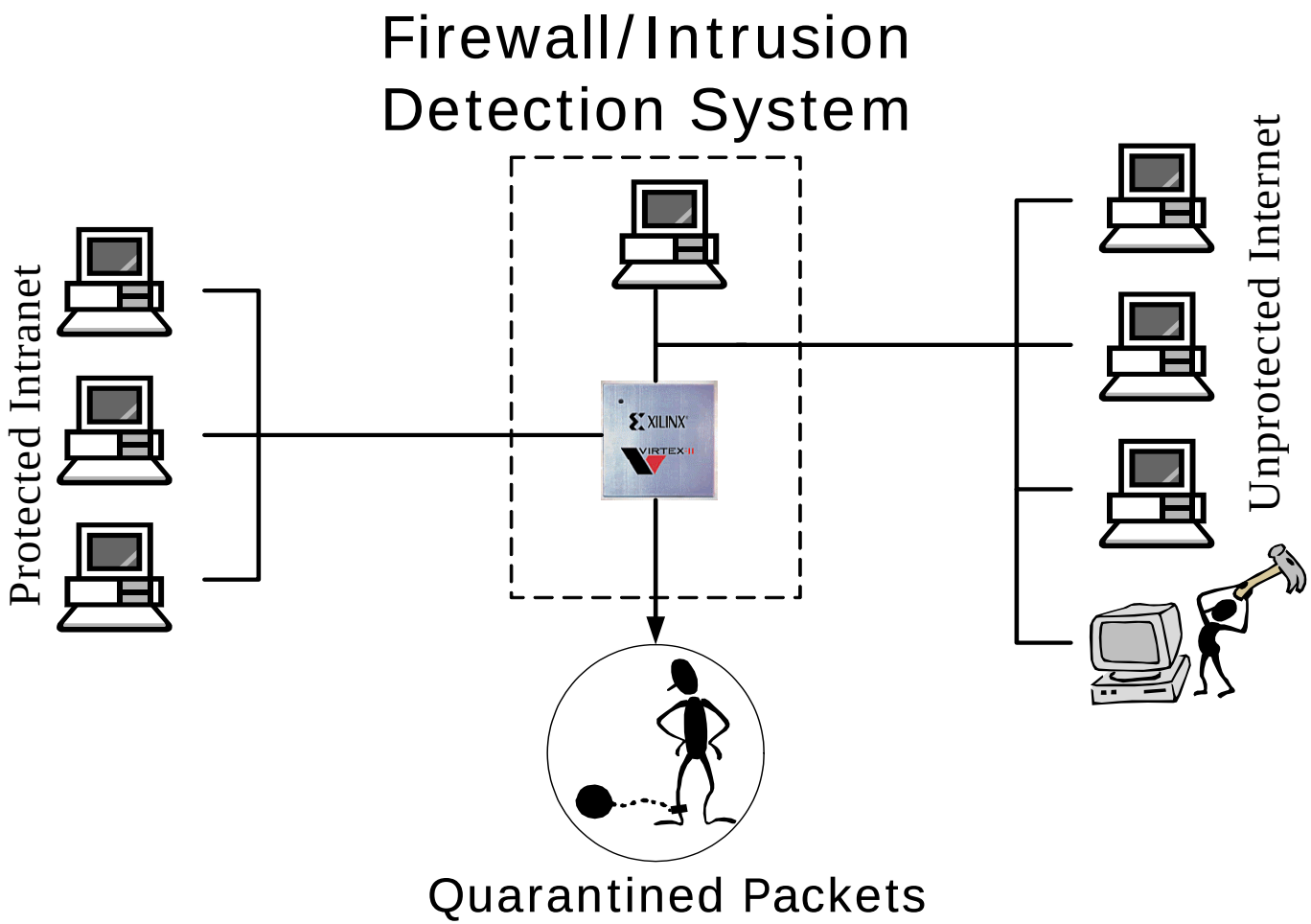
Introduction to Intrusion Detection Systems



What is Intrusion Detection?

- All incoming packets are filtered for specific characteristics or content
- Databases have thousands of patterns requiring string matching
  - FPGA allows fine-grained parallelism and computational reuse
- 10 Gb/s and higher rates desired
  - Provided by pipelined, streaming architectures

Other Approaches

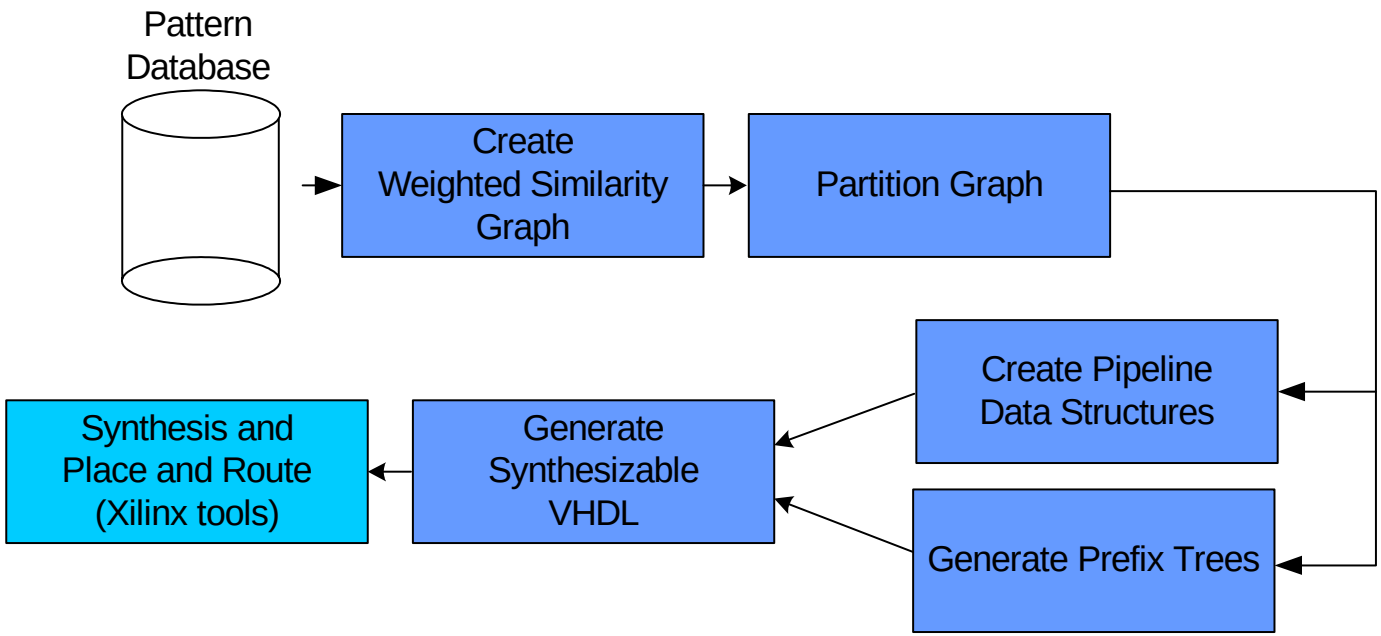
- Objective: find all occurrences of a pattern in an input
- Naïve approach: O((n-m+1)m)
- Shift-and-compare: O(n), large hardware requirements, O(nm) work
- Hashing: O(n), hashing can be complex, O(nm) work
- KMP: O(n): other algorithms may be faster in practice, but do not provide low precise upper bound (2n – m), O(n+m) work

High-Performance Shift-and-Compare Architectures

Various contributions to shift-and-compare architectures:

- Pre-decoded architecture provides significant area and routing improvements over encoded data
- Graph-based partitioning of patterns allows for reduced routing complexity and increased frequency performance through multiple pipelines
    - Average of 15% decrease in area, 5% decrease in clock period over unpartitioned unary

Methodology Flow

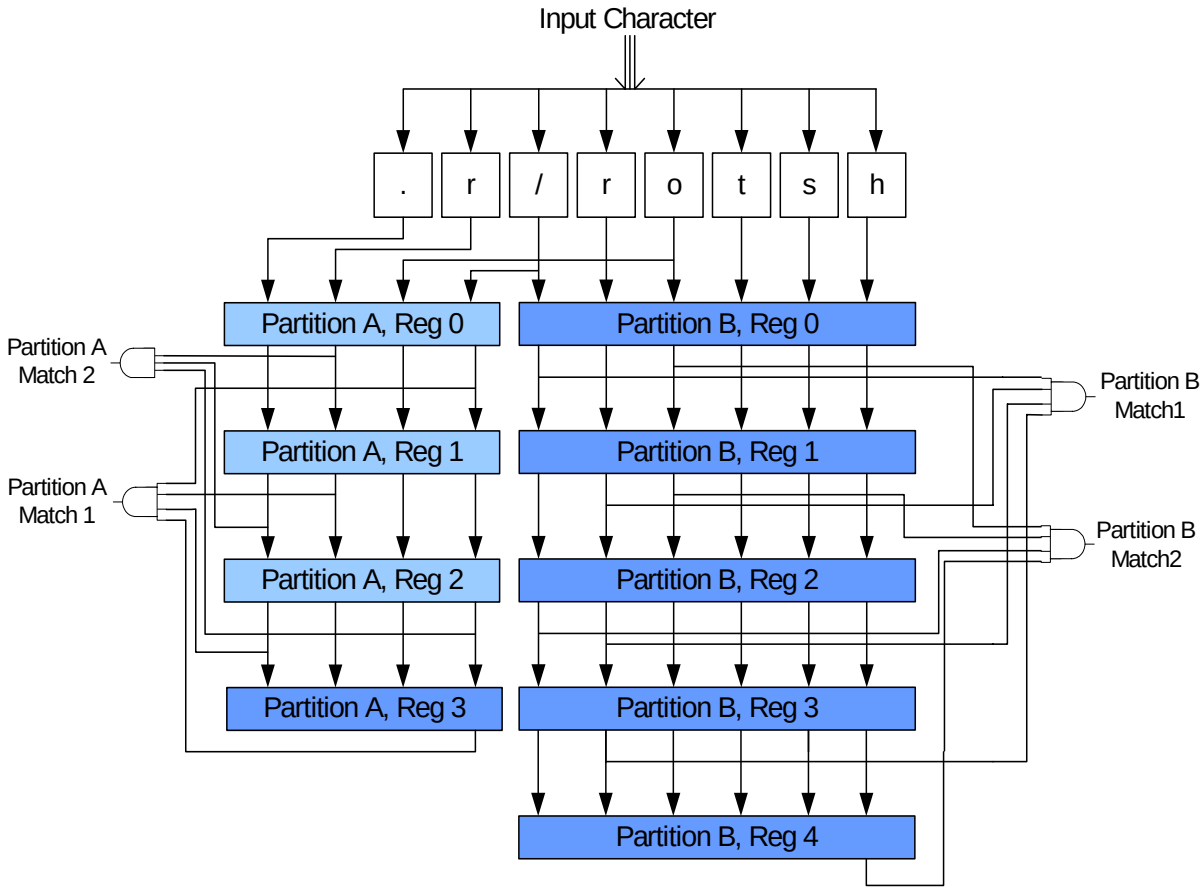


Reduction of Resource Usage

- Trie-based prefix grouping allows for reduced area consumption through lower redundant comparisons
  - 4-byte prefixes turn out to be very appropriate for intrusion detection:

    /cgi-bin/bigconf.cgi
    /cgi-bin/common/listrec.pl
    /cgi-sys/addalink.cgi
    /cgi-sys/entropysearch.cgi

- Replication of hardware and delays allow for multi-byte per cycle throughput at high clock rates
  - Pipeline is not increased in size – large source of slice consumption
  - Front end decoders increases in size by *k*
  - Back end matchers increase in size by *k*



	1 way	4 way	8 way
Number of Slices	299	721	1338
Clock Period	4.2ns	4.6ns	5.3ns
Throughput	1.9Gb/s	6.9Gb/s	12.1Gb/s
Efficiency	1	1.51	1.41

*Efficiency in throughput/area, normalized to 1-way (~100 rules)

Customized Performance

- Variations in tool flow provide customizable performance:
  - Tool Options
    - Small: partitioned and pre-decoded architecture
      - Prefix trees
    - Fast: *k*-way architecture
    - Fast reconfiguration, minimum complexity
      - KMP architecture

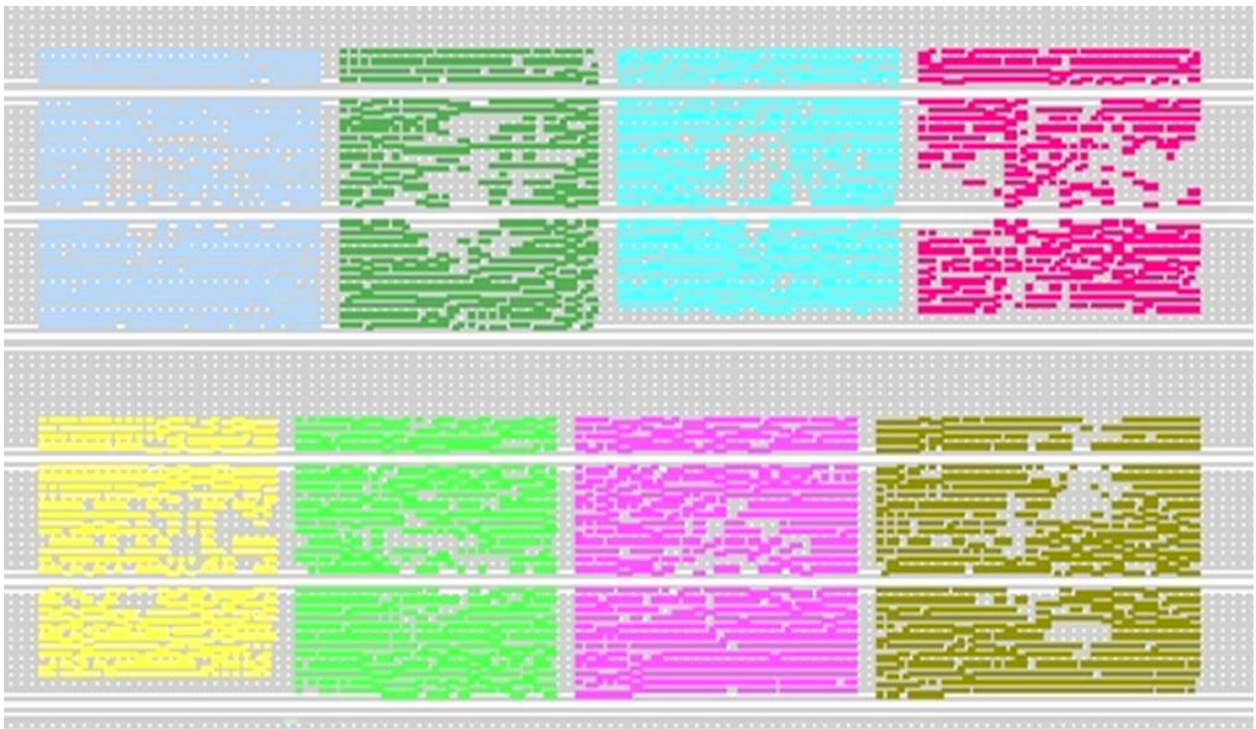
Comparison of Related Architectures

Design	Throughput	Unit Size	Performance
USC Unary	2.1 Gb/s	7.3	283
USC Unary (1 byte)	1.8 Gb/s	5.7	315
USC Unary (4 byte)	6.1 Gb/s	22.3	271
USC Unary (8 byte)	10.3 Gb/s	32	322
USC Unary (Prefilter)	6.4 Gb/s	9.4	682
USC Unary (Tree)	2.0 Gb/s	6.6	303
Los Alamos (FPL '03)	2.2 Gb/s	243	9.1
UCLA (FPL '02)	2.9 Gb/s	160	18
UCLA w/Reuse (FCCM '04)	3.2 Gb/s	11.4	280
U/Crete (FPL '03)	10.8 Gb/s	269	40.1
U/Crete (FCCM '04)	9.7 Gb/s	57	170
GATech (FCCM '04)	7.0 Gb/s	50	140

* Throughput is assumed to be constant over variations in pattern size. Unit size is the average unit size for a 16 character pattern (in logic cells; one slice is two logic cells), and performance is defined as Mb/s/cell).

Incremental Architecture Synthesis

- Goal: Reduce place and route costs
- Cost for changing rules in one of *k* partitions:
  overhead + 1/k
- Key: Predefinition of area constraints



Determining the Optimal Partition

$$\delta_i = \left(S_{p^*} \setminus P_i\right)$$

$$\text{find j such that } \left|\delta_j\right| = \min_{i=0}^{P} \left|\delta_i\right|$$

$$\text{characters to add to partition j are in } \delta_j$$

Definitions:

- S_{p^*} the set of characters required to represent the new pattern p*.
- The set difference between the characters currently represented in P_i and the characters that are present in S_{p^*} is δ_j
- The partition which will require the addition of the minimum number of new characters is the optimal partition P_j.
- The optimal partition is selected from the set of partitions P.

Relevant Publications

"*Time and Area Efficient Pattern Matching on FPGAs*," Proceedings of the 12th Annual ACM International Symposium on Field-Programmable Gate Arrays (FPGA '04)

"*A Methodology for the Synthesis of Efficient Intrusion Detection Systems on FPGAs*," Proceedings of the Twelfth Annual IEEE Symposium on Field Programmable Custom Computing Machines 2004 (FCCM '04)

"*Automated Incremental Design of Flexible Intrusion Detection Systems on FPGAs*," Proceedings of the Eighth Annual Workshop on High Performance Embedded Computing (HPEC '04)

Additional publications: http://ceng.usc.edu/~prasanna