Compiling PCRE to FPGA for Accelerating SNORT IDS

Abhishek Mitra

Walid Najjar

Laxmi N Bhuyan

University of California Riverside

CSE

Outline

- ✓ FPGA Introduction
- ✓ NIDS and SNORT
- ✓ Regular Expressions and PCRE
- ✓ PCRE to FPGA via OPCODES
- ✓ Hardware Details
- ✓ Performance
- ✓ Conclusion

Field Programmable Gate Array

- Silicon devices, Millions of Logic Gates
- Connected by Programmable interconnects
- Can efficiently Abstract a Data path by eliminating load-stores and branch instructions
- Emerging platforms include SGI RASC, XD 1000, Intel QuickAssist Hardware, etc.

a PROCESSOR vs an FPGA (the Highway analogy)

Processor (Dual core)

SPEED LIMIT 100

2 parallel Highways

Throughput = 2 x 100 = **200.**

Field Programmable Gate Array

SPEED LIMIT 10

200 parallel Highways

Throughput = 200 X 10 = **2000 !**

- ✓ What an FPGA loses in speed, it more than makes it up with parallelism
- ✓ Obtaining two orders of speedup is easily obtainable on an FPGA
- ✓ FPGAs are programmed with Hardware Description Languages

Network Intrusion Detection System (IDS)

- ✓ Detects and filters unwanted network packets (worms, spam, etc)
- ✓ Inspects payload as it enters or leaves a network, with set of rules
- ✓ Highly processor intensive with increasing number of rules

Network Intrusion Detection System (IDS)

- The IDS may become a bottleneck
- 10Mbps delivered by a Pentium 4 CPU
- Compare that to 10Gbps throughput of typical networks

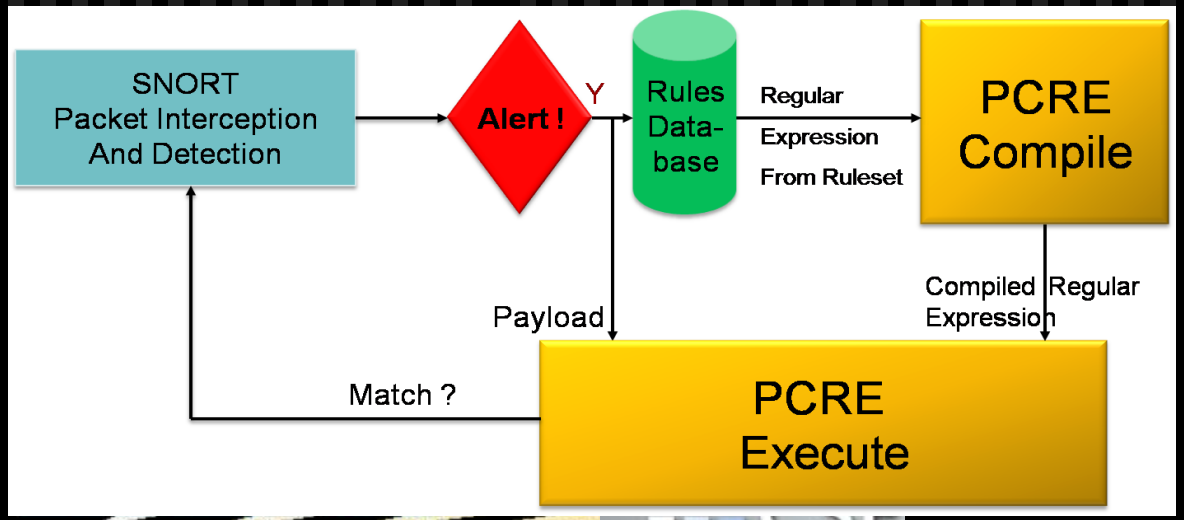
SNORT IDS rules and REGULAR EXPRESSIONS

- SNORT IDS rules are used to capture signatures of malicious activity on the network (e.g. WORM activity)
- A rule written as a regular expression is compact, powerful and highly expressible (one rule matches multiple possible strings)
- SNORT IDS uses the PCRE (PERL compatible regular expressions) as the language in which the rules are written

SNORT and PCRE

SNORT Ruleset	Regular Expression Rule	Attack Type	Implication
backdoor	pcre:"/^Netbus\s+\d+\x2E\d+/smi"	Netbus Trojan	Captures the header of the Netbus trojan i.e. Netbus followed by one or more spaces, one or more digits, character '.' and one or more digits.
web-misc	pcre:"/^[^\x3e\x3f\x26]{63}/R"	Buffer Overflow	Captures a McAfee specific buffer overflow attack sequence i.e. Any 63 characters other than >, ? or & .

- SNORT uses PCRE to match network packets on regular expression based signatures



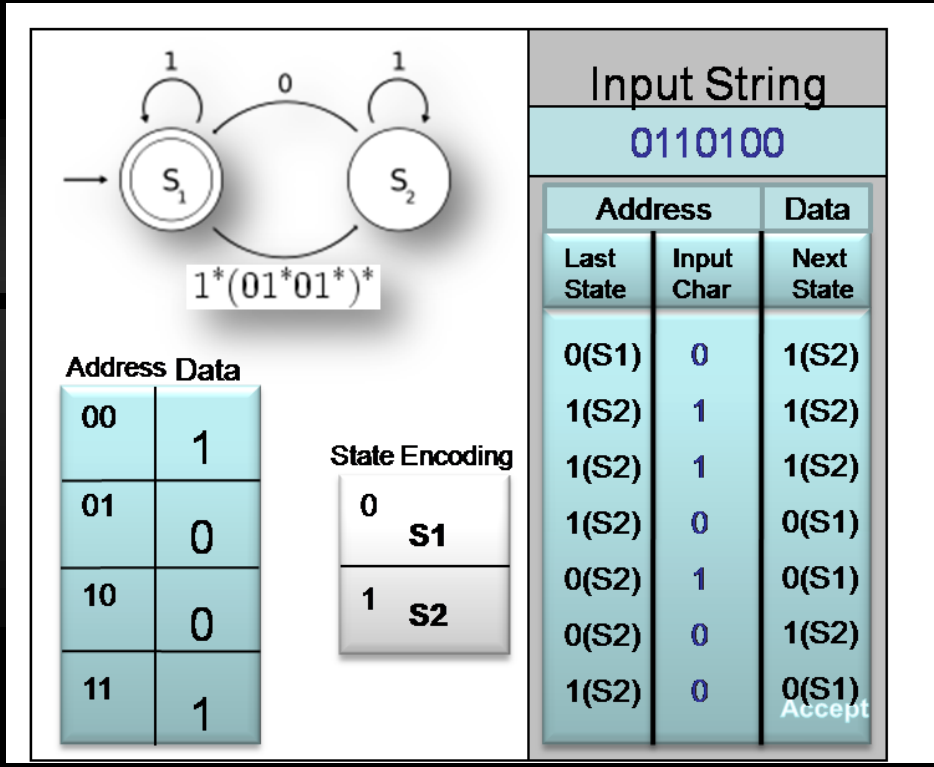
PCRE is an open source software that compiles and matches PERL regular expressions

Regular Expressions and Finite Automata

- A Regular Expression can be implemented as an Finite Automata
- A processor can execute only one finite automata per core
- An FPGA can execute hundreds of finite automata in parallel!
- Possible (Speedup!) over a Processor

Regular Expression and Finite automata on FPGA

- A regular expression viz. 1*(01*01*)* Can Identify Even number of zeros in a string composed of alphabets 0 and 1
- The equivalent Finite Automata of this regular expression can be implemented in hardware
- Each state of the automata is encoded using one bit
- The transitions are encoded using two bits



Compiling Regular Expression to OPCODES

Regular Expression Operator	PCRE Op Code
Match after the first character "^"	OP_CIRC 19
Match after the last character "$"	OP_DOLL 20
Quantifiers "**{}**"	OP_EXACT 32
Ranged Quantifiers "**{n,x}**"	OP_UPTO 30
Negated Character Class "**[^...]**"	OP_NCLASS 60
Repetition "*****"	OP_STAR 24
Repetition "**+**"	OP_PLUS 26
Back References "**\1,\2,\3,\4,\5,\6** "	OP_REF 62

- We utilize the PCRE complier to obtain OP Codes corresponding to regular expression operators in the SNORT rules

Generating Hardware from PCRE OPCODES

- We compile the OPCODES obtained to VHDL
- Each OPCODE corresponds to a VHDL template
- The template is filled, based on additional parameters accompanying each OPCODE

Generating Hardware from OPCODES

- The VHDL blocks are tied together as an NFA (Non deterministic Finite Automata)
- Additional hardware connects the NFA to the memory controller
- Memory controller obtains network payload form the host CPU and transfers it to the NFAs

SNORT Rule to PCRE OPCODES v7.0

- ✓ Example Rule snippet "/^NetBus\s+\d+/"
- ✓ After Compilation

80 0 20 19 21 78 21 101 21 116 21 66

Start ^ -> N -> e -> t -> b

21 117 21 115 44 8 44 6 0 20 0

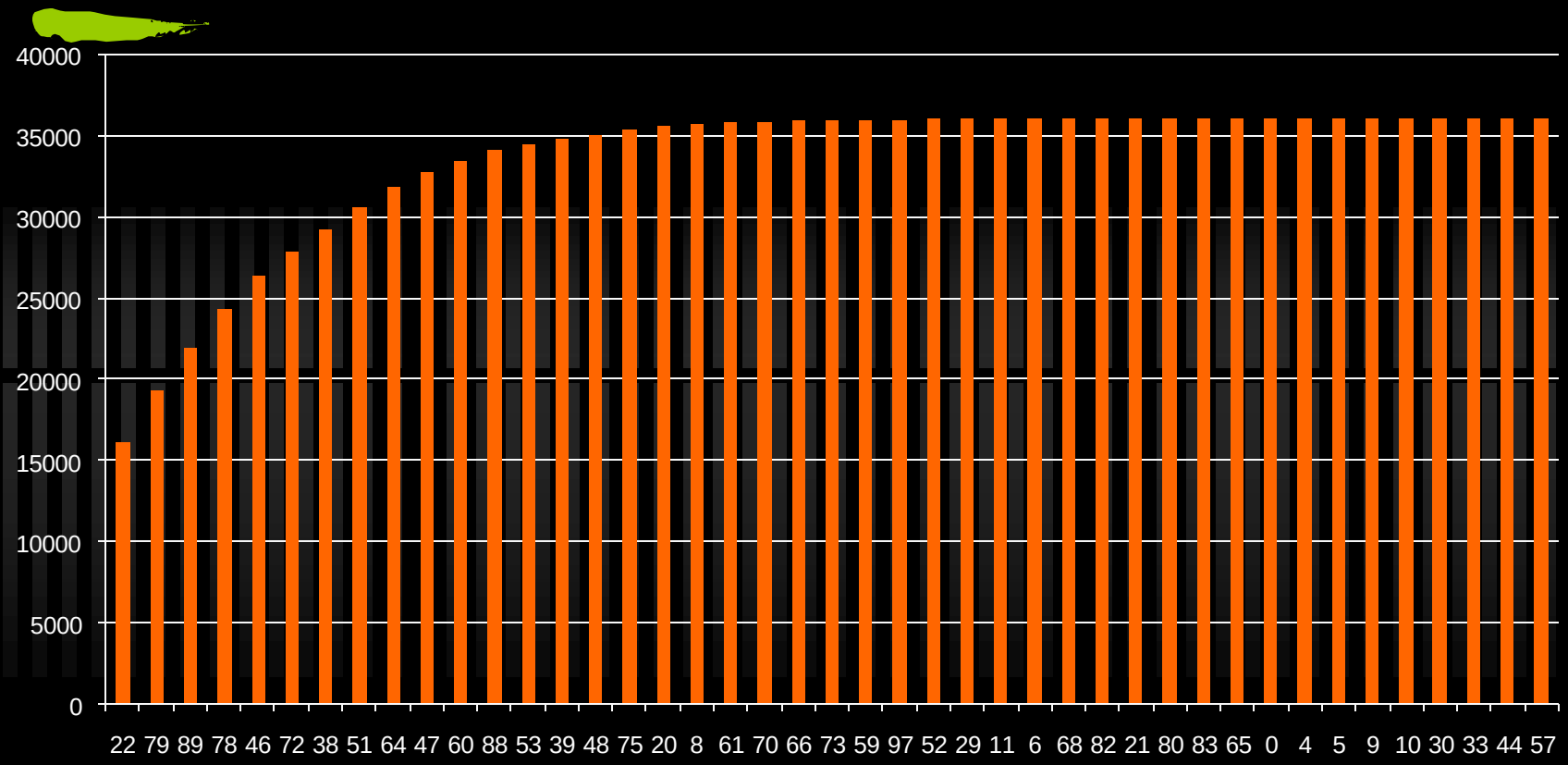
-> u -> s + \s + \d END

OPCODES are the common intermediate representation for software or hardware execution

An example VHDL Block generated by compiling the opcodes

```
if (clk'event and clk = '1') then
    if (start = '1') then
        char1_1 <= mem(conv_integer(nfa1));--N
        char2_1 <= mem(conv_integer(nfa1)+1);--e
        char3_1 <= mem(conv_integer(nfa1)+2);--t
        char4_1 <= mem(conv_integer(nfa1)+3);--b
        char5_1 <= mem(conv_integer(nfa1)+4);--u
        char6_1 <= mem(conv_integer(nfa1)+5);--s
        if ((char1_1 = conv_std_logic_vector(78, 8)) and
            (char2_1 = conv_std_logic_vector(101, 8)) and
            (char3_1 = conv_std_logic_vector(116, 8)) and
            (char4_1 = conv_std_logic_vector(98, 8)) and
            (char5_1 = conv_std_logic_vector(117, 8)) and
            (char6_1 = conv_std_logic_vector(115, 8))
          ) then
            match_1 <= '1';
          else
            match_1 <= '0';
        end if;
    end if;
end if;
```

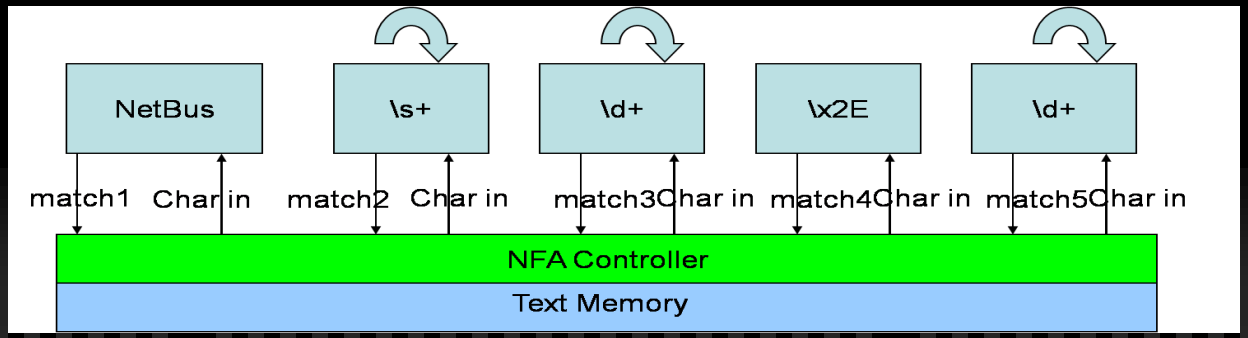
CDF of Opcodes from regexes in SNORT DB 2.6 compiled with PCRE v7.1

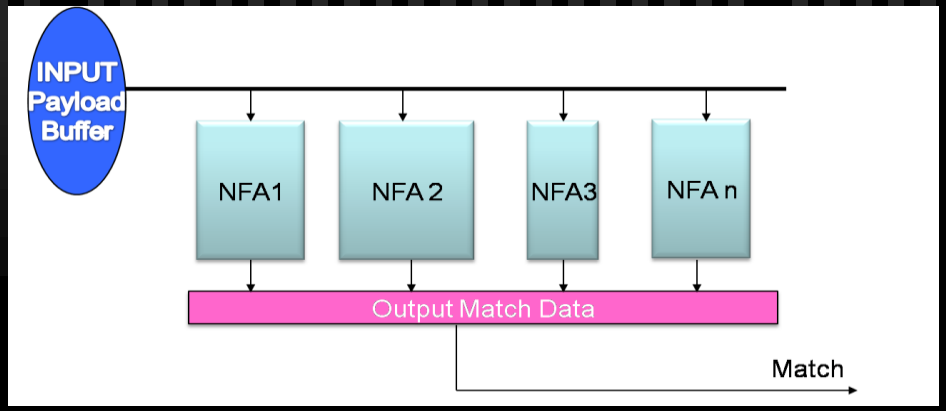


The most frequently occurring OP Code corresponding to a single character match (OPCODE 22)

The Finite Automata on hardware

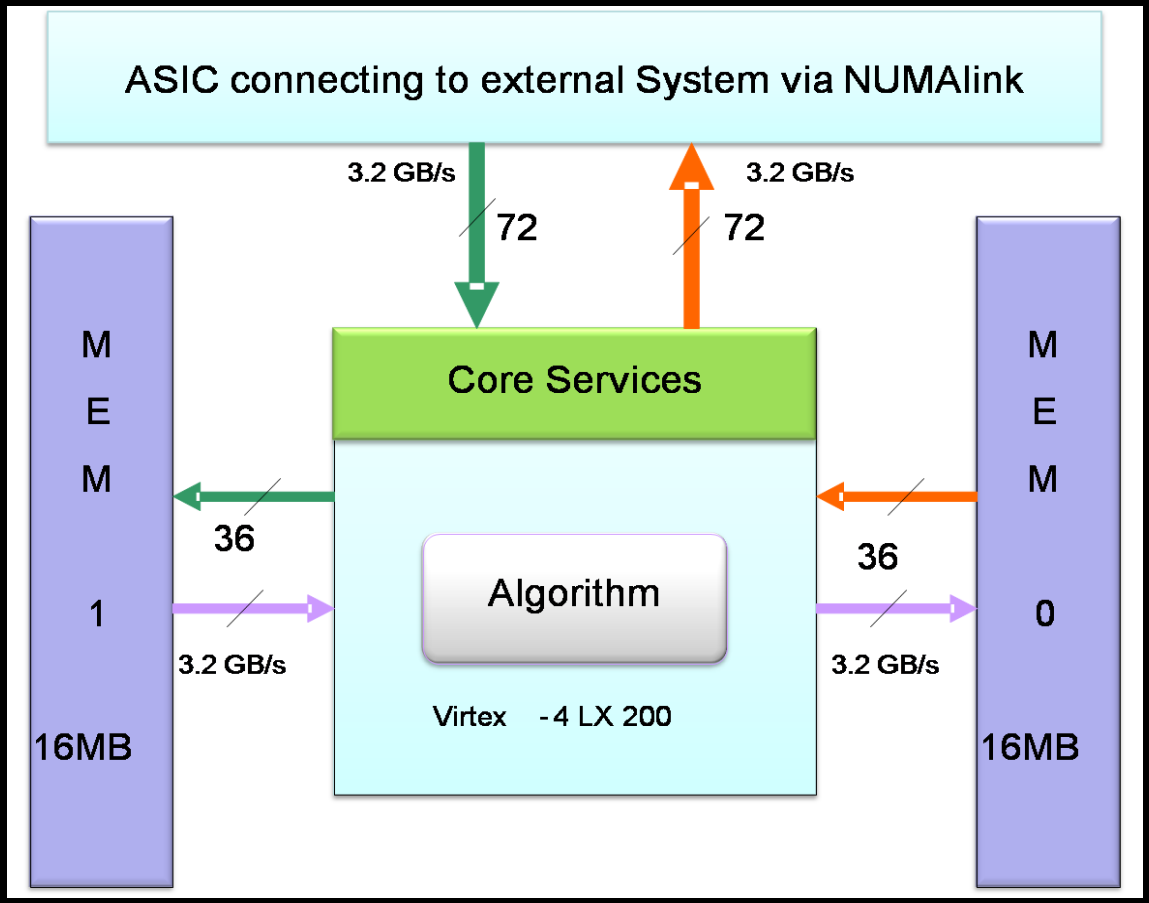
- The NFA





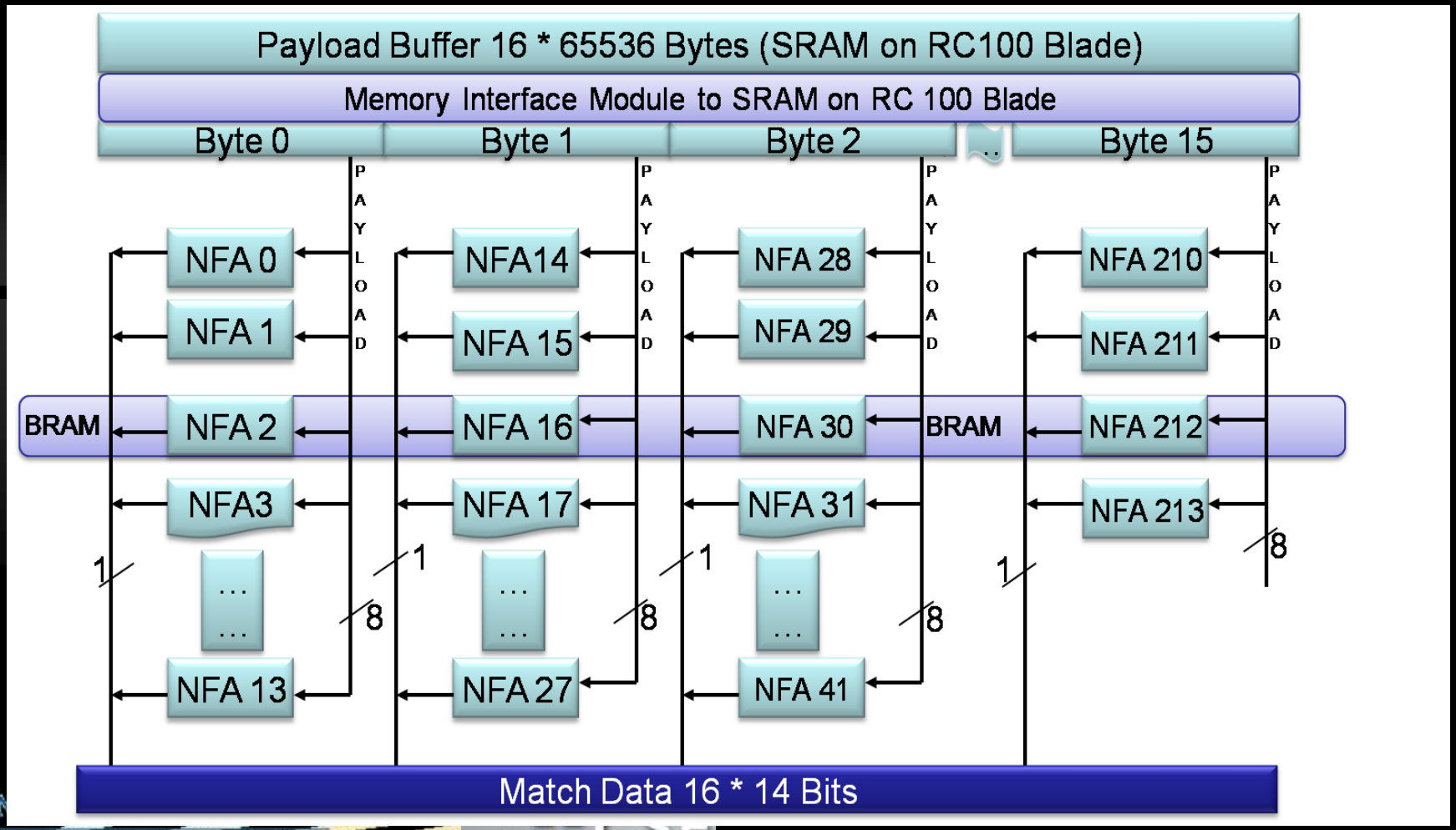
- Multiple NFA engines are implemented to match multiple SNORT rules on a network payload

The SGI RASC RC 100 BLADE

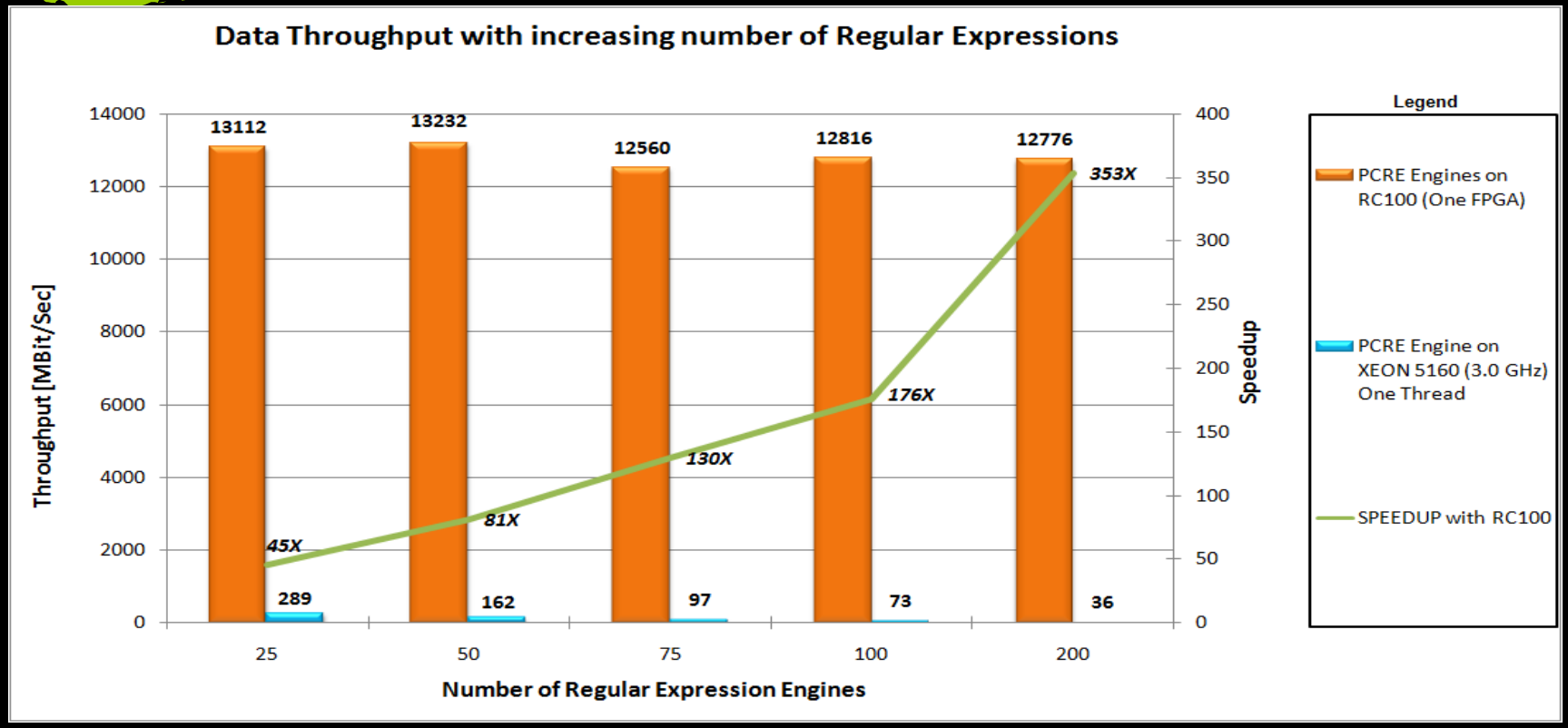


- The RASC Blade contains two XILINX Virtex-4 LX 200 FPGA and provides upto 7.2 GByte/s throughput

Architecture of 214 NFA Engines on a Virtex 4 LX 200 FPGA

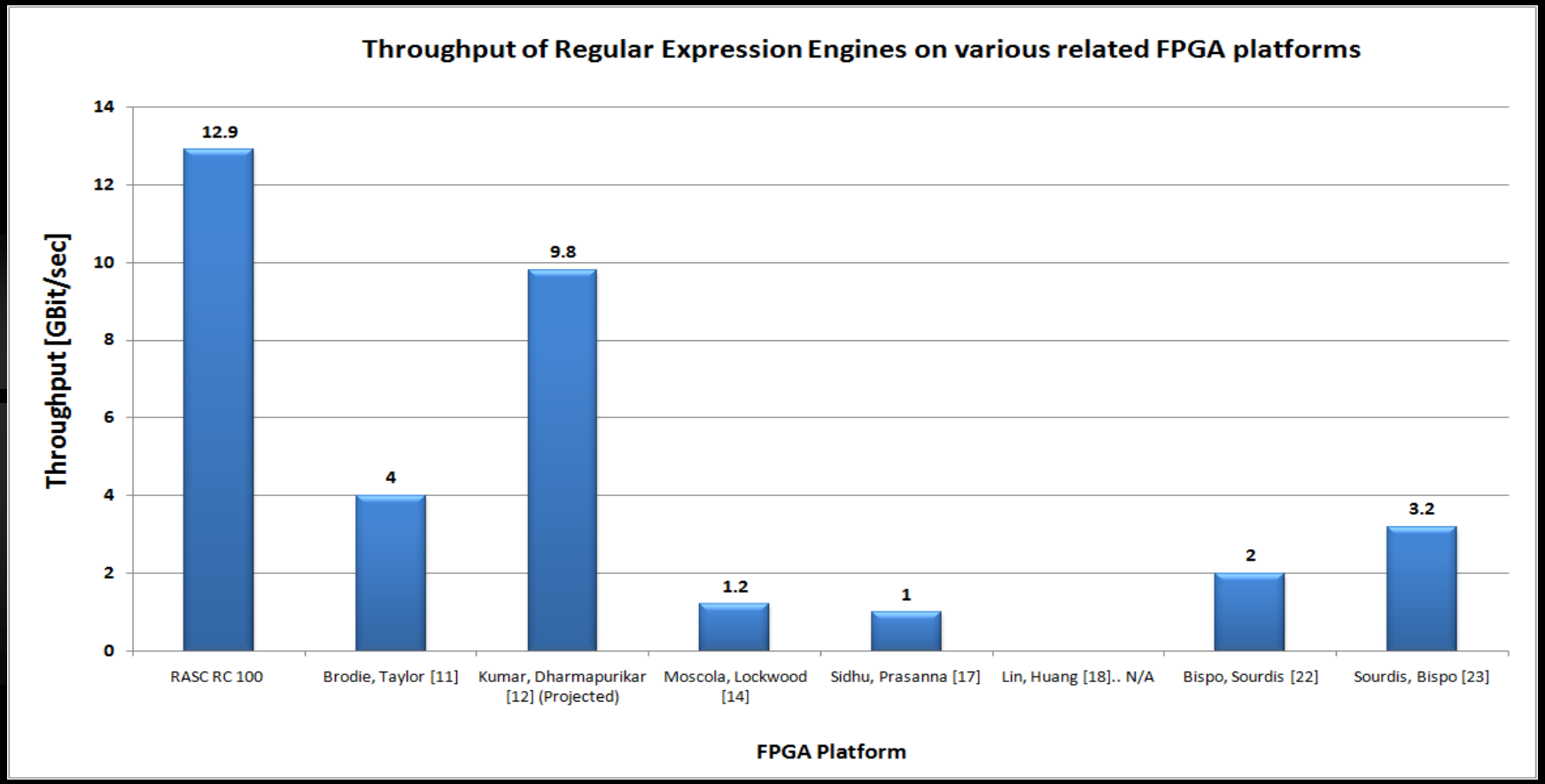


Throughput and Speedup



- ✓ It is possible to obtain 12.9 Gbps throughput on the RASC RC-100 hardware

Comparison with related systems



- ✓ The Compiled PCRE Op Codes on **SGI RASC RC-100** provides the highest throughput among other related FPGA platforms used for NIDS

Conclusion

- ✓ Compiled PCRE OPCODES to VHDL
- ✓ Implemented Fast NFA based Regular Expression engines on FPGA platform
- ✓ Regex engines Operate at 12.9 Gbps for efficient 10GbE IDS

Future Work

- ✓ Utilize multiple FPGAs to encompass all the SNORT rule-sets
- ✓ Use streaming data flow for transferring payload to the FPGA
- ✓ Implement all the OPCODES in hardware

Q&A

THANK YOU!

```
result = pcre_exec(pcre_data->re,          /* result of pcre_compile() */
                   pcre_data->pe,          /* result of pcre_study()   */
                   buf,                    /* the subject string */
                   len,                    /* the length of the subject string */
                   start_offset,           /* start at offset 0 in the subject */
                   0,                      /* options(handled at compile time */
                   ovector,                /* vector for substring information */
                   SNORT_PCRE_OVECTOR_SIZE); /* number of elements in the vector */

    if(result >= 0)
    {
        matched = 1;
    }
    else if(result == PCRE_ERROR_NOMATCH)
    {
        matched = 0;
    }
```

Potential point for Speedup (i.e. implement pcre_exec in hardware)