

An Energy Efficient Datapath for Asymmetric Cryptography

Andrew Targhetta

Dr. Paul V. Gratz

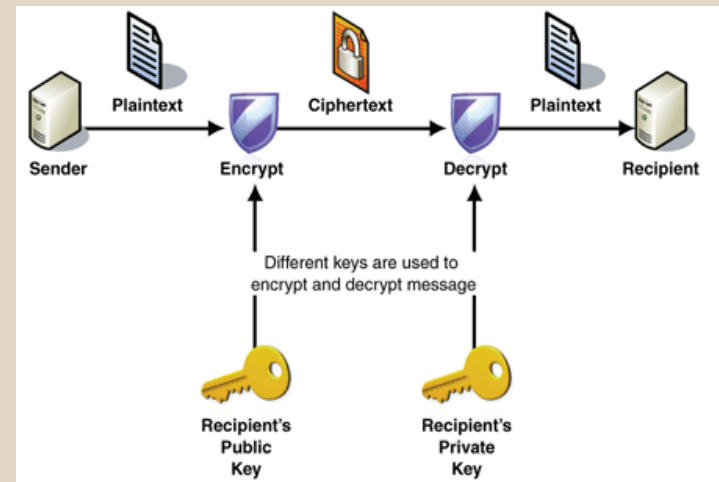
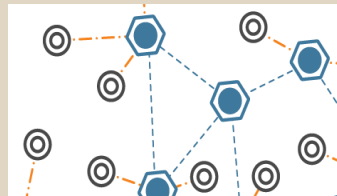
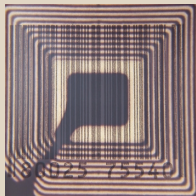
Outline

- **Introduction**
 - **Background**
 - **Parameterizable Microarchitecture**
 - **Methodology**
 - **Results**
 - **Concluding Remarks**
-

Introduction

Asymmetric Cryptography

- Solution for
 - Key establishment
 - Digital Signature



- Energy requirements impractical for
 - Wireless Sensor Networks (WSN)
 - Radio Frequency Identification Tags (RFID)

Introduction

Objective

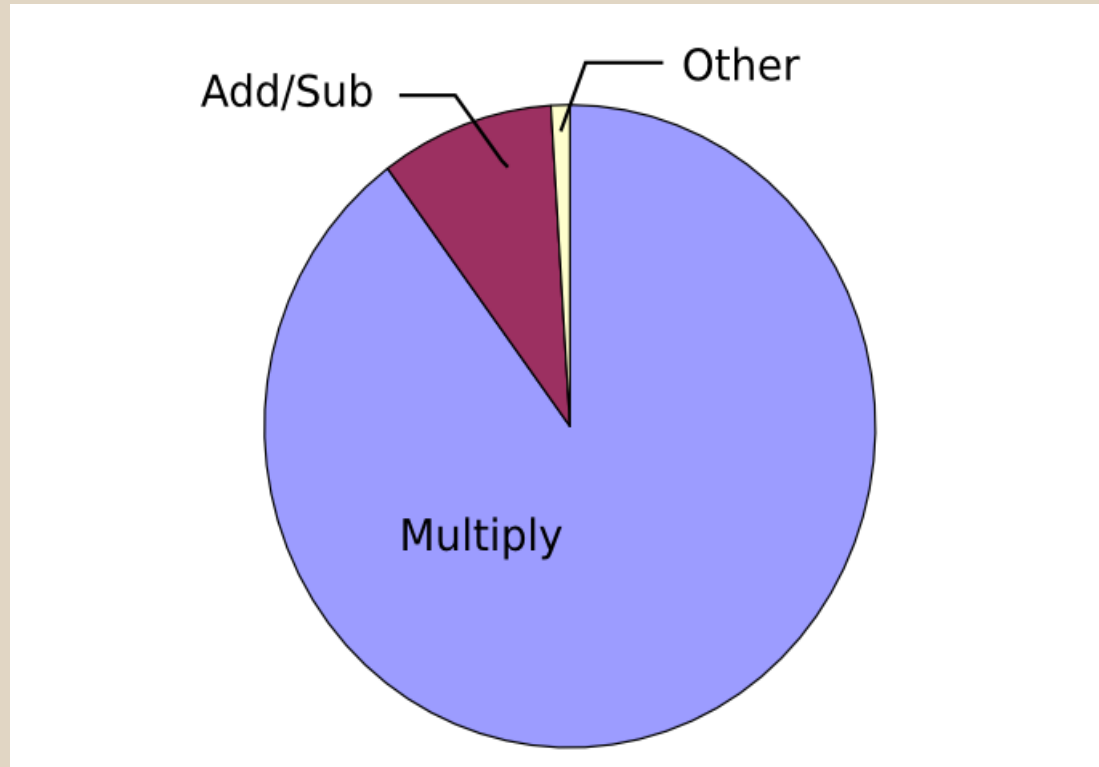
- Introduce a unique microarchitecture
 - Accelerates asymmetric cryptography
 - Parameterizable datapath width
- Evaluate energy consumption
 - In terms of datapath width
 - In terms of key size
- Determine energy optimal datapath width
 - For a given key size



Background

Approximate percentage of execution time for ECC

- **Multiply: 89.8%**
- **Add/Sub: 9.2%**
- **Other: <1%**



Multiply is dominant and the FFAU can accelerate over 99%!

Background

CIOS Montgomery Algorithm

- Computes $A*B$ modulo N
 - Sort of...
 - Multi-precision integers (~256 bits)
- Efficient algorithm
 - Uses Montgomery reduction for modulo
 - Interleaves reduction with multiplication
 - Removes unnecessary computation

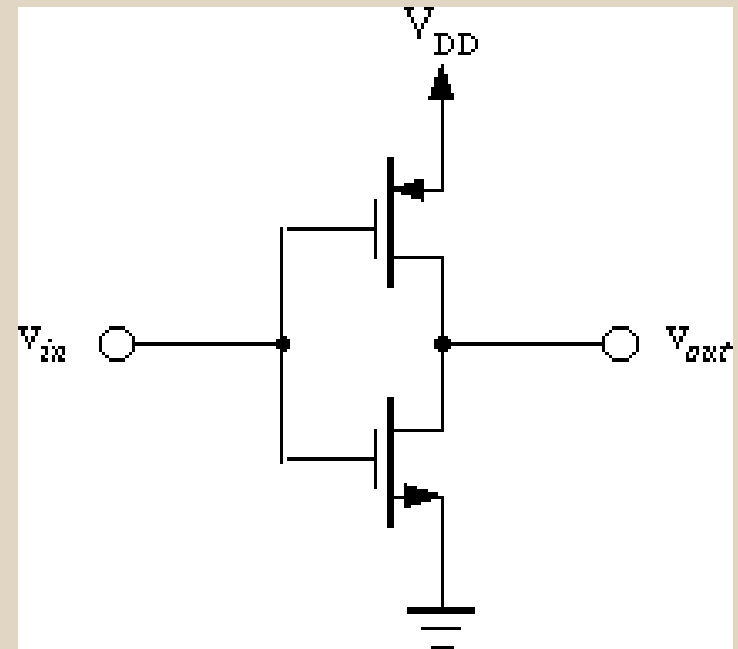
Background

- **Energy**
 - Energy = Power*Time
- **Energy is reduced by**
 - Reduce time of computation
 - Reduce power consumption of computational device

Background

How is energy consumed in CMOS?

- Static Power
 - $V * I_{leak}$
- Dynamic Power
 - Switching power
($0.5 * C * V^2 * f$)
 - Internal power
($.5 * C_{int} * V^2 * f$) + ($V * I_{sc}$)



Microarchitecture

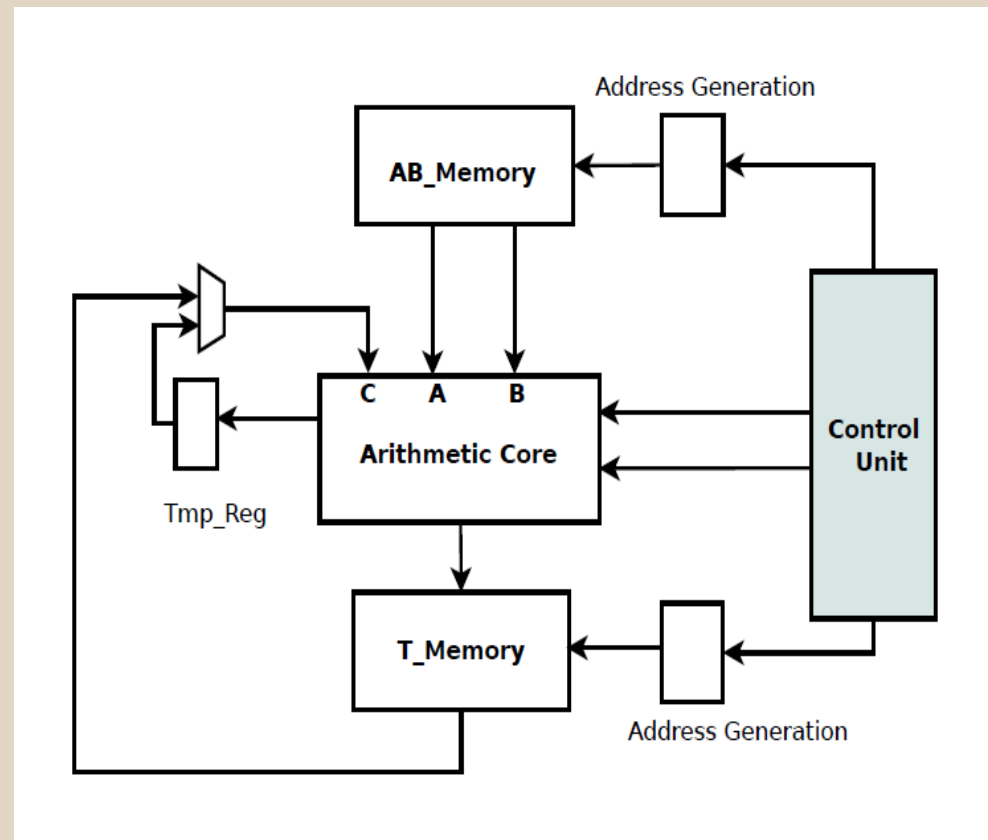
Highlights

- Parameterizable and Reconfigurable
- Computes modular math (192 to 384 bit)
- Accelerates ECC and RSA
- ~100% utilization of Arithmetic core
- Referred to as Finite Field Arithmetic Unit (FFAU)

Microarchitecture

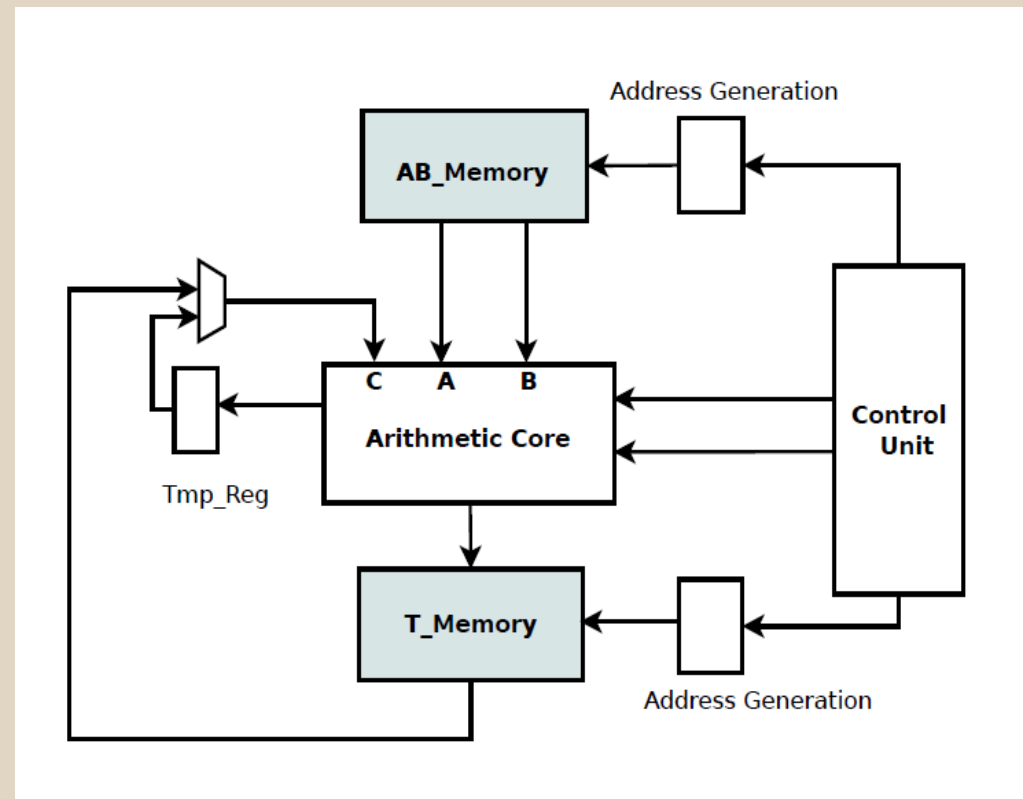
FFAU

- Control Unit
 - Constant memory
 - Microcoded with built-in loop indexing
 - Fast (<64 deep ROM)



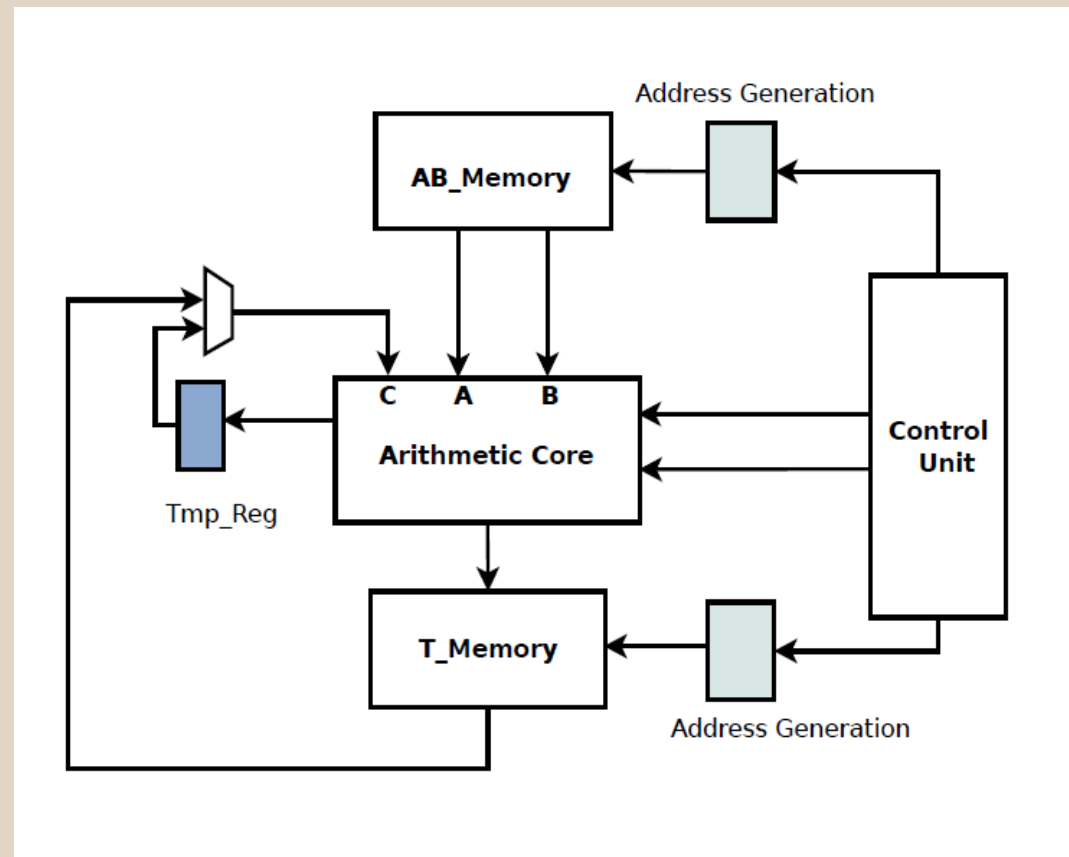
Microarchitecture

- Data Memory
 - Split memory allows 3 operand fetching
 - Each module ~1kb depending on size of finite field



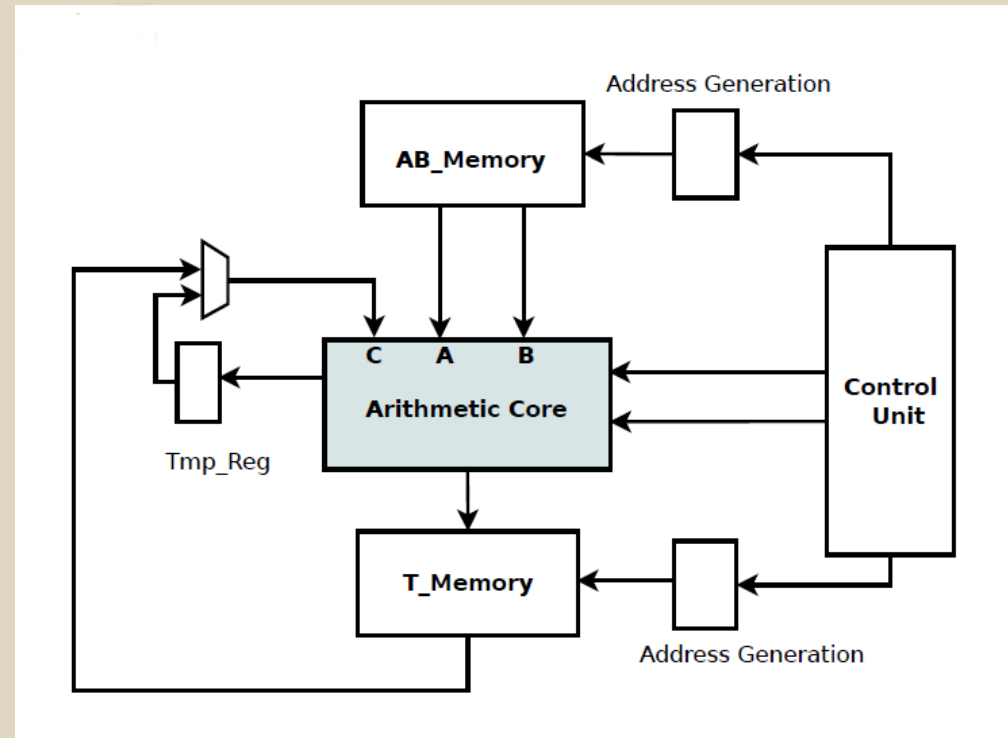
Microarchitecture

- Address Generation Logic
 - Concurrent memory address calculation
- Temp Register
 - Temporary Intermediate value storage



Microarchitecture

- Arithmetic Core
 - $\{\text{carry}, \text{result}\} \leftarrow A * B + C + \text{carry}$
 - $\{\text{carry}, \text{result}\} \leftarrow A + B + C + \text{carry}$
 - Pipelined
 - Self-draining

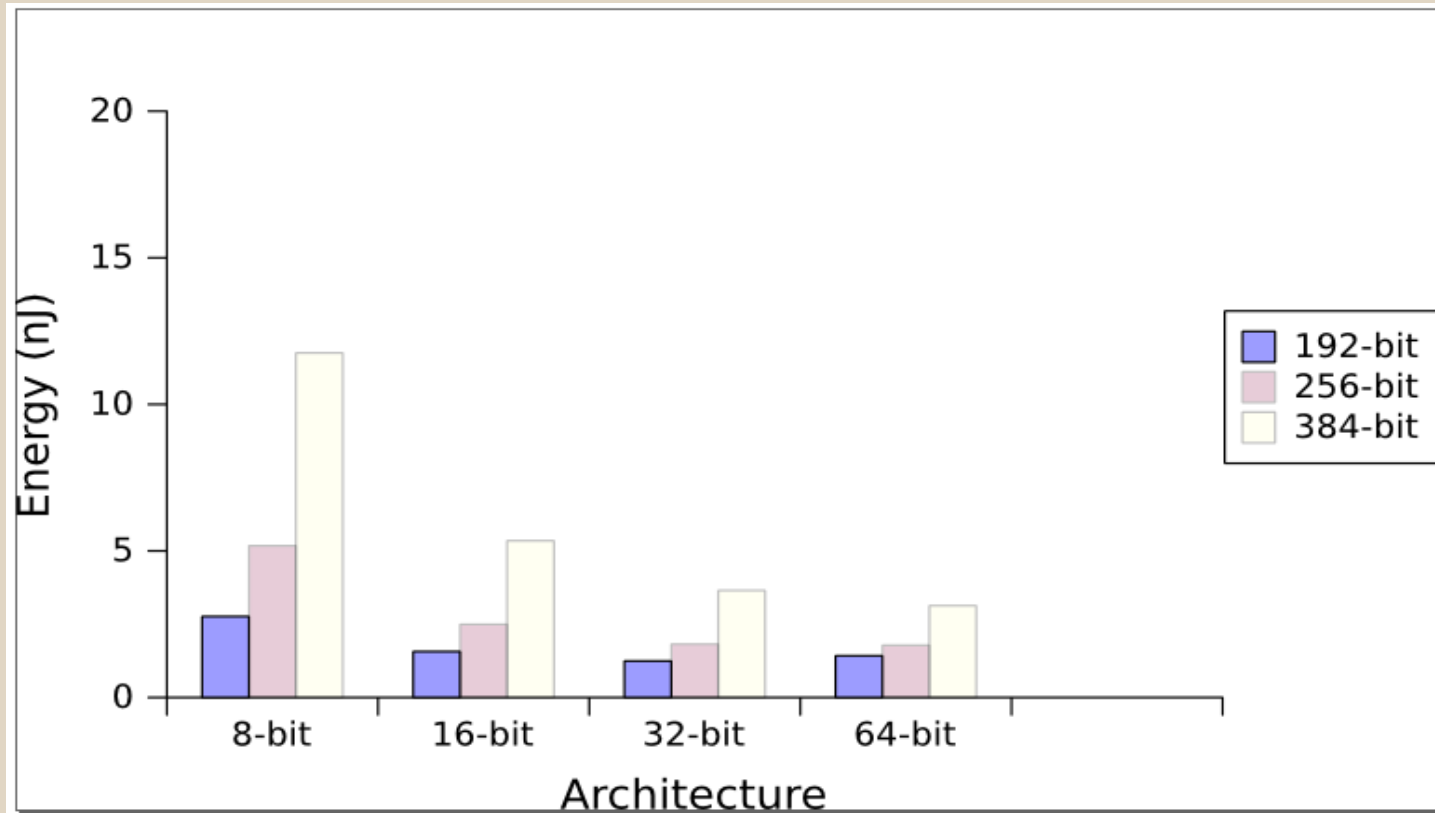


Methodology

Energy Estimation

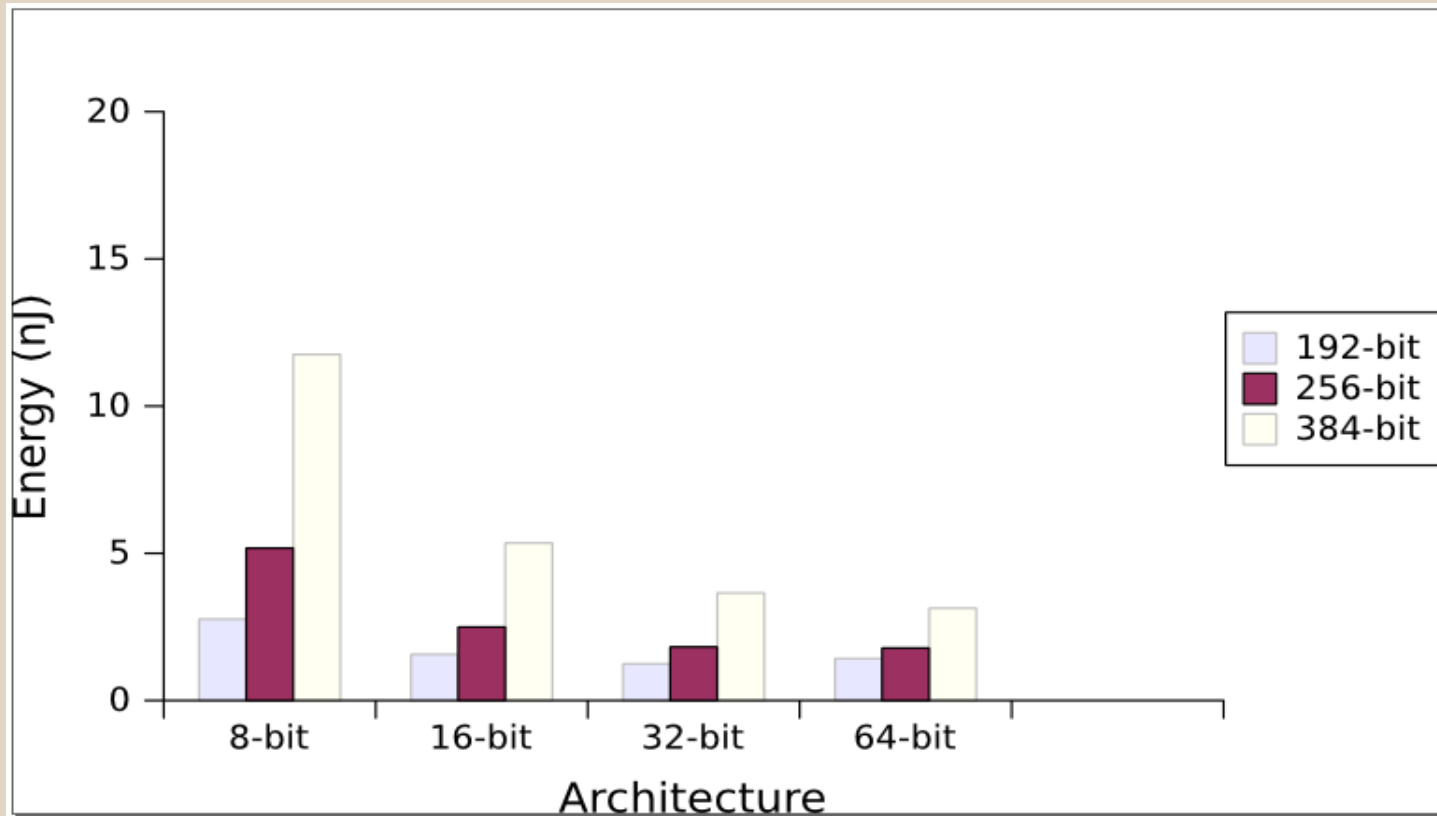
- Synopsys tool chain for logic power
 - HDL Compiler with 45nm tech library
 - Verilog Compiler Simulator (activity info)
 - PrimeTime PX with activity info
- Cacti for RAM power
 - Estimate static and dynamic power
- Extract computation time from simulation
- Datasheet, simulator, and scaling formula for ARM energy estimation

Results



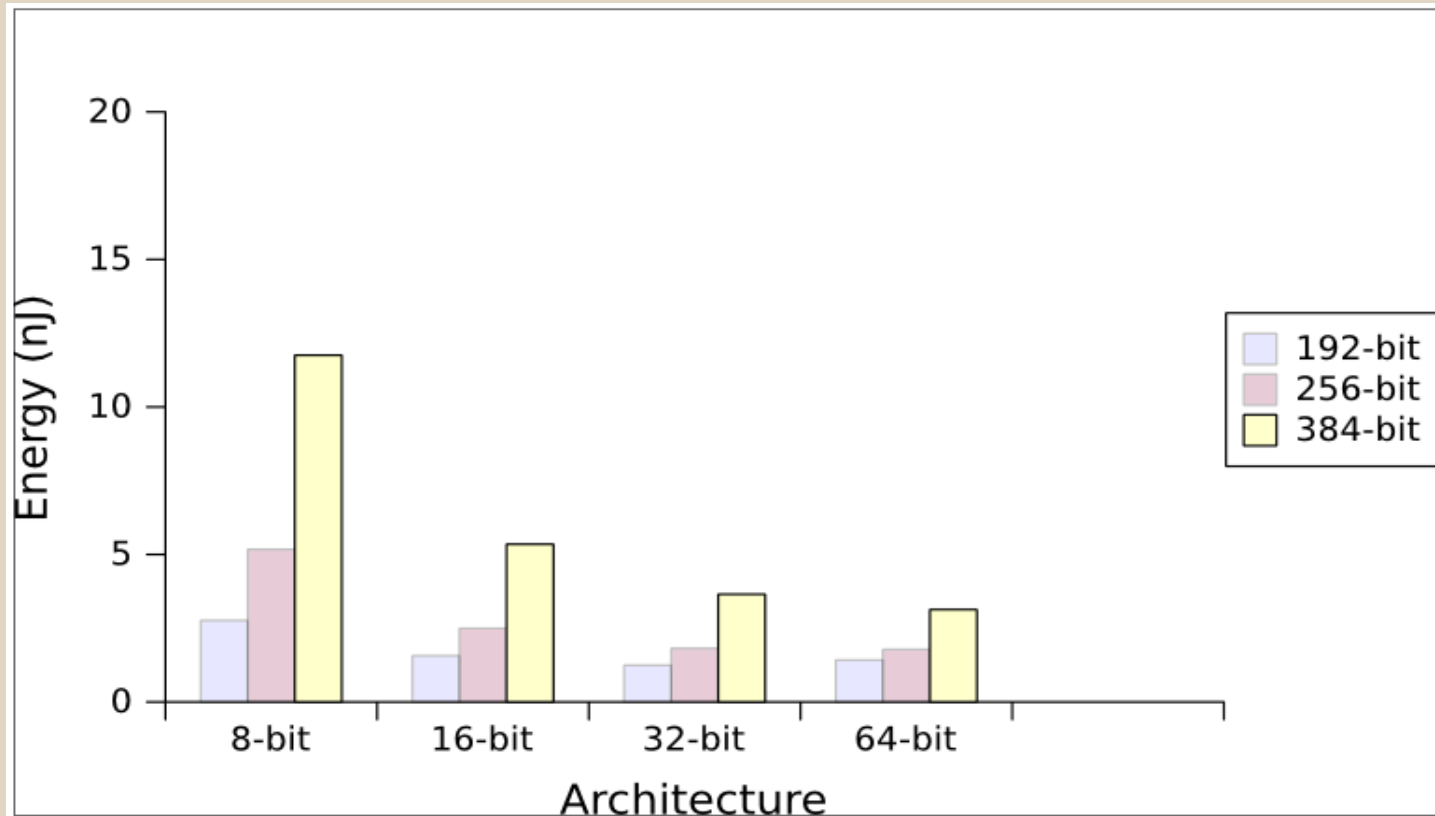
**Energy per Montgomery
Multiplication**

Results



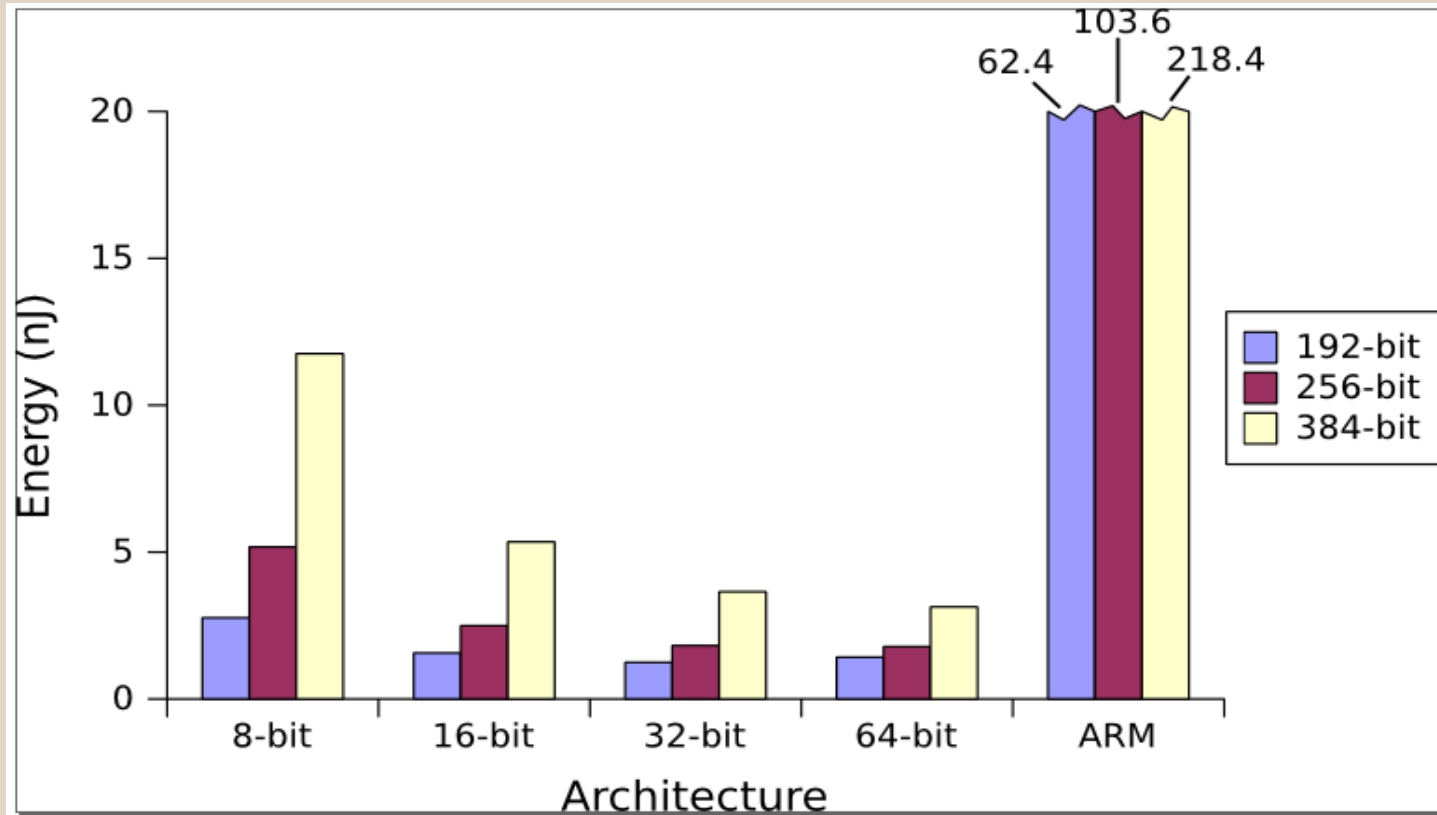
**Energy per Montgomery
Multiplication**

Results



**Energy per Montgomery
Multiplication**

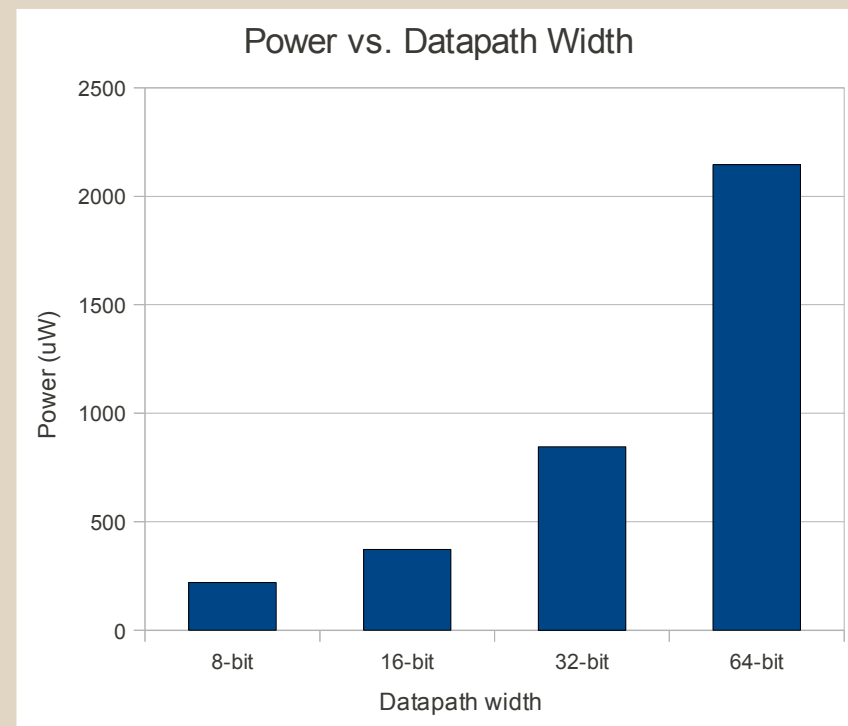
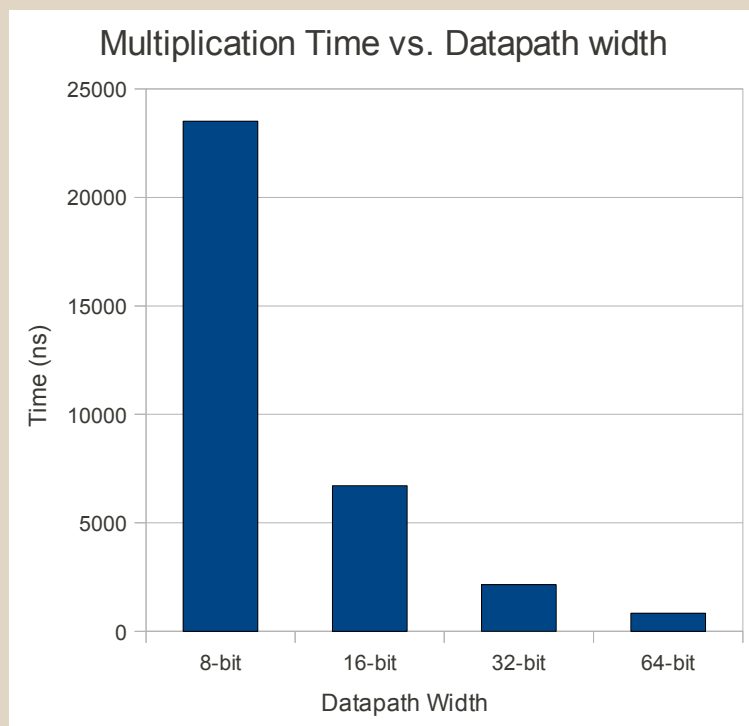
Results



**Energy per Montgomery
Multiplication**

Results

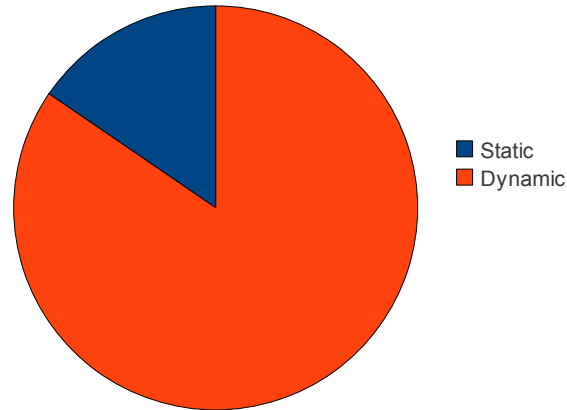
Execution time and Average Power



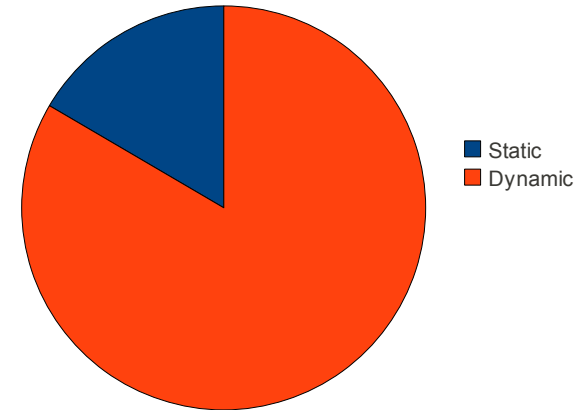
Results

- Dynamic power is dominant
- Percentage varies only slightly with width

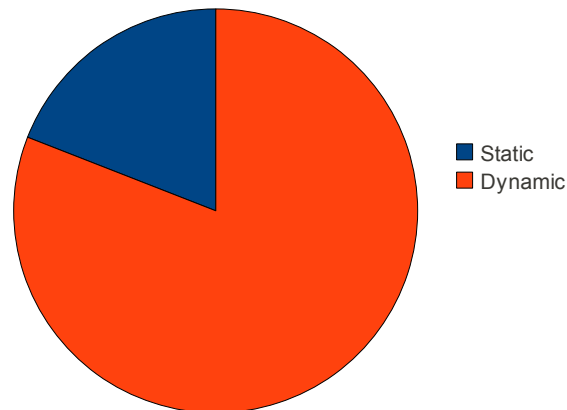
Average Power
8-bit Datapath



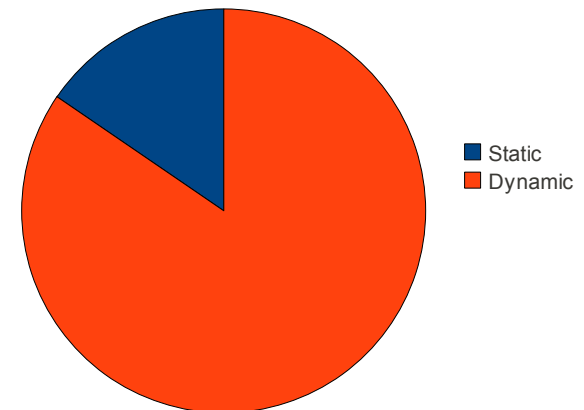
Average Power
16-bit Datapath



Average Power
32-bit Datapath



Average Power
64-bit Datapath



Concluding Remarks

Accelerator Generalizations

- $O(n^2)$ computational complexity tends to favor larger datapath
- $O(n)$ only slightly affected by changes in datapath size
- $O(1)$ favors smaller datapath width
- Custom hardware greatly improves energy efficiency compared to software

Concluding Remarks

FFAU Observations

- Power increases $<$ quadratically with word size
- Time per operation decreases by k^2
- Lower energy design favors larger word size
- Computational complexity not perfectly quadratic creating a sweet spot

Questions??

Previous Work

- GroBschadl and Kamendje
 - Designed a “low power” MAC unit
 - Did not report power
- Goodman and Chandrakasan
 - Implemented entire crypto processor
 - Targeted FPGA
 - Showed order of decreasing power consumption to be
SW → FPGA → ASIC

Algorithm 4.6 Calculate $t = \text{MontMult}(a, b, n, n'_0)$ (CIOS) [12]

```
1:  $t \leftarrow 0$ 
2: for  $i$  from 0 to  $k - 1$  do
3:    $C \leftarrow 0$ 
4:   for  $j$  from 0 to  $k - 1$  do
5:      $(C, S) \leftarrow T[j] + A[j] * B[i] + C$ 
6:      $T[j] \leftarrow S$ 
7:   end for
8:    $(C, S) \leftarrow T[k] + C$ 
9:    $T[k] \leftarrow S$ 
10:   $T[k + 1] \leftarrow C$ 
11:   $C \leftarrow 0$ 
12:   $m \leftarrow T[0] * n'_0$  modulo  $2^w$ 
13:   $(C, S) \leftarrow T[0] + m * N[0]$ 
14:  for  $j$  from 1 to  $k - 1$  do
15:     $(C, S) \leftarrow T[j] + m * N[j] + C$ 
16:     $T[j - 1] \leftarrow S$ 
17:  end for
18:   $(C, S) \leftarrow T[k] + C$ 
19:   $T[k - 1] \leftarrow S$ 
20:   $T[k] \leftarrow T[k + 1] + C$ 
21: end for
22: if  $t \geq n$  then
23:   return  $t - n$ 
24: else
25:   return  $t$ 
26: end if
```
