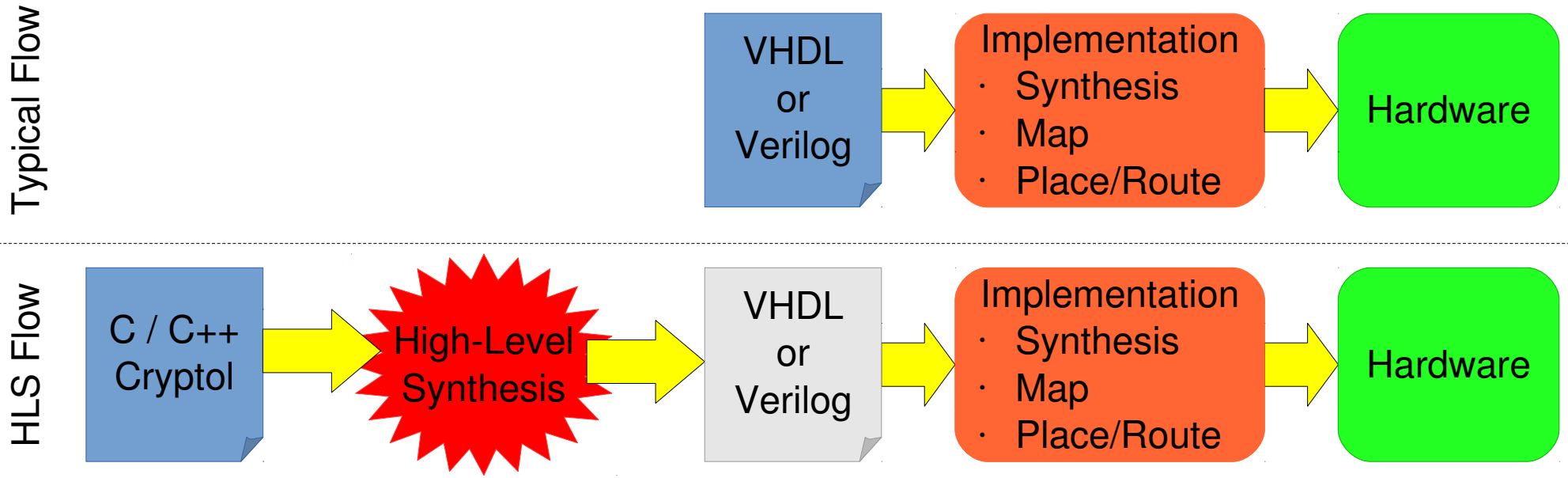
High-Level Synthesis of Cryptographic Hardware

Jeremy Trimble
ECE 646

High-Level Synthesis

- Synthesize (FPGA) hardware using software programming languages:
  - C / C++,
  - Domain-specific Languages (“DSL”)



Why HLS?

- Improve time-to-market
  - HDL development is time-consuming, complex.
- Verification becomes more practical
  - Write “testbenches” in C/C++.
  - Runs faster than RTL simulation.
- Architecture Exploration
  - Multiple architectures from a single high-level spec.
  - Use directives to apply different architecture strategies.

High-Level Synthesis Tools

- Vivado HLS
  - Xilinx, Inc.

- LegUp
  - University of Toronto

UNIVERSITY OF TORONTO

- Cryptol
  - Galois, Inc.

Vivado HLS

- Generates Verilog/VHDL/SystemC from C/C++.
  - Given a top-level function
  - Control synthesis through directives
    - Loop Unrolling
    - “Inlining”
    - “Reshaping”
    - Clock rate
  - Variety of interfaces
    - Handshaking, FIFO, AXI Master/Slave/Stream
  - HW / SW “Co-simulation” for verification
    - Can write testbench in C / C++ and simulate
- Generates portable HDL code.

LegUp

UNIVERSITY OF TORONTO

- Synthesizes C to Verilog, given a top-level function.
  - Hardware generation is more “static.”
    - Forces a particular memory architecture.
      - *In Legup, each variable that uses memory is stored in a separate altsyncram and identified by a unique number called a tag.*
- Two “flows”
  - Pure Hardware
  - “Hybrid”
    - Automatically partition between hardware/software
- Altera-specific.

Cryptol

- Functional Domain-Specific Language
  - “Executable description” of a cryptographic algorithm.
  - Cryptol 1.X can compile to Verilog
    - Closed-source
  - No HDL support in Cryptol 2.X (open-source) yet
    - But it is on the way.

This Project

- Implement AES-GCM using HLS
  - “Galois/Counter Mode” – NIST SP800-38D
  - Authenticated Cipher mode
    - Uses AES as block cipher (128-bit in this case)
    - Builds on HLS implementation of AES developed by Homsirikamol, et. al. (“Ice”).
  - Similar to “CTR” mode discussed in class.
    - Uses Galois Multiplication for computation of tag.

- Implement AES using Cryptol.

Galois/Counter Mode Encryption

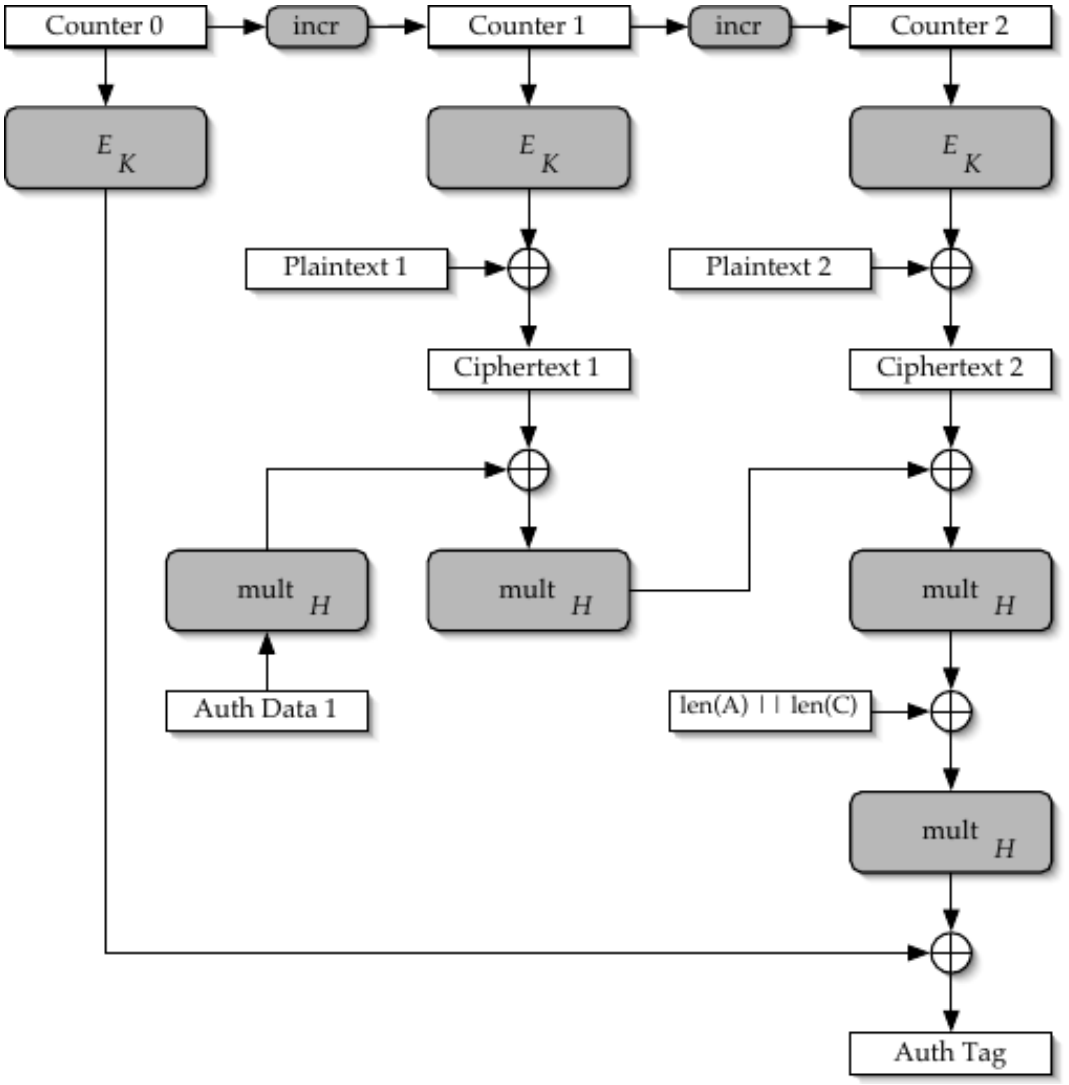


Diagram: Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, NIST Special Publication 80038D, November, 2007.

Galois/Counter Mode Decryption

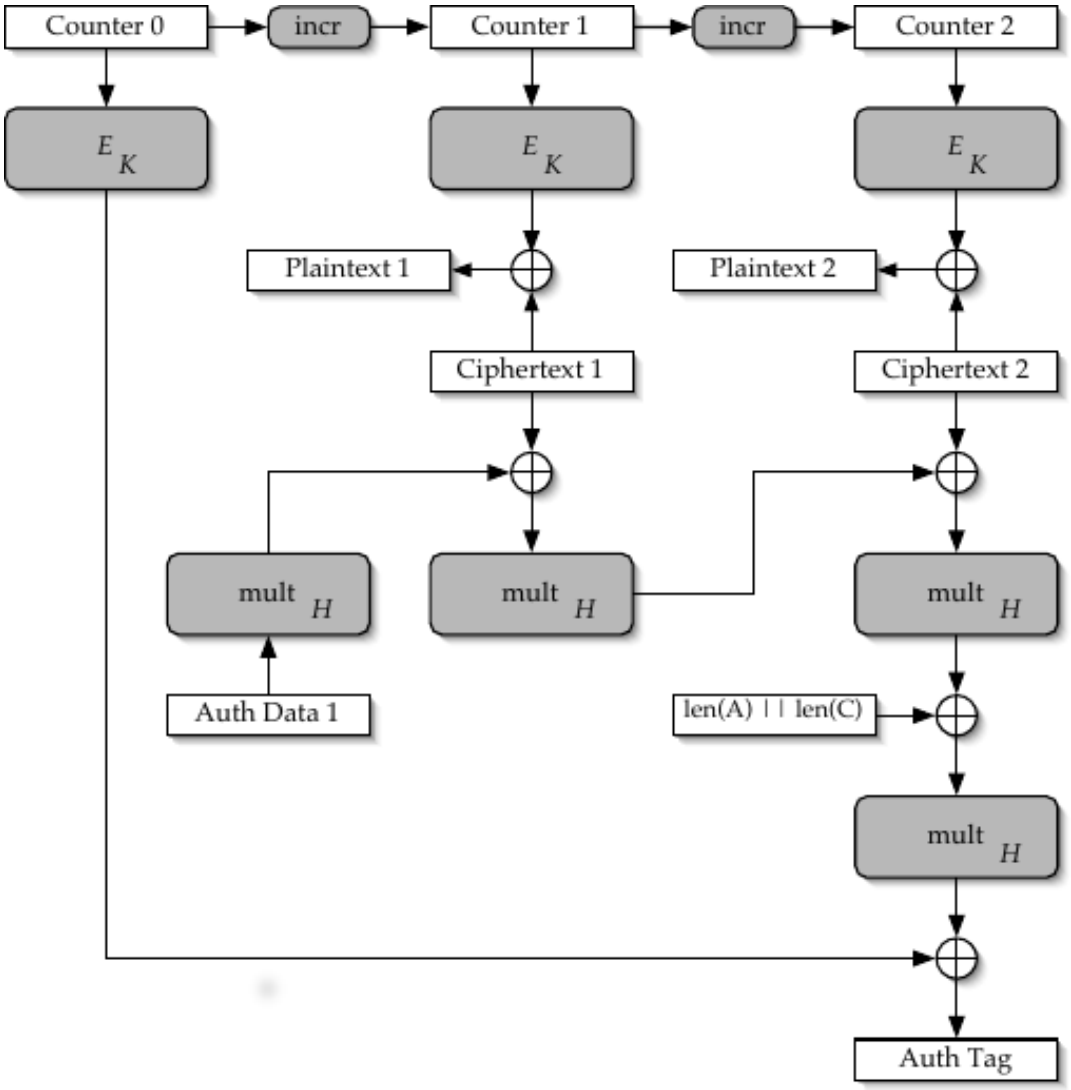
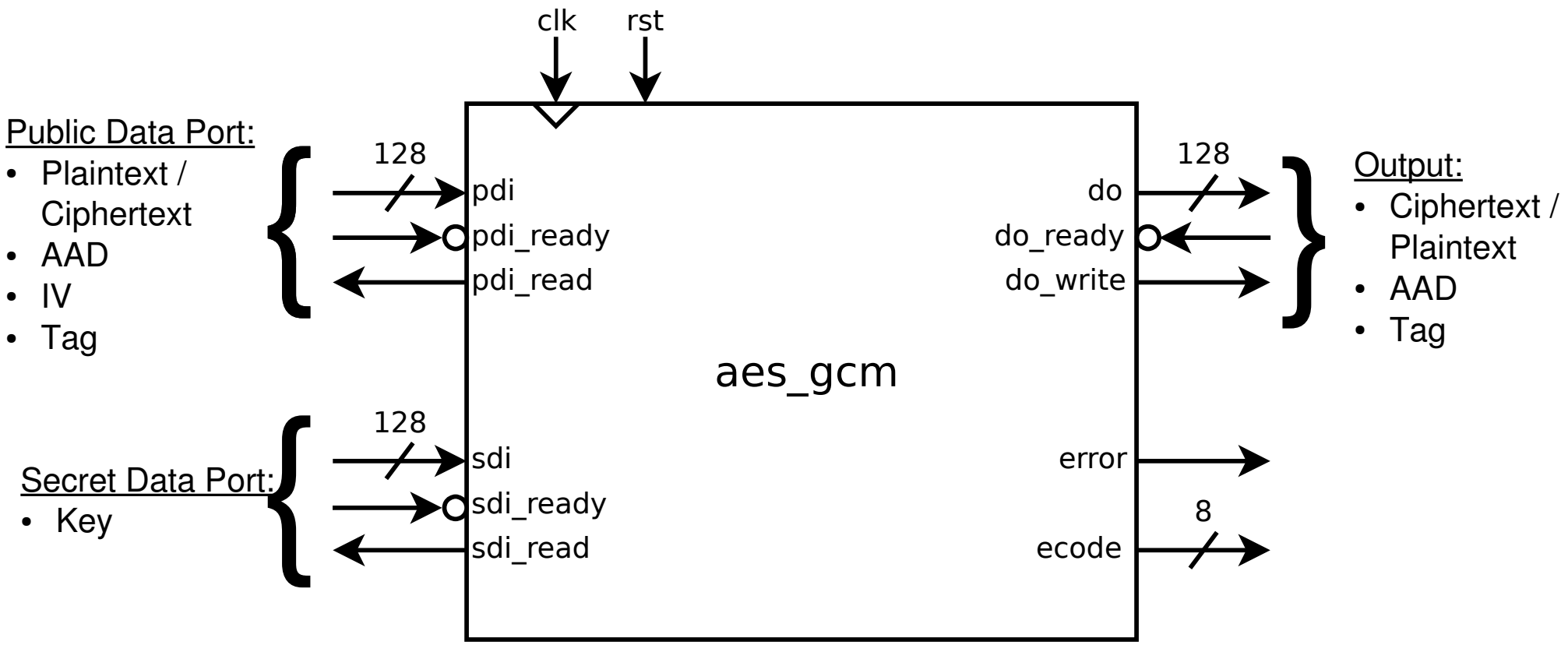


Diagram: Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, NIST Special Publication 80038D, November, 2007.

Top-Level Interface



Interface proposed in:

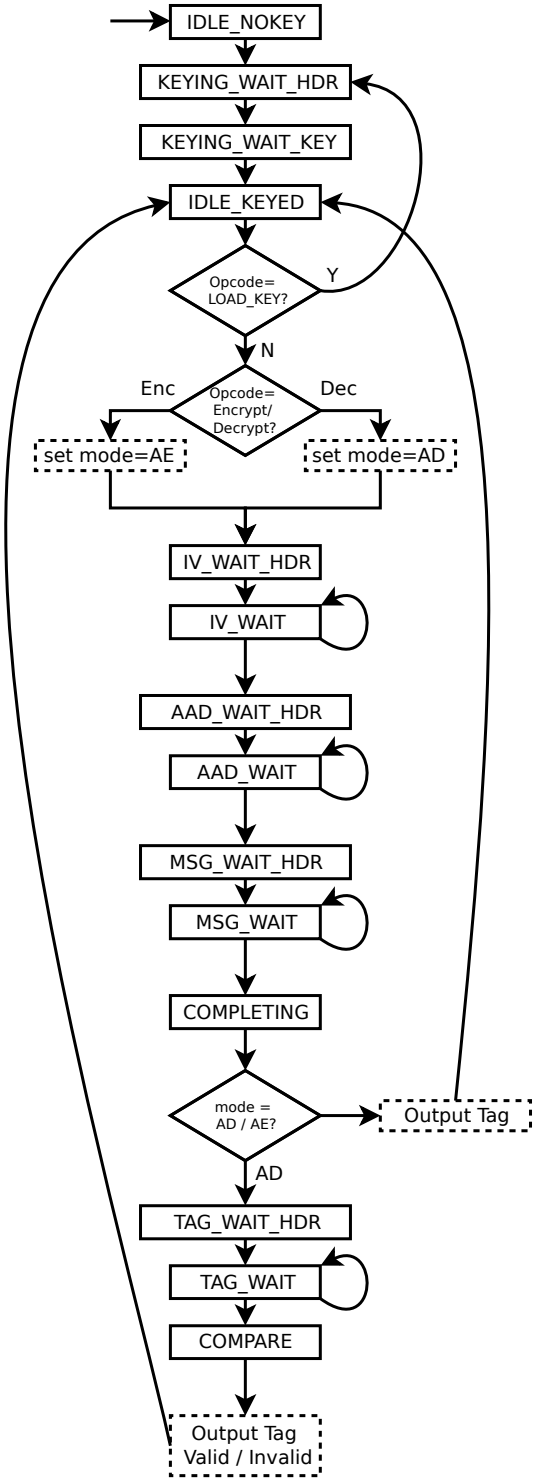
K. Gaj, E. Homsirikamol, and M. Rogawski, “Fair and Comprehensive Methodology for Comparing Hardware Performance of Fourteen Round Two SHA3 Candidates Using FPGAs,” in Cryptographic Hardware and Embedded Systems, CHES 2010

AES-GCM C Notes

- C prototype matches HW interface:

```
void aes_gcm(
    /* Public data port */
    pdi_t   * pdi,

    /* Secret data port */
    sdi_t   * sdi,

    /* Output data ports */
    dout_t  * dout,
    word8   * ecode
);
```

  - This may not have been the right choice...
- Used C-based testbench to verify correctness.
  - Repeatedly calls top-level `aes_gcm()` function.



AES-GCM C Notes

- Initial design latency too high (>200 cycles/blk)
  - Applied directives
    - Unroll, Inline, Reshape, Pipeline
  - C Optimizations
    - Optimized Galois Multiplication (based on Satoh, et. al.)
      - Reduced initiation interval from 128 to 8 cycles
    - Rewriting expressions
      - HLS sometimes “doesn't recognize” opportunities to save hardware.

AES-GCM C Notes

- Optimizations improved latency
  - But still not good (~38 cycles per 128-bit block)
  - Resource utilization became **very** large
- Issues
  - HLS
    - Directives help, but still heuristic trial-and-error.
  - Wrong top-level interface?
    - Better to separate computation from interface logic?

Cryptol AES Notes

- Based on AES example provided.
  - Small modifications to allow synthesis.
- Basic “directives”
  - Parallel by default
  - seq() – generate sequential implementation
  - reg() – add pipelining registers

GCM Results

(preliminary)

AES-GCM – Vivado HLS	
FPGA Part	xc7a200tffg1156-3
LUTs	12534
FFs	7134
BRAMs	0
SLICEs	4063

# cycles 	
n = PT bits
m = AAD bits
	On average: 38 cycles/128 bits

Clk Freq. (MHz)	
Throughput	
TP / Area	

Pitfalls

- Requires writing C code in peculiar ways.
  - Only call functions from one location (or inline)
  - “Arbitrary Precision” Types
    - Code is cleaner.
    - Less portable.
      - Write repetitive loops to keep code portable.
      - No GDB support in Vivado HLS.
  - Difficult to control/understand cycle scheduling.

Thank you.

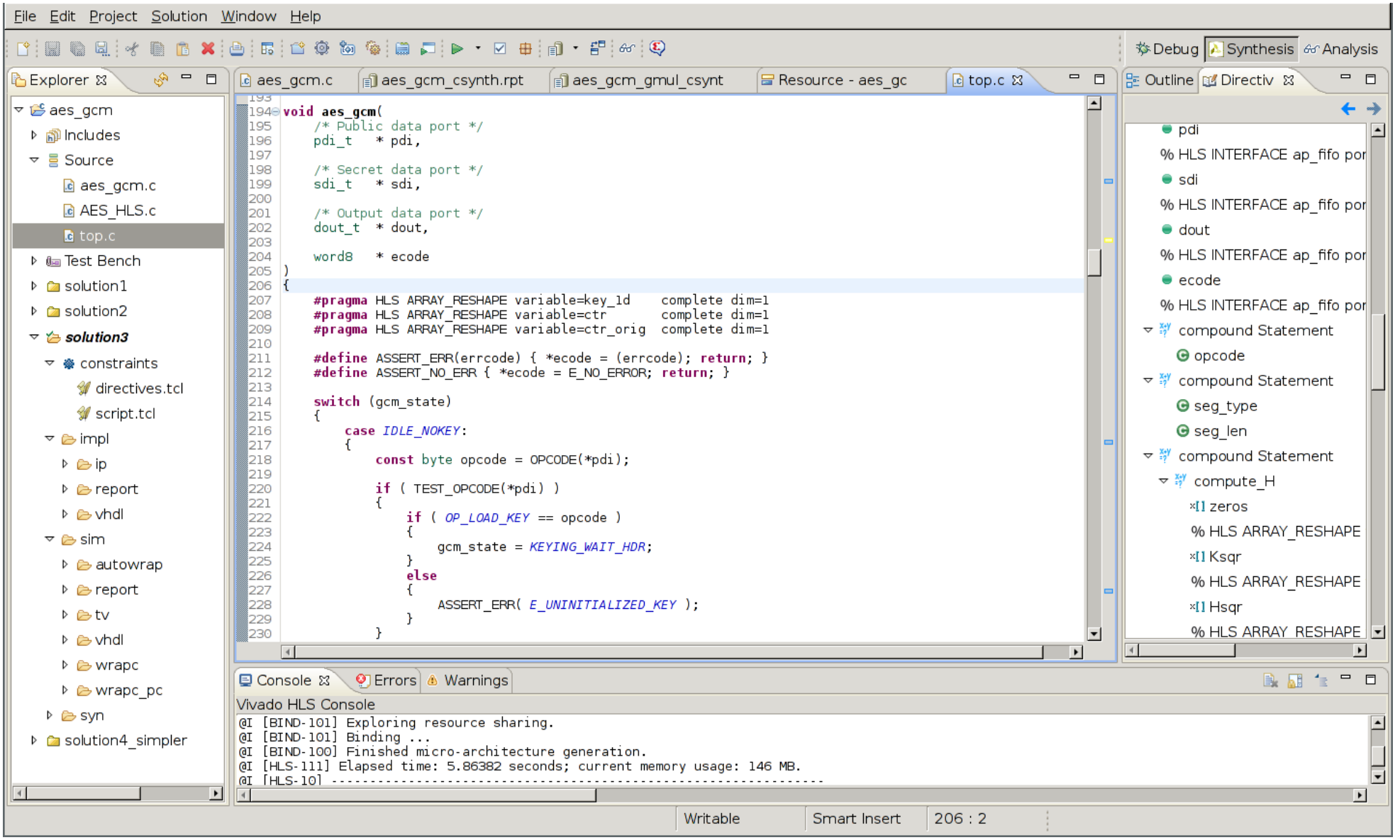
Questions?

- Special Thanks to:
  - Ekawat Homsirikamol (“Ice”) for providing HLS-ready AES implementation.
  - Adam Foltzer of Galois, Inc. for providing academic license to Cryptol 1.x.

Sample Cryptol Code

```
encrypt       : (KS(Nr), [128]) -> [128];
encrypt (xkey, PT) = join (reverse (blockEnc (xkey, reverse (split PT))));

  type KS(Nr) = ([4][4*8], [Nr-1][4][4*8], [4][4*8]);

  blockEnc (XK, PT) = unstripe (Rounds (Round, State, XK))
    where {
      State : [4][4*8];
      State = stripe PT;
    };

  stripe : [4*4][8] -> [4][4*8];
  stripe block = [| join b || b <- split block |];

  unstripe : [4][4*8] -> [4*4][8];
  unstripe state = join [| split s || s <- state |];

  Rounds (Next, State, (initialKey, rndKeys, finalKey))
    = final
    where {
      istate = State ^ initialKey;
      rnds = [istate] # [| Next (state, key, False)
                         || state <- rnds
                         || key <- rndKeys |];
      final = Next (last rnds, finalKey, True);
    };

  InvMixColumns : [4][4*8] -> [4][4*8];
  InvMixColumns rows = [| join x || x <- [[s00' s10' s20' s30'] [s01' s11' s21' s31']
                                          [s02' s12' s22' s32'] [s03' s13' s23' s33']] |]
   where {
      [[s00 s10 s20 s30] [s01 s11 s21 s31] [s02 s12 s22 s32] [s03 s13 s23 s33]] = [| (split r) : [4][8] || r <- rows
|];
      s00' = (mE s00) ^ (mB s10) ^ (mD s20) ^ (m9 s30);
      s10' = (m9 s00) ^ (mE s10) ^ (mB s20) ^ (mD s30);
...
```

Vivado HLS


File Edit Project Solution Window Help
Debug Synthesis Analysis
Explorer
aes_gcm
Includes
Source
aes_gcm.c
AES_HLS.c
top.c
Test Bench
solution1
solution2
solution3
constraints
directives.tcl
script.tcl
impl
ip
report
vhdl
sim
autowrap
report
tv
vhdl
wrapc
wrapc_pc
syn
solution4_simpler
aes_gcm.c
aes_gcm_csynth.rpt
aes_gcm_gmul_csynt
Resource - aes_gc
top.c
void aes_gcm(
/* Public data port */
pdi_t * pdi,
/* Secret data port */
sdi_t * sdi,
/* Output data port */
dout_t * dout,
word8 * ecode
)
{
#pragma HLS ARRAY_RESHAPE variable=key_1d complete dim=1
#pragma HLS ARRAY_RESHAPE variable=ctr complete dim=1
#pragma HLS ARRAY_RESHAPE variable=ctr_orig complete dim=1
#define ASSERT_ERR(errcode) { *ecode = (errcode); return; }
#define ASSERT_NO_ERR { *ecode = E_NO_ERROR; return; }
switch (gcm_state)
{
case IDLE_NOKEY:
{
const byte opcode = OPCODE(*pdi);
if (TEST_OPCODE(*pdi))
{
if (OP_LOAD_KEY == opcode)
{
gcm_state = KEYING_WAIT_HDR;
}
else
{
ASSERT_ERR(E_UNINITIALIZED_KEY);
}
}
Outline
Directiv
pdi
% HLS INTERFACE ap_fifo por
sdi
% HLS INTERFACE ap_fifo por
dout
% HLS INTERFACE ap_fifo por
ecode
% HLS INTERFACE ap_fifo por
compound Statement
opcode
compound Statement
seg_type
seg_len
compound Statement
compute_H
zeros
% HLS ARRAY_RESHAPE
Ksqr
% HLS ARRAY_RESHAPE
Hsqr
% HLS ARRAY_RESHAPE
Console
Errors
Warnings
Vivado HLS Console
@I [BIND-101] Exploring resource sharing.
@I [BIND-101] Binding ...
@I [BIND-100] Finished micro-architecture generation.
@I [HLS-111] Elapsed time: 5.86382 seconds; current memory usage: 146 MB.
@I [HLS-10] ----------
Writable
Smart Insert
206 : 2

GCM Results

- LegUp
  - Able to compile to Verilog, but not able to verify correctness or get synthesis/performance results.

AES Results

AES-128 – Cryptol	
FPGA Part	xc6vlx75t-3ff784
LUTs	2093
FFs	1891
BRAMs	96
SLICEs	1485
# cycles (per 128-bit encryption)	20
Clk Freq. (MHz)	250.0
Throughput (Mbits/sec)	1600
TP / Area	1.08