ECE 385 – Digital Systems Laboratory

Lecture 21 – Final Exam Review

Zuofu Cheng

Fall 2018

Link to Course Website

ECE ILLINOIS

ILLINOIS

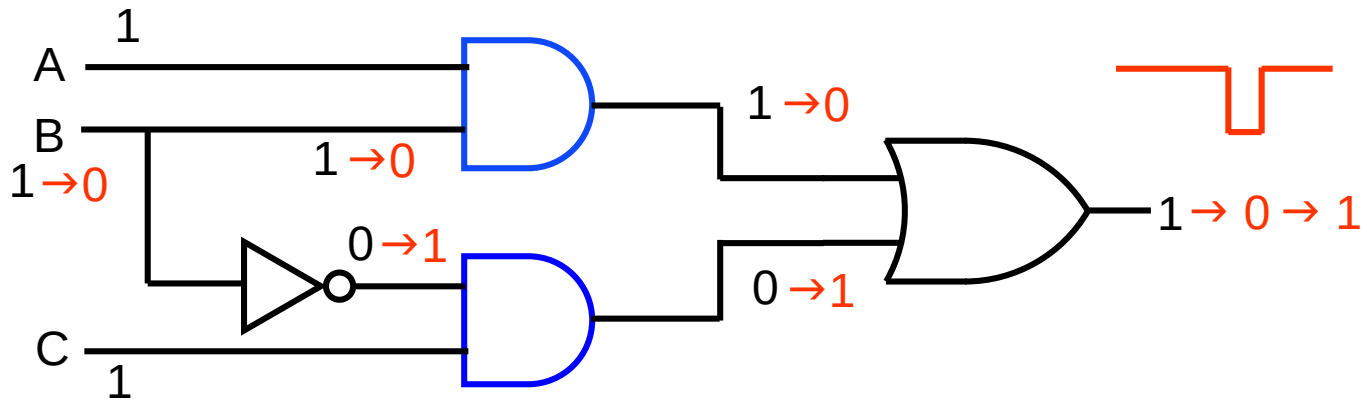
Final Exam Announcements

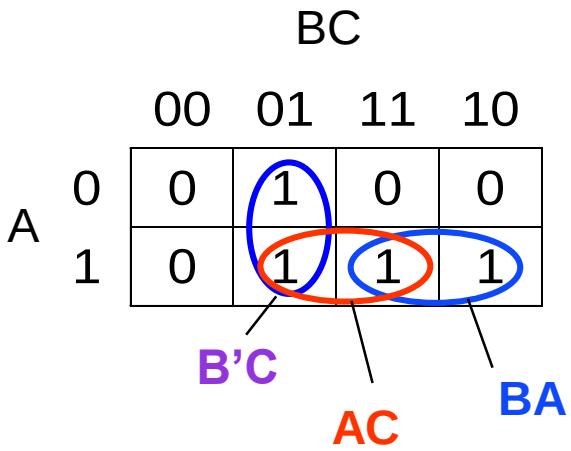
- Final Exam on 4/14 in class (Wednesday, 4:00 PM)
- 30 multiple choice questions, 40 minutes, closed notes
- Exam is comprehensive, but only ~5 questions explicitly on earlier labs
- Most questions will be on labs 7-9 and general SystemVerilog, SoC, IP, and other digital design questions.
- No Lectures or Q/A sessions after Wednesday, but office hours will still go on
- Bring your ID to turn in exam
- Rooms 1002, 2017 ECEB (by Last Name)
  - 1002 ECEB: A – S
  - 2017 ECEB: T – Z

General Studying Strategies

- Make sure you can sketch out the block diagrams for labs 1-9 (exception for lab 6, since block diagram is complex)
  - This is the most important point
  - Exam will test to make sure you understand labs as an individual student
  - Also make sure you can write a short (couple sentence) description for each module
- In addition, make sure you understand how the "core algorithm" in each lab works
  - Adders in Lab 4
  - Multiplication in lab 5
  - AES encoding in Lab 9 week 2…
- If lab has a state machine, know generally what the states are and do

Experiment 1: Static Hazards



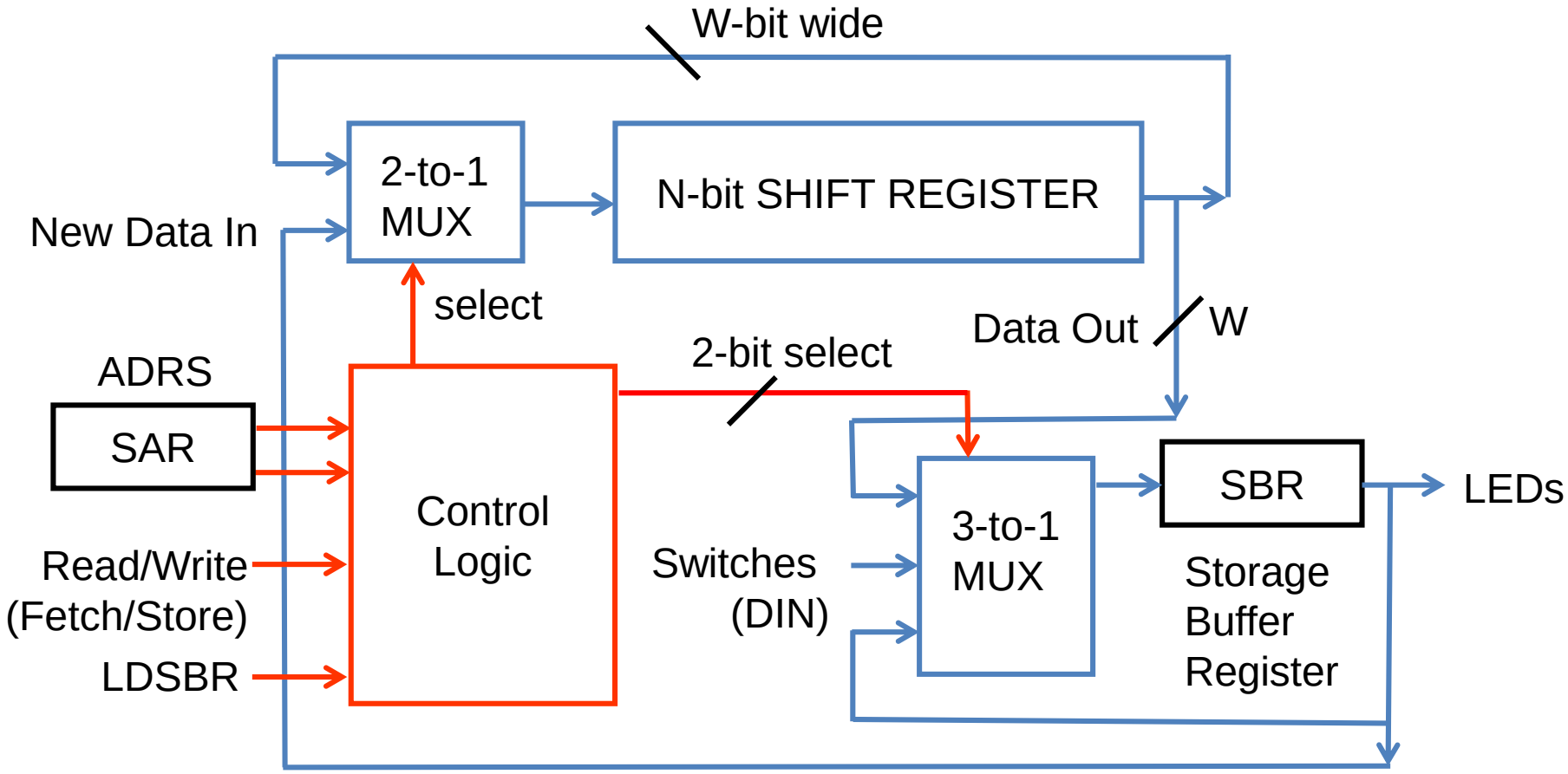


To avoid Static-1 hazard in an AND-OR (SOP) circuit, cover all adjacent min-terms in the K-map.
In this example, add the term AC

- What is the final Boolean function for glitch-free circuit?
  - Z = B'C + BA + AC

Experiment 2: Data Storage with TTL

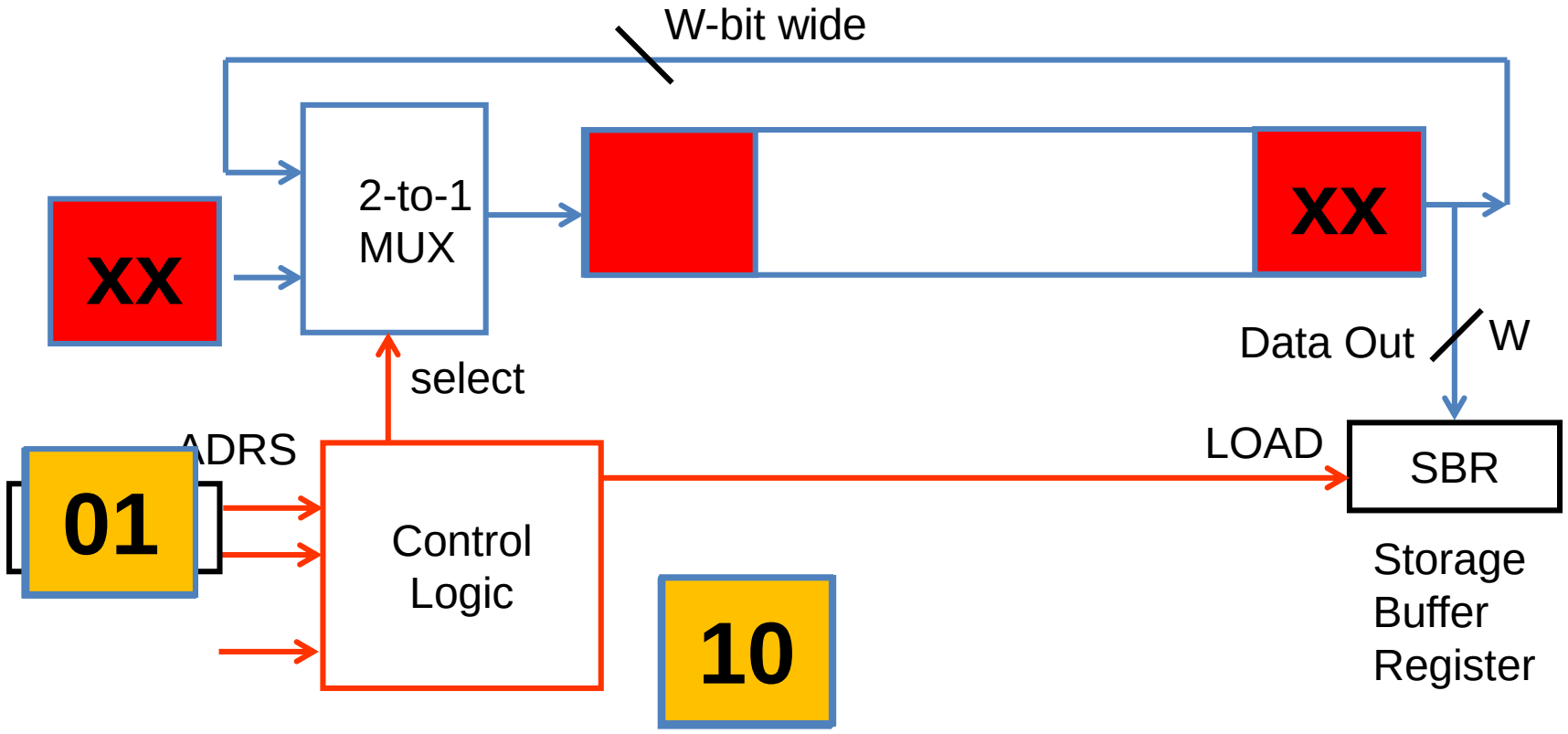
Use two 74LS194 shift-registers
Serial Input and Output
Circulating data

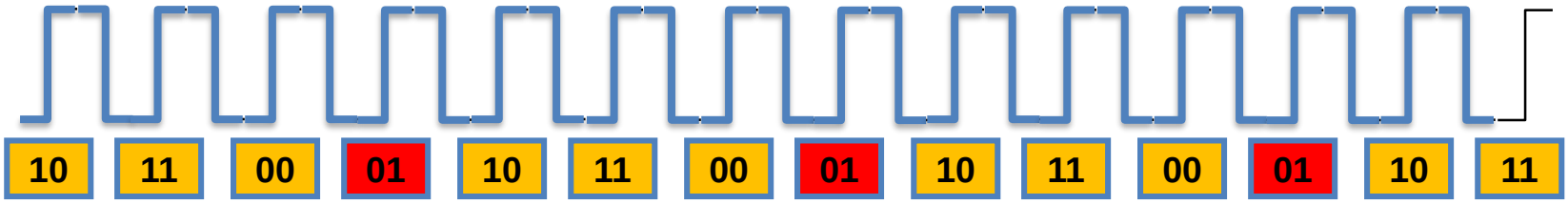


Experiment 2: Inputs and Outputs

- FETCH (Switch)
  - When FETCH is high, the value in the data word specified by the SAR is read into the SBR.
- STORE (Switch)
  - When STORE is high, the value in the SBR is stored into the word specified by the SAR.
- SBR1, SBR0 (Flip-flops)
  - The data word in the SBR; either the most recently fetched data word or a data word loaded from switches
- SAR1, SAR0 (Switches)
  - The address, in the SAR, of a word in the storage
- DIN1, DIN0 (Switches)
  - Data word to be loaded into SBR for storing into storage
- LDSBR (Switch)
  - When LDSBR is high, the SBR is loaded with the data word DIN1, DIN0

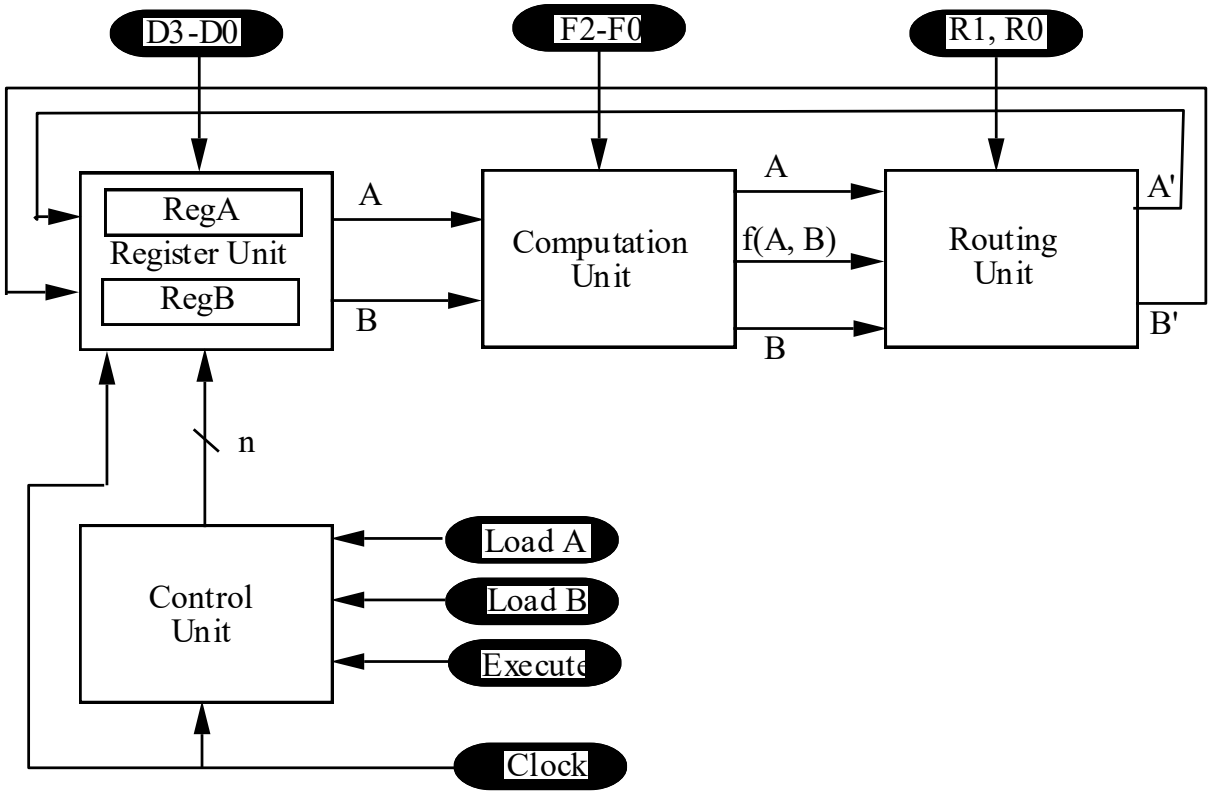
Experiment 2: Operation of Circuit





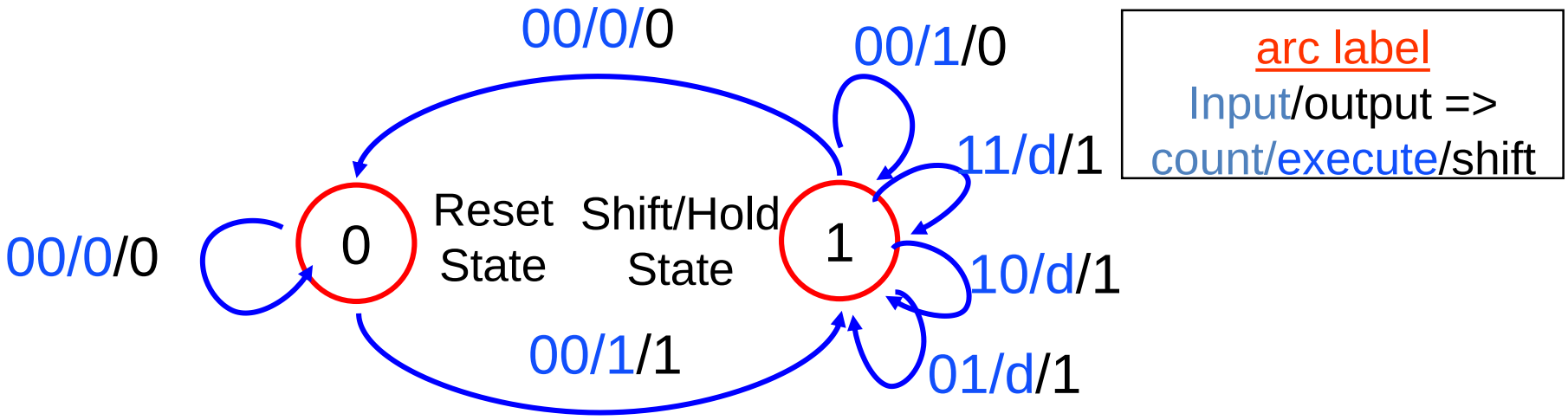
Experiment 3: 4-bit Serial Processor

- A Bit-Serial Logic Processor
  - Two 4-bit registers A, B
  - A or B also serve as a destination register.
  - E.g., AND A, B, A /* A AND B=>A */



Experiment 3: State Machine (Mealy)

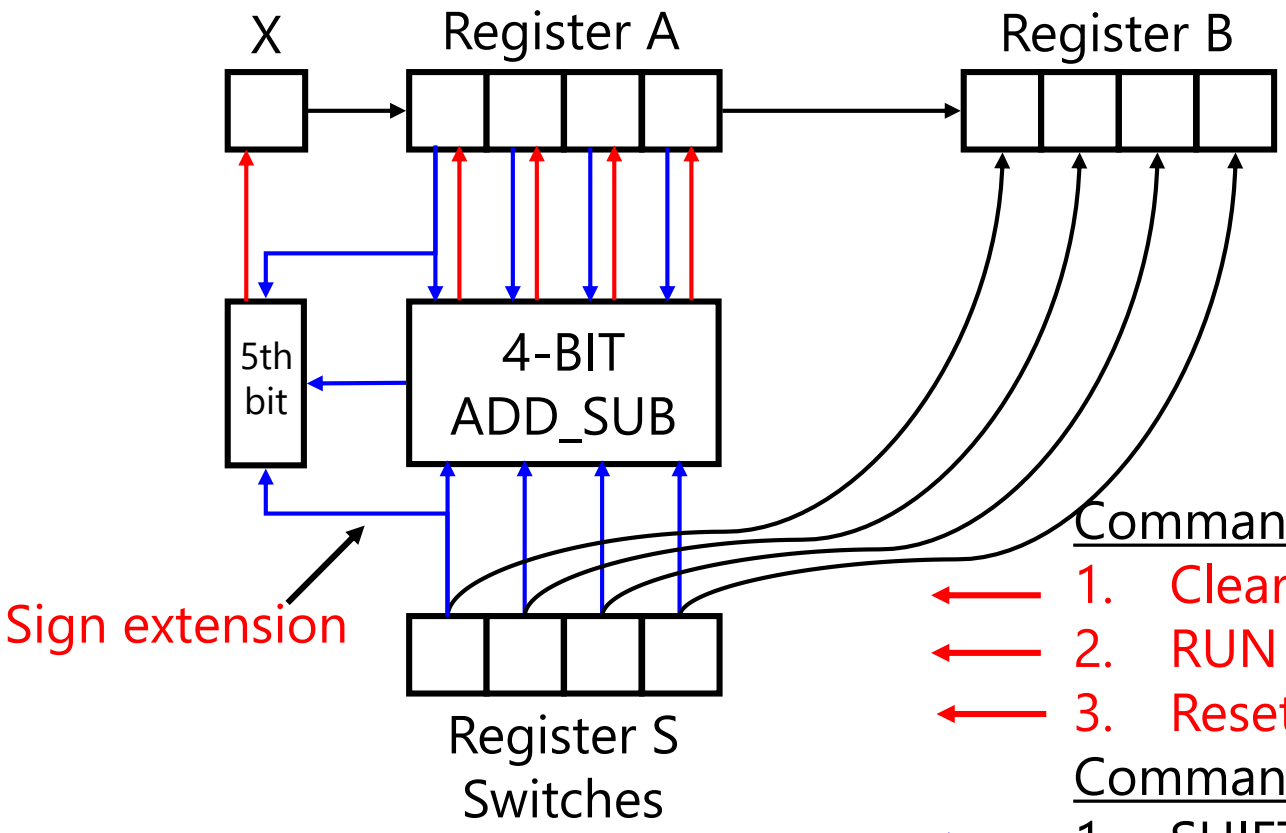
- Arcs are labeled with both inputs & outputs
- Fewer states than Moore machine (outputs depend on both state and inputs)



Experiment 4: Adders in SystemVerilog

- Experiment 4 had 3 different types of adders
  - Carry ripple adder
  - Carry look-ahead adder
  - Carry select adder
- Know how they work and what are tradeoffs
  - Carry ripple adder – smallest and simplest
  - Carry look-ahead adder – generate P and G bits beforehand to predict carry
  - Carry select adder – pre-compute both carry cases and select correct one
- Know block diagrams for all 3 adders

Experiment 5: Multiplier in SV



Commands (push buttons)

1. ClearXA_loadB
2. RUN (Execute)
3. Reset

Commands (control)

1. SHIFT XAB
2. ADD (A+S → XA)
3. SUB (A−S → XA)

SHIFT, ADD *and* SUB *are derived signals from State Controller. They last only one clock cycle*

Experiment 5: Example

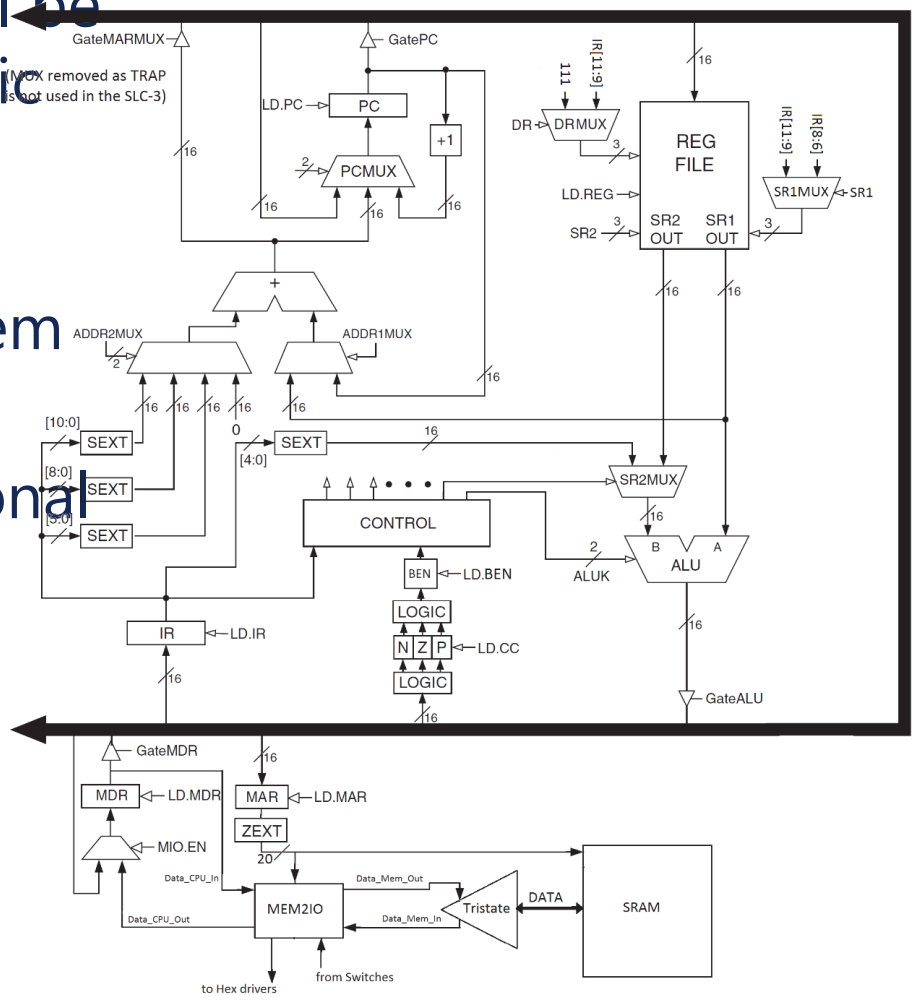
Multiply 0000 0111 × 1100 0101

Function	X	A_{7-0}	B_{7-0}	M	Next Step in words
ClearA, LoadB	0	0000 0000	*11000101*	B_0 1	Since M = 1, multiplicand (available from switches S) will be added to A.
ADD	0	0000 0111	*11000101*	1	Shift XAB by one bit after ADD complete
SHIFT	0	0000 0011	1 *1100010*	0	Do not add S to A since M = 0. Shift XAB.
SHIFT	0	0000 0001	11 *110001*	1	Add S to A since M = 1.
ADD	0	0000 1000	11 *110001*	1	Shift XAB by one bit after ADD complete
SHIFT	0	0000 0100	011 *11000*	0	Do not add S to A since M = 0. Shift XAB.
SHIFT	0	0000 0010	0011 *1100*	0	Do not add S to A since M = 0. Shift XAB.
SHIFT	0	0000 0001	00011 *110*	0	Do not add S to A since M = 0. Shift XAB.
SHIFT	0	0000 0000	100011 *11*	1	Add S to A since M = 1
ADD	0	0000 0111	100011 *11*	1	Shift XAB by one bit after ADD complete
SHIFT	0	0000 0011	1100011 *1*	1	Subtract S from A since 8th bit M = 1.
SUB	1	1111 1100	1100011 *1*	1	Shift XAB after SUB complete
SHIFT	1	1111 1110	01100011	1	8th shift done. Stop. 16-bit Product in AB.

Experiment 6: SLC-3 CPU

- Because block diagram and state machine are complicated, they will be provided if required for any specific questions
- Instruction encoding will also be provided if necessary to do problem (as on Midterm)
- Make sure you understand additional components you wrote
  - Gate XYZ and MUXes
  - NZP, BEN
  - Sign extension
  - IR, register file, PC, ALU
  - Mem2IO

SLC-3 Updated Datapath for ECE 385

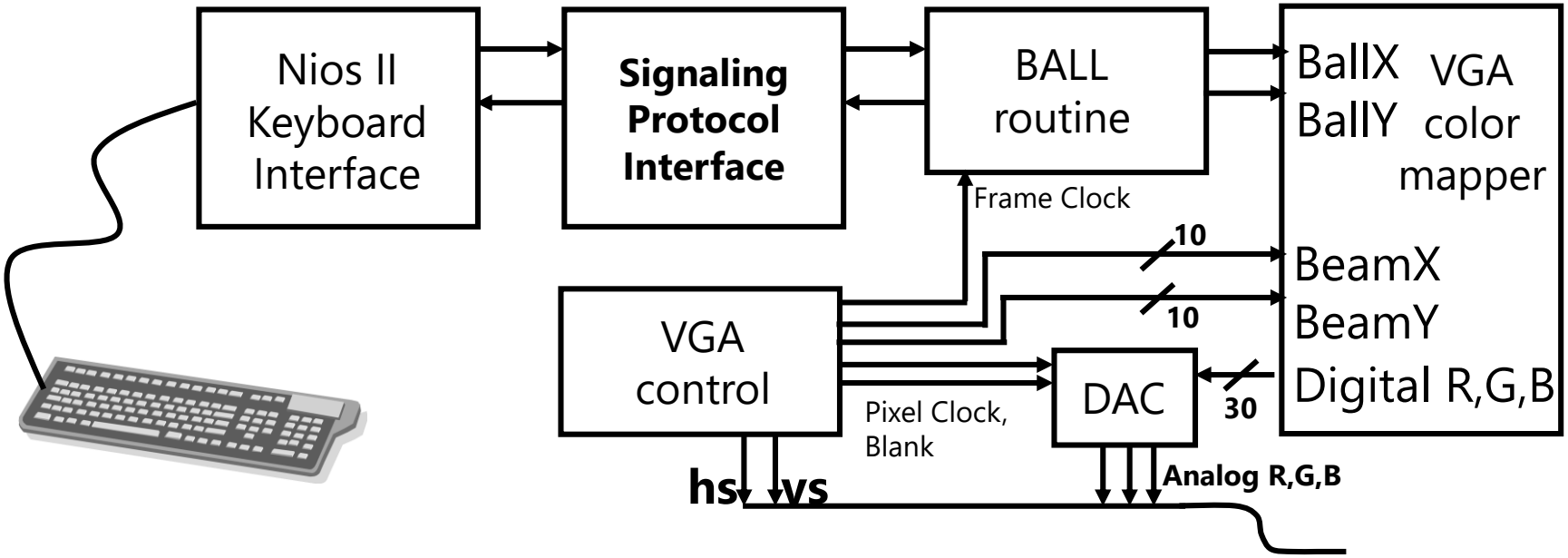


Experiment 7: Basic SoC

- Understand function of each device in Q-Sys configuration used for Lab 7
- Understand purpose of each of the steps in SoC tutorial
- Know how to read SDRAM datasheet for settings (necessary pages will be provided if specific details are required)
- Know basic embedded systems operations in C (bitwise operations, going to and from hex to binary, etc…)
- Know some basics about timing constraints (what are setup times, hold times, how is Fmax determined)

Experiment 8: SoC with USB and VGA

- Understand block diagram, what each block does, what each signal does



Ball routine: partially given

Color mapper: given

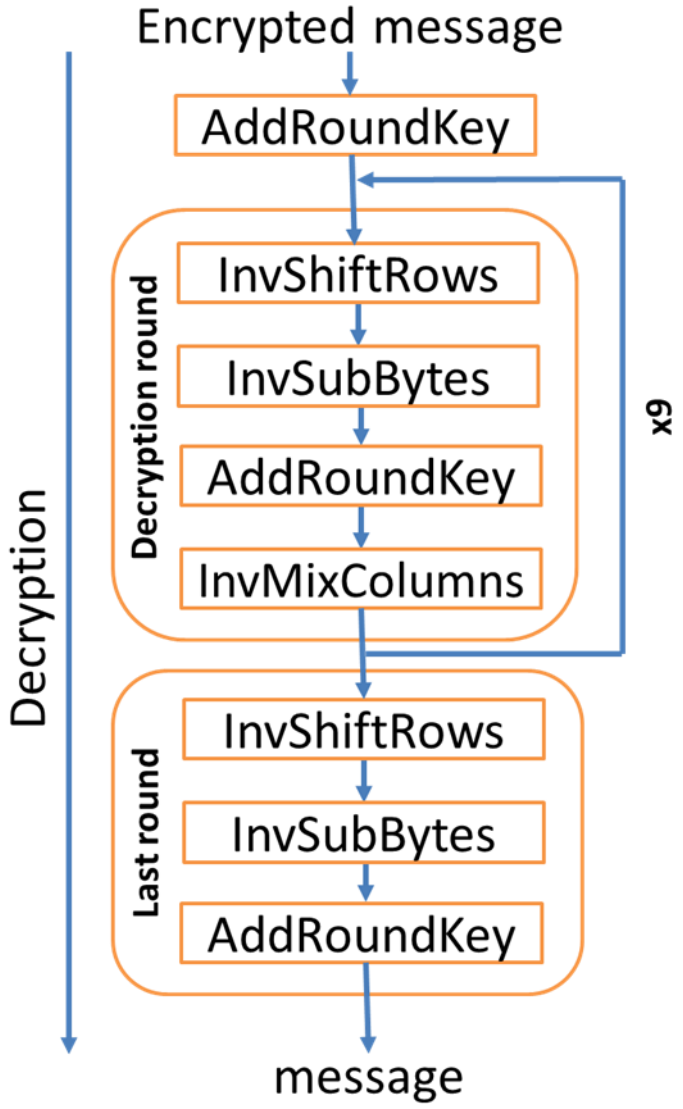
VGA controller: given

Experiment 8: USB and VGA

- Review changes we made to NIOS II configuration and why
  - What do the added modules do?
- Must know about specific changes needed to get USB working (code you filled in in USB interface module)
- Not necessary to understand details of USB protocol (mostly abstracted in the EZ-OTG firmware and provided USB.C files)
- Review basics of VGA signal generation, what does a VGA signal look like timing wise? What do H and V sync signals do?

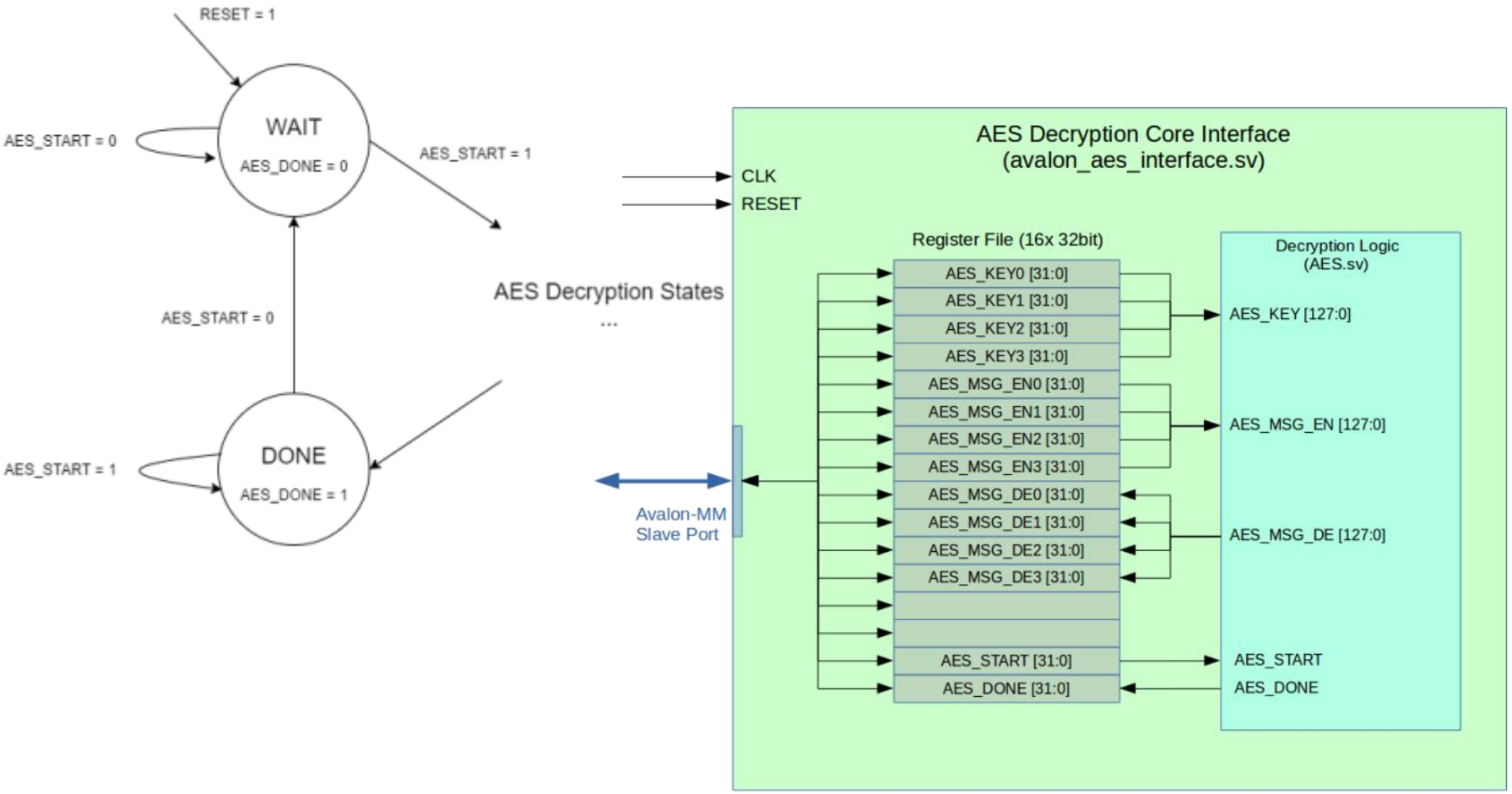
Experiment 9: AES en/decryption

- Review AES operations in a round (for both encryption/decryption)
- Understand provided look up tables for weeks 1 and 2
- Be able to describe in a paragraph the function of each operation (e.g. what does InvSubBytes do)
- Know how the provided modules work for week 2, have basic idea for what kind of hardware is inside each module
- Understand the overall state machine that you had to write for week 2
- Know differences between encryption and decryption rounds



AES Decryption Core Interface

- Review HW/SW interface in Lab 9
- Understand provided state machine states



Additional Topics

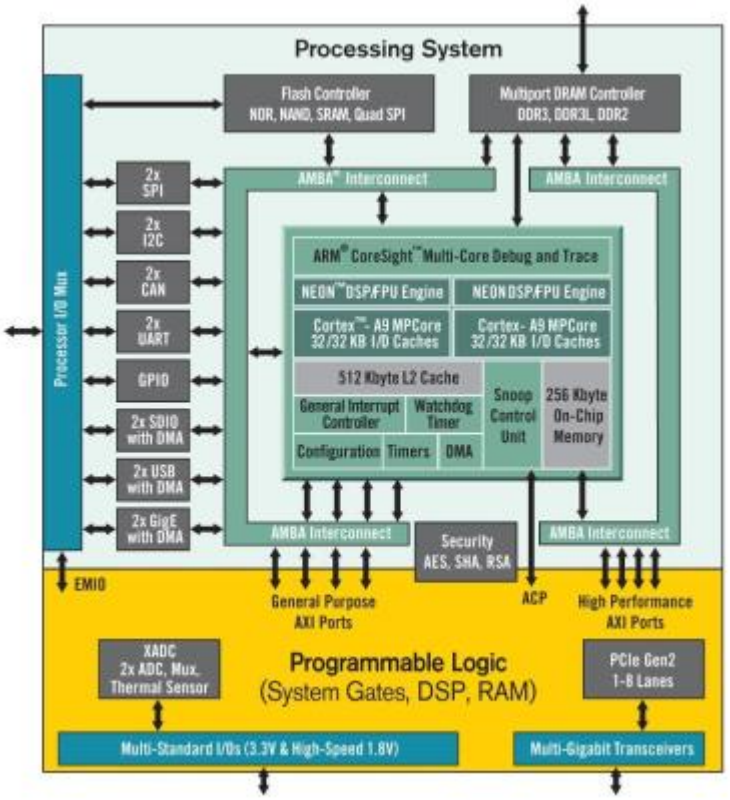
- Questions on general SystemVerilog
  - E.g. when to use each type of assignment
  - How do encode state machines
  - What are limitations of each type of always block
- Questions specific to lectures
  - Know different approaches to drawing graphics
  - Differences in each type of memory on the DE2-115
  - Basic issues with timing and synchronization (setup time, hold time, going to and from asynchronous signals)
  - SDRAM operation and structure

Beyond ECE 385 – High Level Synthesis

- Case study: Xilinx Vivado HLS
- General workflow consist of 5 steps
  - SDSoC
    - Automatic Hardware-Software partitioning
  - HLS
    - High level C/C++ synthesis
  - Logic Synthesis
  - Placement and Routing
  - Bitstream generation
- The last 3 should be familiar (Quartus in Intel/Altera, Vivado in Xilinx)

Example Platform: Zynq SoC

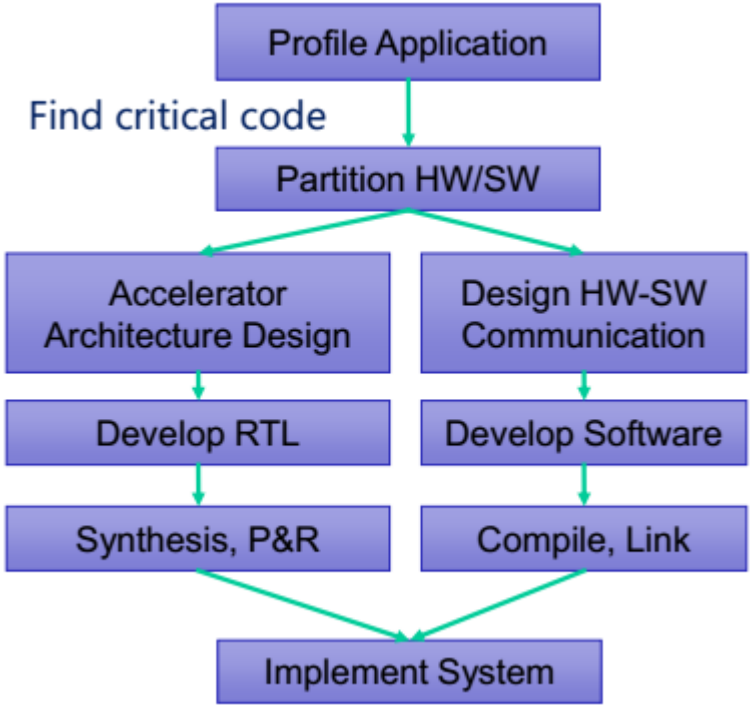
- ARM Cortex A9 core + FPGA
- FPGA accelerator operation
- Typically interfaces with bus
  - Altera uses Avalon MM
  - Xilinx uses AXI
- We've done all this in Lab 9



Zynq System:

- Case study: Xilinx Vivado HLS
- General workflow consist of 5 steps
  - SDSoC
    - Automatic Hardware-Software partitioning
  - HLS
    - High level C/C++ synthesis
  - Logic Synthesis
  - Placement and Routing
  - Bitstream generation
- The last 3 should be familiar (Quartus in Intel/Altera, Vivado in Xilinx)

Traditional Workflow



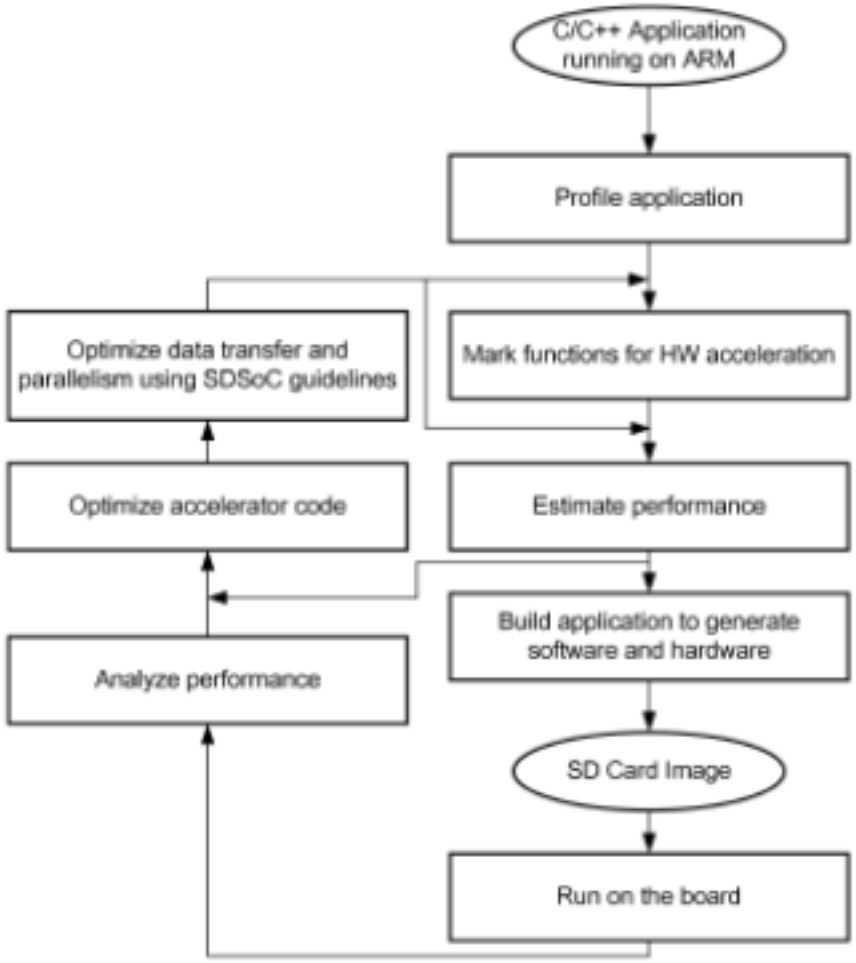

```
main()
{
 readData(a,b,c)
 preprocessData()

 for(i=0 to N)
   c[i]=a[i]+b[i]

 checkData()
}
```

SDSoC

- "Software Defined SoC"
- Development environment for software/embedded designers
- IDE that helps automate HW/SW co-design
  – Includes profiling tools
  – Performance estimation tools
- Includes a library of communication/data movers
  – Simple, Scatter-gather, 2D DMA
- System-wide performance optimization



Vivado HLS

- Generate Verilog code automatically
- Use pragmas to guide design and do design space exploration

```
void vectoradd(float
a[1000], float b[1000],
float c[1000])
{
  int i=0;
  for(i=0; i<1000; i++)
    c[i] = a[i] + b[i];
}
```

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	14
FIFO	-	-	-	-
Instance	-	2	205	390
Memory	-	-	-	-
Multiplexer	-	-	-	14
Register	-	-	135	-
Total	0	2	340	418

Latency		Interval		
min	max	min	max	Type
8001	8001	8002	8002	none

```
void vectoradd(float
a[1000], float b[1000],
float c[1000])
{
  int i=0;
  for(i=0; i<1000; i++)
  #pragma HLS UNROLL
    c[i] = a[i] + b[i];
}
```

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	-	-
FIFO	-	-	-	-
Instance	-	4	410	780
Memory	-	-	-	-
Multiplexer	-	-	-	11393
Register	-	-	1211	-
Total	0	4	1621	12173

Latency		Interval		
min	max	min	max	Type
506	506	507	507	none