



# Modeling the MSP430 in Verilog

## Project Proposal

Victoria Rodriguez

Texas Tech University

Wednesday 12<sup>th</sup> June, 2019



## 1 Project Proposal

- Proposed Project Description
- Background Information
- Proposed Rubric

## 2 Pending Design

- Designing a Pipeline Structure
- What is Already Known About MSP430

## 3 Logistics

- Gantt Chart
- Budget
- Next Week's Deliverable



## Project Proposal



The objective of the proposed project is to model the MSP430 microcontroller in Verilog. This model will utilize the same structure of the MSP430 microcontroller, which will be reverse-engineered from Texas Instruments (TI)'s descriptions of the architecture in their family guide and datasheets.



# Background Information

```
module mod_barrel_shifter #(parameter SIZE=32)
  (input  [4:0] SH, FS,
   input  [SIZE-1:0] D,
   output reg C_out ,
   output reg [SIZE-1:0] Y);

  wire [2*SIZE-1:0]
padded_shift_0 = {{SIZE{1'b0}}, D};
  wire [2*SIZE-1:0]
padded_shift_1 = {{SIZE{1'b1}}, D};
  wire [2*SIZE-1:0]
padded_rot
= {D
    , D};

  always @ (*)
  case (FS)
    5'b10000: {C_out,Y} = padded_shift_0 << SH;
    5'b10001: {Y,C_out} = padded_shift_0 >> SH-1;
    5'b10100: {C_out,Y} = padded_shift_1 << SH;
    5'b10101: {Y,C_out} = padded_shift_1 >> SH-1;
    5'b10010: Y = padded_rot >> (SIZE-SH);
    5'b10011: Y = padded_rot >> SH;
    default: Y = D;
  endcase // case (FS)
endmodule // shifter
```

- Verilog is used to verify the digital logic at the register-transfer level (RTL) of abstraction, and is considered integral to SoC design [1]
- Using the higher level (RTL) of abstraction makes the verification process more efficient. Previously, verification was done at the gate level and took much longer than it does today with Verilog [1]



- (15%) The system's interrupt handler responds accurately to the following system interrupts:
  - (10%) System Reset
  - (2.5%) Non-Maskable Interrupts
  - (2.5%) Maskable Interrupts
- (15%) The system is able to maintain a stack memory structure which reflects that of the MSP430 family.
- (15%) The system's CPU registers accurately reflect those of the MSP430 family:
  - (5%) PC
  - (5%) SP
  - (5%) SR

# Proposed Rubric (Cont.)



- (15%) The system contains a memory space structure and organization which reflects the description provided by TI on the MSP430 family, as far as the extent of this project.
- (5%) The system is able to address memory accurately with all addressing modes.
- (5%) The system's status bits are accurate for the instruction it's reflective of.
- (15%) The system's instruction set accurately reflects that of the MSP430 (27 instructions).
- (15%) The system's instructions follow the same format as the MSP430's.
- (5%) Functionality of at least one peripheral.
- (10%) Demonstration of the functionality of the pipeline via implementation of an assembly script.



Pending Design





- The RISC pipeline works by performing multiple tasks on the same clock cycle, rather than waiting for the last one to be finished [2], [3]
- Classic RISC architecture had 5 pipeline stages: instruction fetch (IF), instruction decode (ID), execute (EX), memory access (MEM), and write back (WB)

[4]



- The project first needs a pipeline structure designed.
- This design will come from reading the MSP430 Family Guide and reverse engineering the RISC architecture, under the guidance of Dr. Nutter.
- Most of the project's design plan relies on this structure.

# MSP430 Architecture

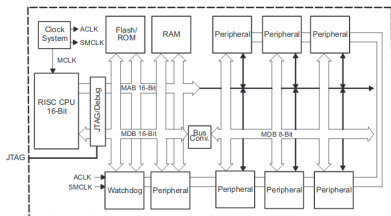


Figure 1-1. MSP430 Architecture

Texas Instruments [5]

- Uses a RISC architecture (16-bits)
- TI claims a large register file reduces bottleneck [5], presumably referring to the von-Neumann bottleneck [6]
- Addressable memory space is 128 kB.

# MSP430 Address Space



- The von-Neumann style architecture has one address space for SFRs, peripherals, RAM, Flash/ROM, etc [5]
- This means system is subject to famous von-Neumann bottleneck, because instruction fetch and data access cannot occur on the same clock cycle [6]

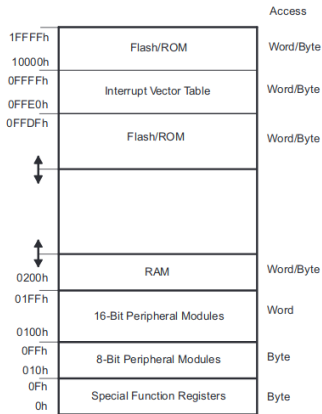


Figure 1-2. Memory Map

Texas Instruments [5]

# MSP430 Instruction Set Quirks



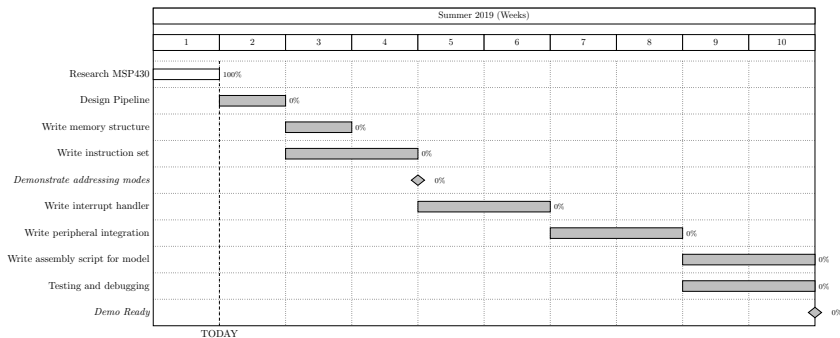
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Double-Operand (Format I)															
Op-code				S-Reg				Ad	B/W	As	D-Reg				
Single-Operand (Format II)															
Op-code								B/W		Ad	D/S-Reg				
Jumps															
Op-code				C			10-bit PC Offset								

- The instructions come in 3 different formats [5]
- Each format has a different size op-code
- There doesn't appear to be a way to identify that an instruction is of a certain format, so the pipeline will need to account for every format type



## Logistics

# Gantt Chart



This Gantt chart is subject to change after the pipeline has been designed!



- Labor
  - \$25/hr at 20 hr/wk  $\times$  10 weeks = \$5000
- Misc.
  - MSP430 (\$30)
- Total is \$5030



# Next Week's Deliverable



Design the pipeline.



## 1 Project Proposal

- Proposed Project Description
- Background Information
- Proposed Rubric







## 2 Pending Design

- Designing a Pipeline Structure
- What is Already Known About MSP430

## 3 Logistics

- Gantt Chart
- Budget
- Next Week's Deliverable



-  R. Dekker, *What's the difference between vhdl, verilog, and systemverilog?* [Online](#), 2014.
-  A. Louri, *Lecture 3: Introduction to pipelining, structural hazards, and forwarding*, [Online](#), University of Arizona.
-  E. Roberts, *How pipelining works*, [Online](#), <https://stanford.io/2IAMW0h>.
-  *Five stage pipeline*, [Online](#), <https://commons.wikimedia.org/wiki/File:Fivestagespipeline.png>.
-  *MSP430x2xx Family User's Guide*, [Texas Instruments](#).
-  J. Backus, "Can programming be liberated from the von neumann style?: A functional style and its algebra of programs," *Commun. ACM*, vol. 21, no. 8, pp. 613–641, Aug. 1978, ISSN: 0001-0782. DOI: 10.1145/359576.359579. [Online]. Available: <http://doi.acm.org/10.1145/359576.359579>.