EECS 373

Design of Microprocessor-Based Systems

Robert Dick

University of Michigan

Lecture 5: Memory-mapped I/O, APB

20 September 2017

Review

- Power-on reset
- Pickmin example
- IP register and veneers
- Memory mapped IO

Comments

- Typically high variance
  - Homework grades
  - Exam grades
- Typically low variance
  - Lab grades
  - Project grades (with the exception of 0-2 teams)
- At a minimum, skim practice exams

Outline

- Memory-mapped IO
- Design and debugging
- Stack use for parameter passing
- Project problem selection
- Start on Advanced Peripheral Bus

Example

```
#include <stdio.h>
#include <inttypes.h>

#define REG_FOO 0x40000140

int main(void) {
  uint32_t *reg = (uint32_t *)(REG_FOO);
  *reg += 3;

  print_uint(*reg);
  return 0;
}
```

“*reg += 3” is turned into a ld, add, str sequence

- Load instruction
  - A bus read operation commences
  - The CPU drives the address “reg” onto the address bus
  - The CPU indicated a read operation is in process (e.g., R/W#)
  - Some “handshaking” occurs
  - The target drives the contents of “reg” onto the data lines
  - The contents of “reg” is loaded into a CPU register (e.g., r0)
- Add instruction
  - An immediate add (e.g., add r0, #3) adds three to this value
- Store instruction
  - A bus write operation commences
  - The CPU drives the address “reg” onto the address bus
  - The CPU indicated a write operation is in process (e.g., R/W#)
  - Some “handshaking” occurs
  - The CPU drives the contents of “r0” onto the data lines
  - The target stores the data value into address “reg”

Outline

- ~~Memory-mapped IO~~
- Design and debugging
- Stack use for parameter passing
- Project problem selection
- Start on Advanced Peripheral Bus

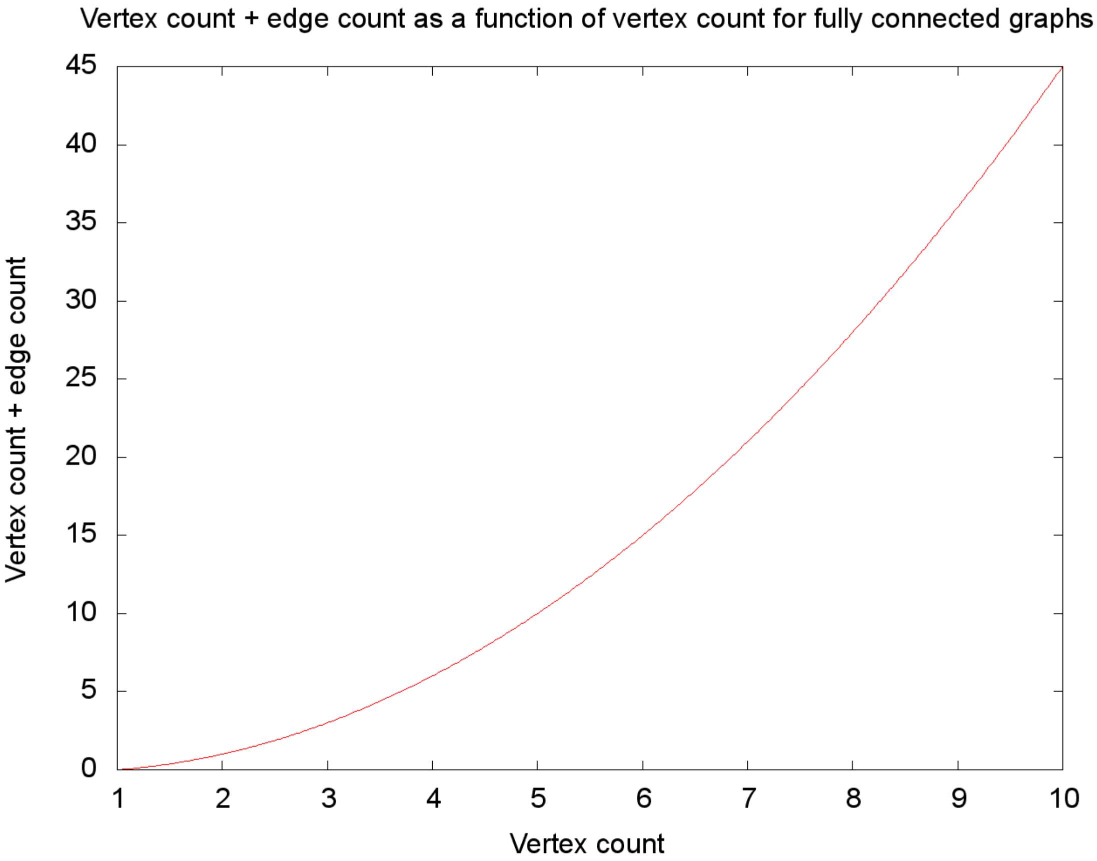
Design and debugging: computer systems are graphs

- Listen close. This is quick and seems simple but if you understand it, it will change your life.

- A computer system is a graph.
- Each component, e.g., a line of code or transistor, is a vertex (v).
- Each effect that influences other components is an edge (e).
- Complexity is a function of $|v| + |e|$.

Graph sizes

- For undirected fully connected graphs
  - |e| = |v|(|v| – 1) / 2
- But it's much worse than that because your ability to analyze systems decreases dramatically with system size.
- So system complexity (debug time) is a superlinear function of |v|, like $|v|^k$
- k is generally >= 2, and probably quite a bit bigger.



Design and debugging: how to make your life easy and make your embedded systems work

- Control |v|
  - Get a very simple version of the system tested and functioning and add to it in small pieces, testing after each addition.
  - Never build something big and then start testing.
- Control |e|
  - Build and test isolated, side-effect free components with narrow and easy-to-understand interfaces.

Debugging process

- Search process.
- Large search space.
- Probing specific locations is expensive.
- Initial conditions.
  - Don't have the data necessary to understand the problem.
  - Haven't done the analysis necessary to convert those data to information.
- Don't neglect either weakness. Iterate.
  - Conduct naïve experiments to gather information.
  - Stop testing and reason about problem, using conclusions to devise additional tests.
- Most engineers are better at analysis or testing.
  - Don’t stay under the streetlight.

Outline

- ~~Memory-mapped IO~~
- ~~Design and debugging~~
- Stack use for parameter passing
- Project problem selection
- Start on Advanced Peripheral Bus

Outline

- ~~Memory-mapped IO~~
- ~~Design and debugging~~
- ~~Stack use for parameter passing~~
- Project problem selection
- Start on Advanced Peripheral Bus

Identifying important problems

- The easy, luck-dependent way
  - Pick a problem that you understand deeply and that is very important to you.
  - Only works if many people are similar to you.
  - That's rare, especially for engineers.
- The easy, risky way
  - Take the problem from a competitor.
- The hard, more reliable way
  - Customer discovery interview process
  - Counterintuitive. Hard.
  - Social pressures undermine the process.

How you see your idea

How others see your idea

How others see you

That's a great idea!

- That's a great idea!
  - I don't want to hurt your feelings you cute little engineer.
- I would definitely use that!
  - There are almost no circumstances in which I would use that, let alone consider paying for or supporting it.
- I have some ideas on how to make it better!
  - I'm pretending to be someone who would use it and leading you down a false path.
- Follow your dreams man!
  - Squander your life, man!

If you want to follow the reliable path

- Talking to Humans by Giff Constable
- The Startup Owner's Manual by Steve Blank and Bob Dorf
- Market Research on a Shoestring by Naeem Zafar
- Steve Blank's online videos (steveblank.com)
- Contact me for more

What do they teach?

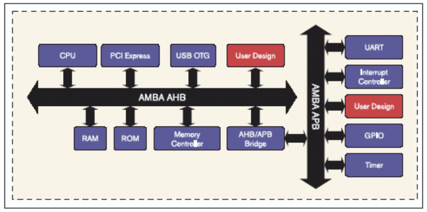
- How to design, sequence, and deliver questions to minimize bias (both yours and theirs)
- How to talk with strangers.
- This is ~~un~~ preternatural. Most must learn it.

Outline

- ~~Memory-mapped IO~~
- ~~Design and debugging~~
- ~~Stack use for parameter passing~~
- ~~Project problem selection~~
- Start on Advanced Peripheral Bus

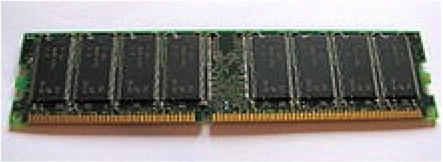
Details of the bus "handshaking" depend on the particular memory/peripherals involved

- SoC memory/peripherals
  - AMBA AHB/APB

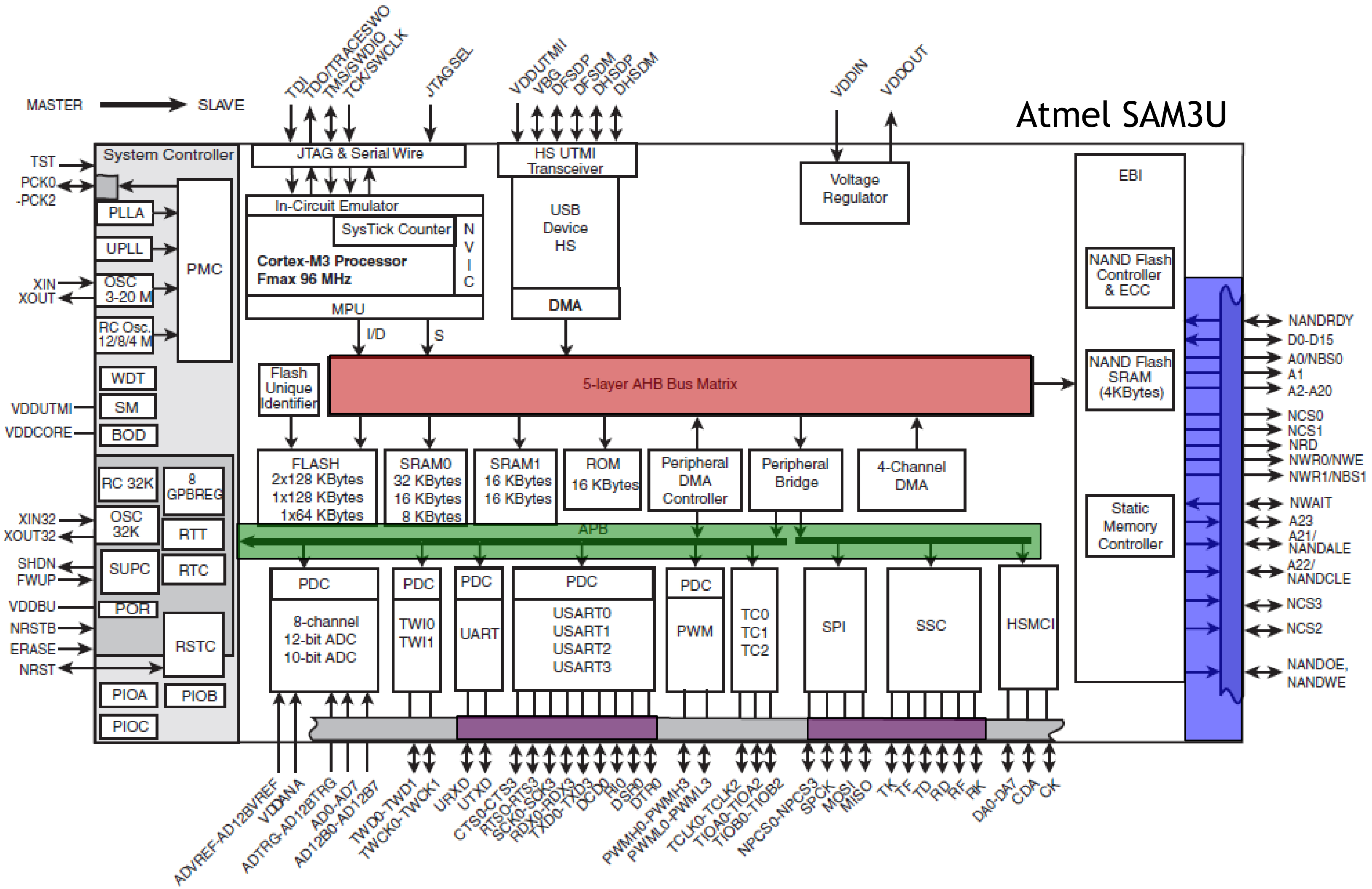


- NAND Flash
  - Open NAND Flash Interface (ONFI)

- DDR SDRAM
  - JEDEC JESD79, JESD79-2F, etc.

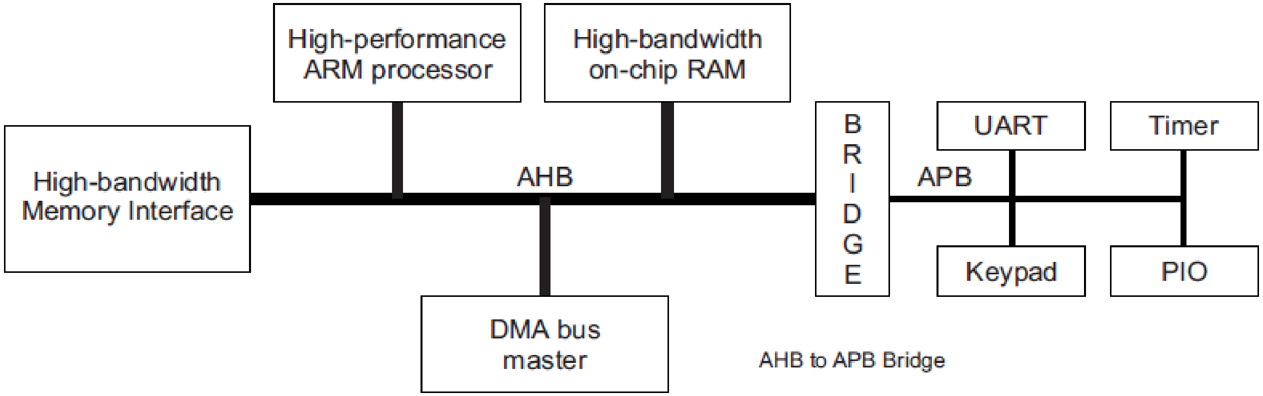


Modern embedded systems have multiple busses



Atmel SAM3U

Advanced Microcontroller Bus Architecture (AMBA)
- Advanced High-performance Bus (AHB)
- Advanced Peripheral Bus (APB)



High-performance ARM processor
High-bandwidth on-chip RAM
High-bandwidth Memory Interface
AHB
DMA bus master
B R I D G E
APB
UART
Timer
Keypad
PIO

AHB to APB Bridge

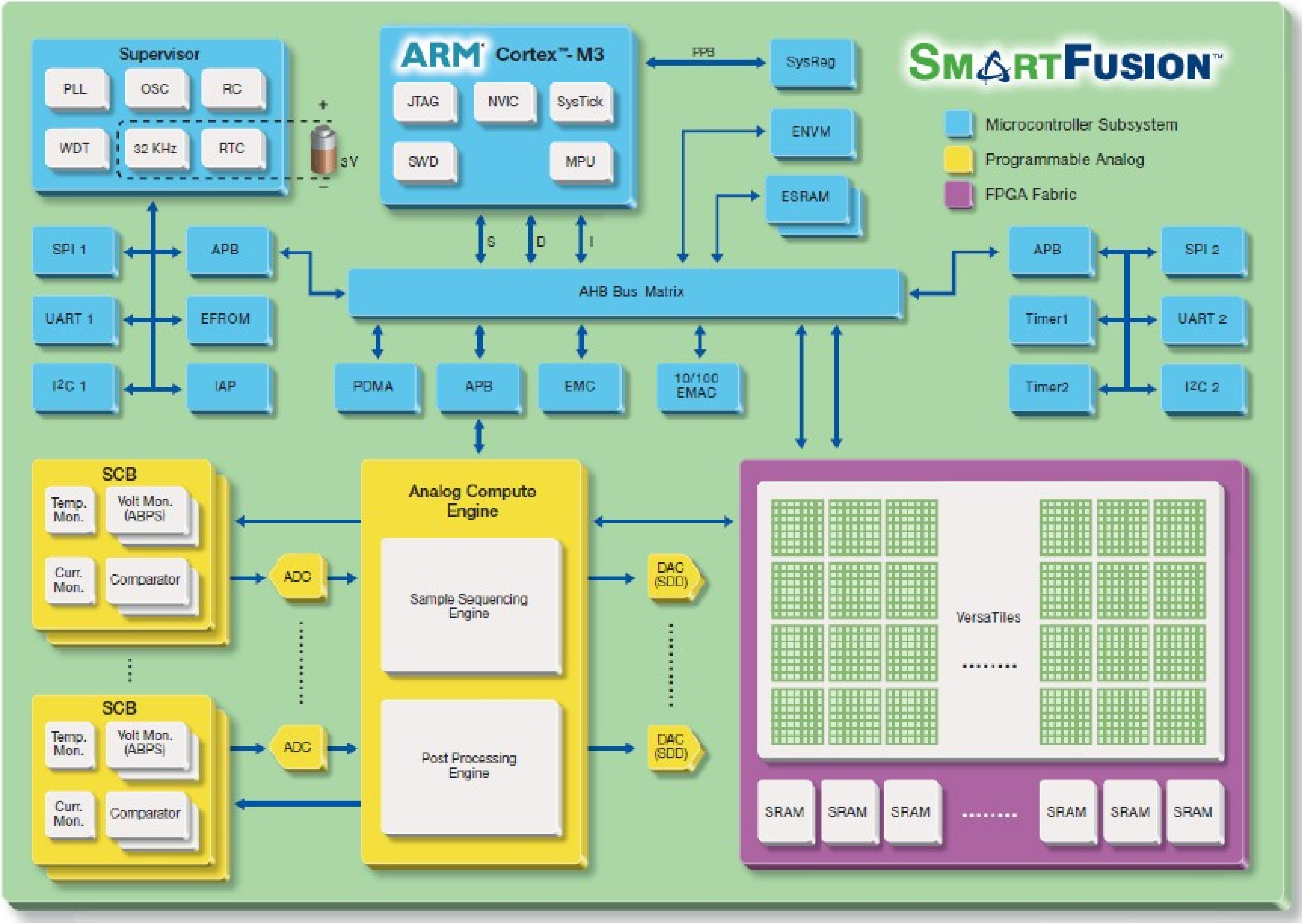
AHB

- High performance
- Pipelined operation
- Burst transfers
- Multiple bus masters
- Split transactions

APB

- Low power
- Latched address/control
- Simple interface
- Suitable of many peripherals

Actel SmartFusion system/bus architecture



Bus terminology

Transactions have “*initiators*” and “*targets*”

- Only “*bus masters*” can be initiators.
  - In many cases there is only one bus master (*single master* vs. *multi-master*).
- Slave devices can only be targets. They can't start transactions.
- Some wires might be shared among all devices while others might be point-to-point connections (generally connecting the master to each target).

Driving shared wires

- Some shared wires might need to be driven by multiple devices.
- In that case, we need a way to allow one device to control the wires while the others “stay out of the way”
- Most common solutions are
  - tri-state drivers
  - open-collector connections

Another option: avoid shared wires

- Expensive.
- Problematic when connecting chips on a PCB as you are paying for pins and wiring area.
- Quite doable (though not pretty) inside of a chip.

Wire count

- Consider a single-master bus with 5 other devices connected and a 32-bit data bus.
- Shared bus → 32 pins
- Separate buses
  - Each slave would need _____ pins for data
  - The master would need _____ pins for data
- Pins and wiring area cost money.

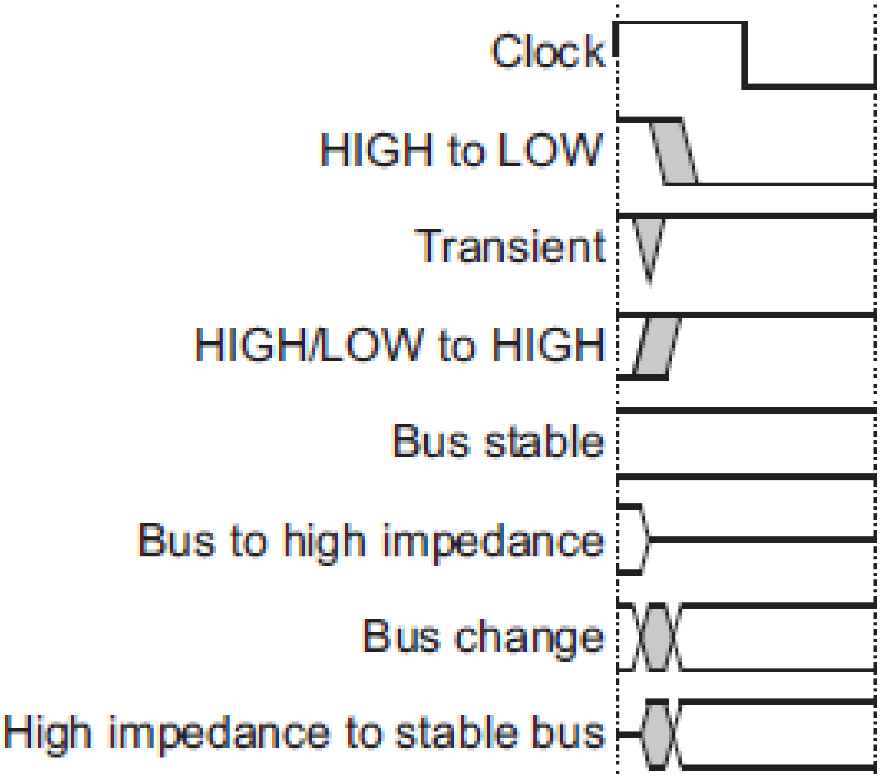
APB is designed for ease of use

- Low-cost
- Low-power
- Low-complexity
- Low-bandwidth
- Non-pipelined
- Ideal for peripherals

Start with APB writes

- We’ll add reads shortly.

Notation



Clock

HIGH to LOW

Transient

HIGH/LOW to HIGH

Bus stable

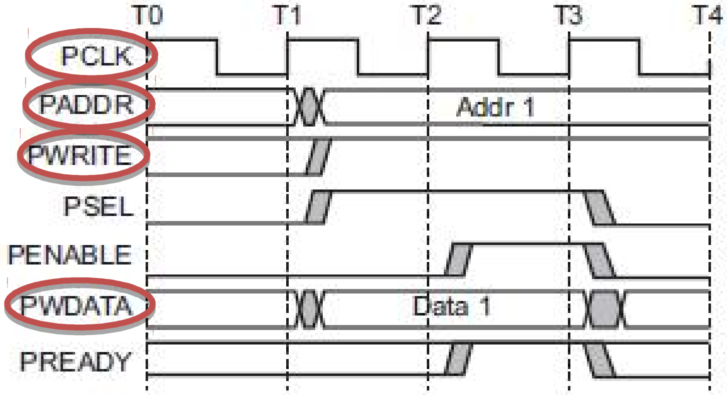
Bus to high impedance

Bus change

High impedance to stable bus

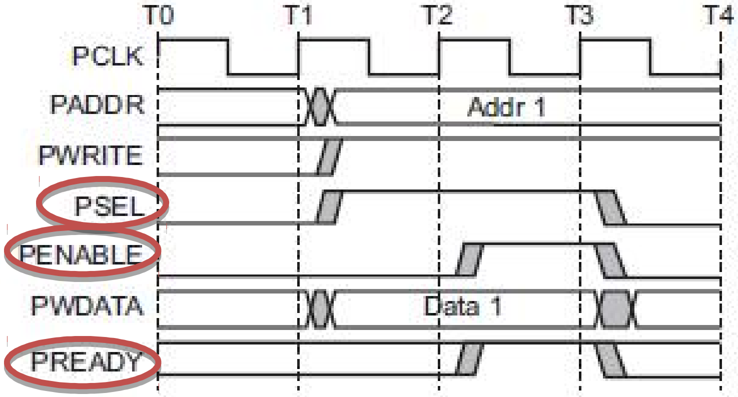
APB bus signals

- PCLK
  - Clock
- PADDR
  - Address on bus
- PWRITE
  - 1=Write, 0=Read
- PWDATA
  - Data from processor

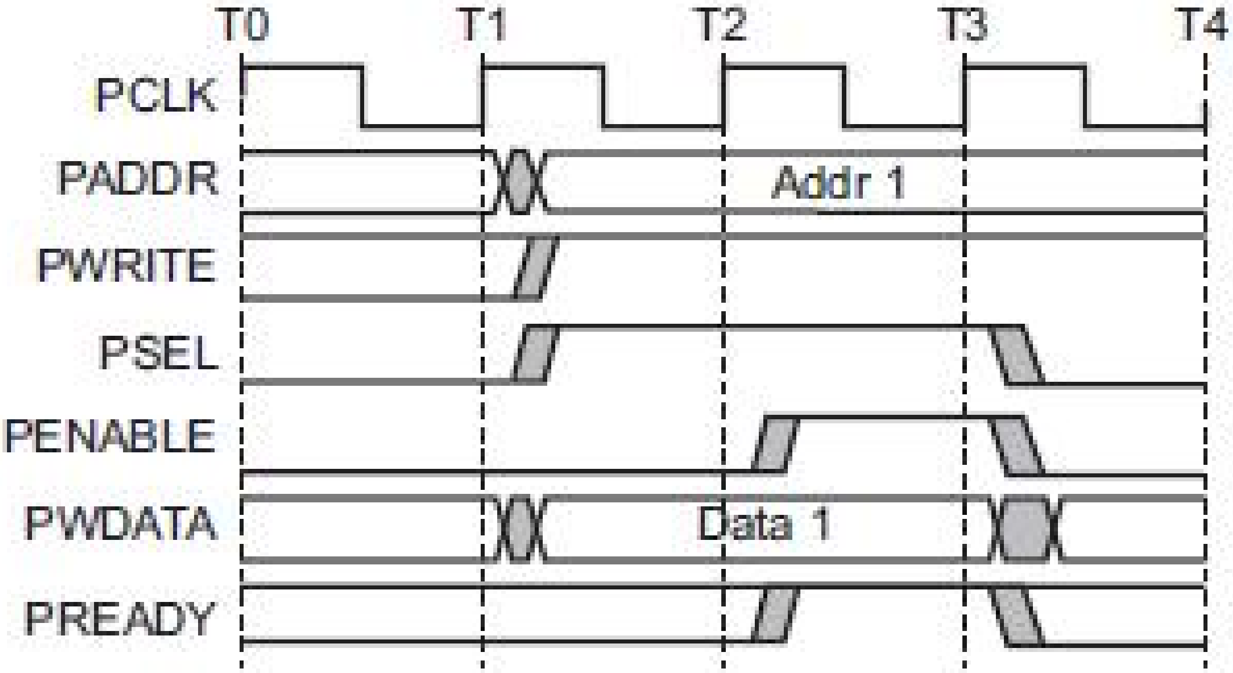


APB bus signals

- PSEL
  - Asserted if the current bus transaction is targeted to ***this*** device.
- PENABLE
  - High during entire transaction *other than* the first cycle. Distinguishes between idle, setup, and ready.
- PREADY
- Driven by target. Similar to #ACK. Means target is *ready*.
- Each target has it's own PREADY line.



What is happening?

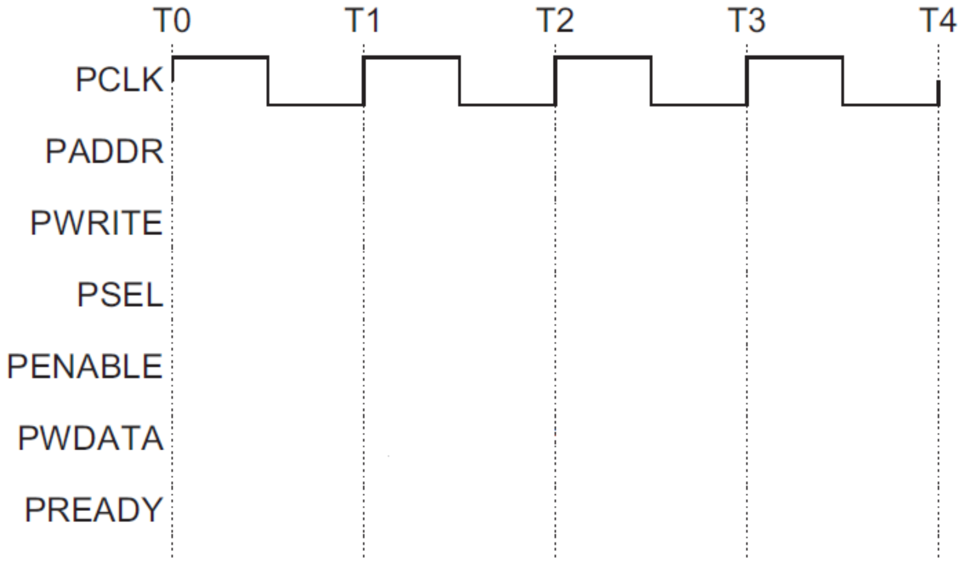


Example setup

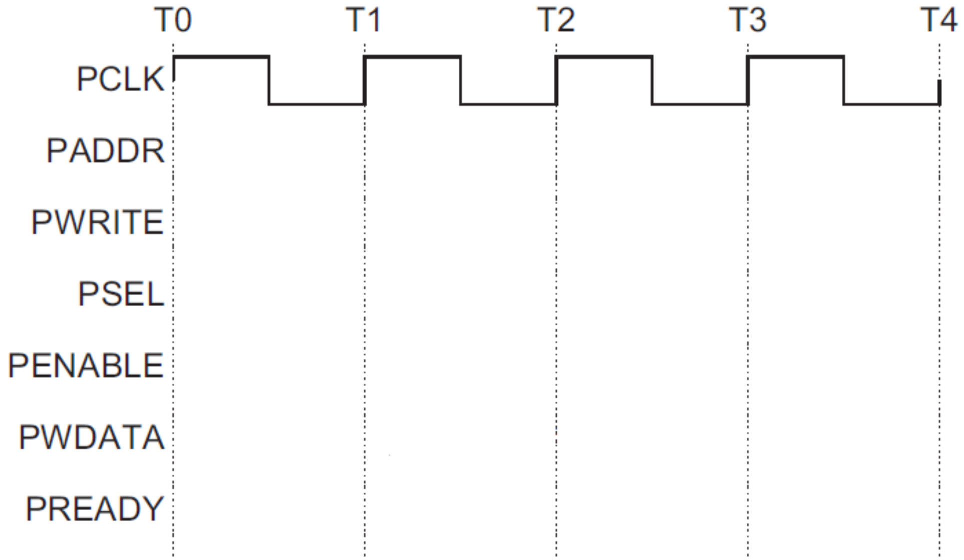
- Assume one bus master “CPU” and two slave devices (D1 and D2)
- D1 is mapped to address 0x00001000-0x0000100F
- D2 is mapped to 0x00001010-0x0000101F

CPU stores to 0x00001004 w.o. stalls

D1



D2



LSB of register controls LED

PWDATA[31:0]

PWRITE

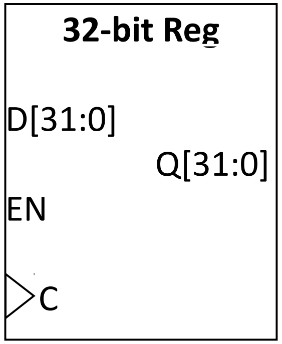
PENABLE

PSEL

PADDR[7:0]

PCLK

PREADY



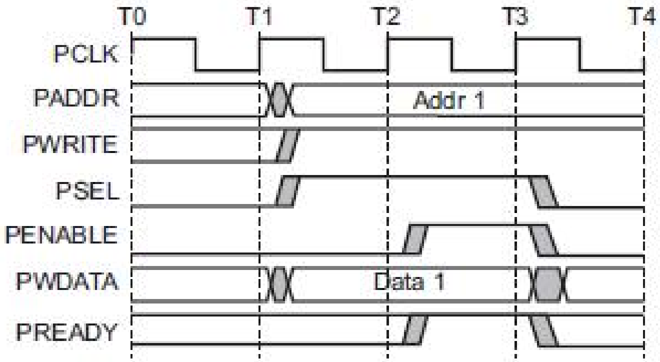
32-bit Reg

D[31:0]

Q[31:0]

EN

C



Assuming APB only gets lowest 8 bits of address

Reg A should be written at address 0x00001000
Reg B should be written at address 0x00001004

PWDATA[31:0]

PWRITE

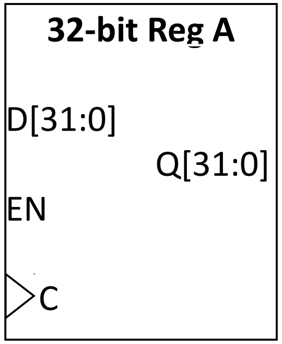
PENABLE

PSEL

PADDR[7:0]

PCLK

PREADY



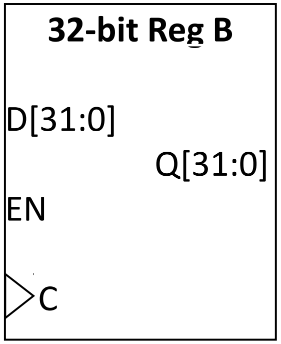
32-bit Reg A

D[31:0]

Q[31:0]

EN

C



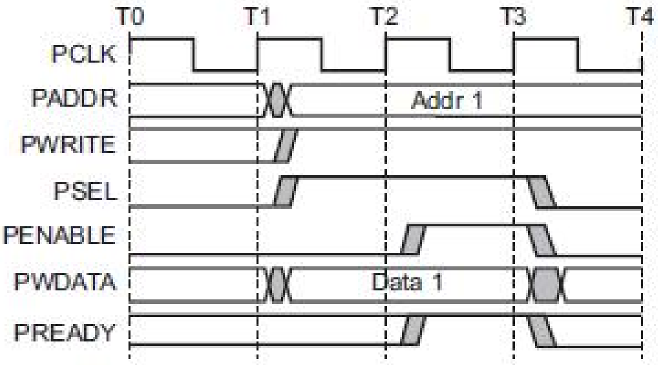
32-bit Reg B

D[31:0]

Q[31:0]

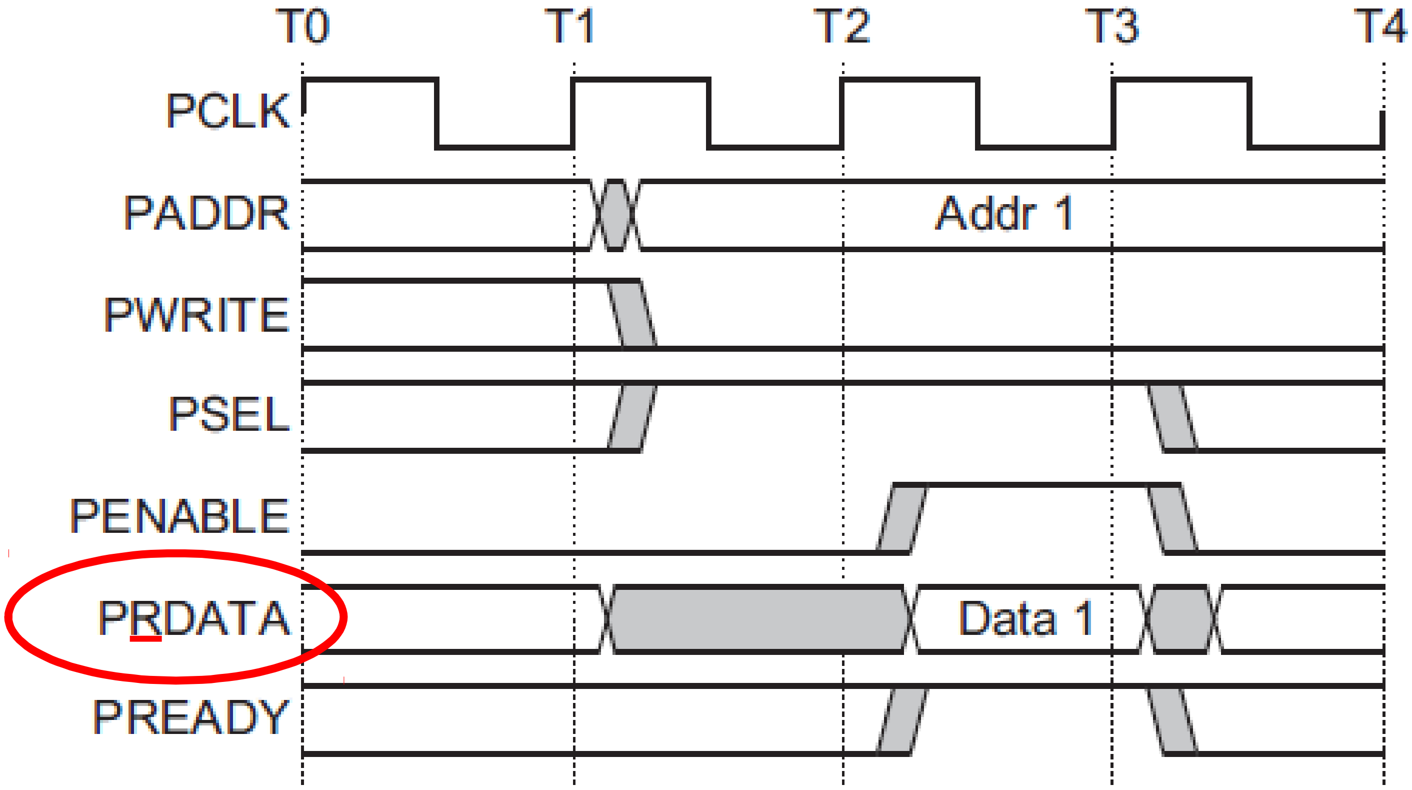
EN

C



Assuming APB only gets lowest 8 bits of address

Reads



Each slave device has its own read data (PRDATA) bus.

Recall that “R” is from the initiator’s viewpoint—the device drives data when read.

Device provides data from switch for any of its addresses

PRDATA[31:0]

PWRITE

PENABLE

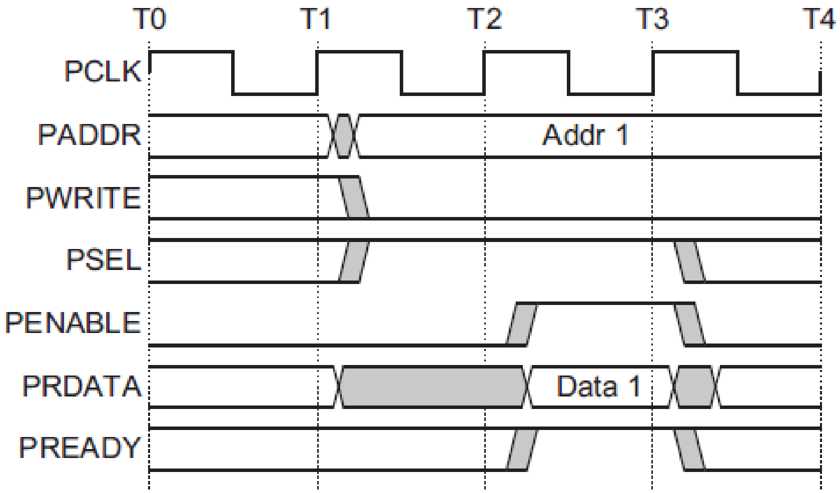
PSEL

PADDR[7:0]

PCLK

PREADY

Switch



Switch A for 0x00001000, B for 0x00001004

PRDATA[31:0]

PWRITE

PENABLE

PSEL

PADDR[7:0]

PCLK

PREADY

Switch A

Switch B

All reads read from register, all writes write

PWDATA[31:0]

PWRITE

PENABLE

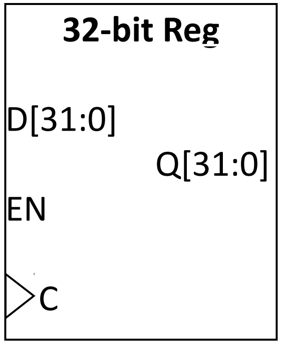
PSEL

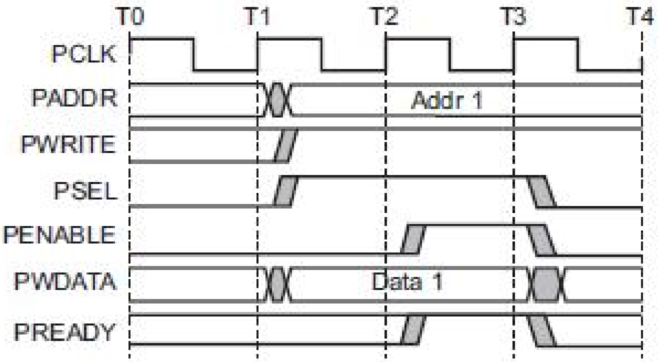
PADDR[7:0]

PCLK

PREADY

PRDATA[31:0]





Assuming APB only gets lowest 8 bits of address

Additional capabilities

- There is another signal, PSLVERR (APB Slave Error) which we can drive high on failure.
- Tie that to 0 if failure impossible.
- Assuming that our device never stalls.
  - We **could** stall if we needed.
  - PREADY.

Verilog

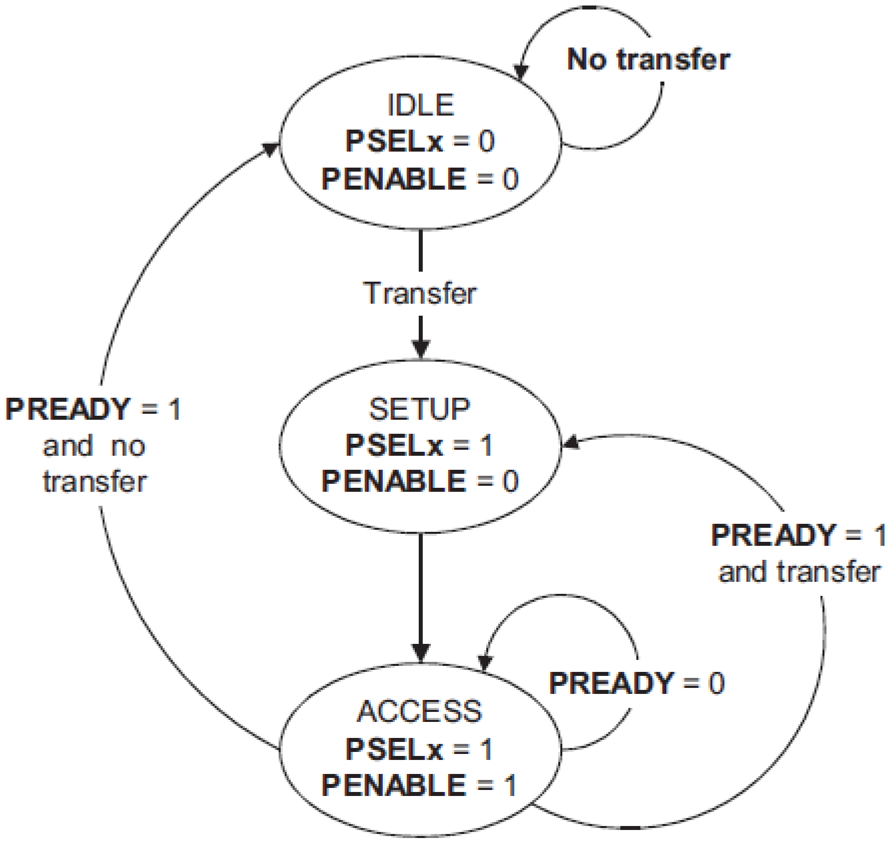
```verilog
/*** APB3 BUS INTERFACE ***/
input PCLK,                   // clock
input PRESERN,                // system reset
input PSEL,                   // peripheral select
input PENABLE,                // distinguishes access phase
output wire PREADY,           // peripheral ready signal
output wire PSLVERR,          // error signal
input PWRITE,                 // distinguishes read and write cycles
input [31:0] PADDR,           // I/O address
input wire [31:0] PWDATA,     // data from processor to I/O device (32 bits)
output reg [31:0] PRDATA,     // data to processor from I/O device (32-bits)

/*** I/O PORTS DECLARATION ***/
output reg LEDOUT,            // port to LED
input SW                      // port to switch
);

assign PSLVERR = 0;
assign PREADY = 1;
```

APB state machine

- IDLE
  - Default APB state
- SETUP
  - When transfer required
  - PSELx is asserted
  - Only one cycle
- ACCESS
  - PENABLE is asserted
  - Addr, write, select, and write data remain stable
  - Stay if PREADY = L
  - Goto IDLE if PREADY = H and no more data
  - Goto SETUP is PREADY = H and more data pending



Done.