TSTE12 Design of Digital Systems Lecture 11

Kent Palmkvist

LINKÖPING UNIVERSITY

Agenda

- Microprogrammed control structures
- Microprogramming

Practical issues

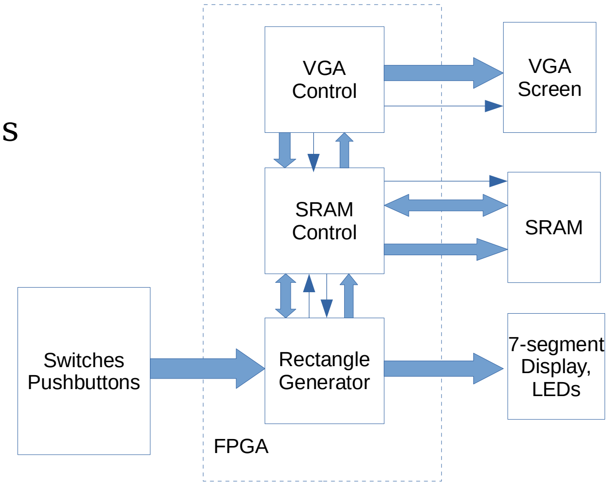
- Handin part 2 deadline today (monday 7/10 at 23:30)
- Individual work!

Lab 3

- Deadline one week after project completion
- Uses an existing design, only add microcode definition
  - Ones and zeros in a memory
- Results always checked by me
  - Send email and ask for me to check (or go to my office and ask)

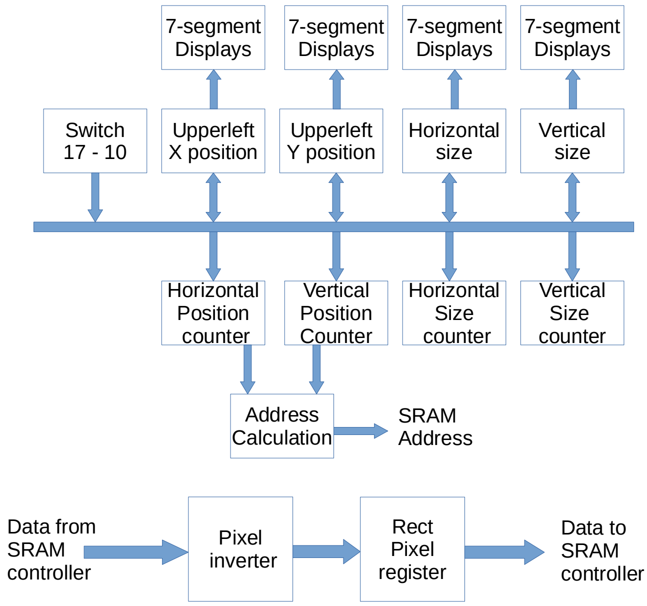
Larger system description

- Lab 3 system
  - Image stored in SRAM
  - Enter rectangle coordinates from switches
  - Invert color in the image
- Mixed control and data flow



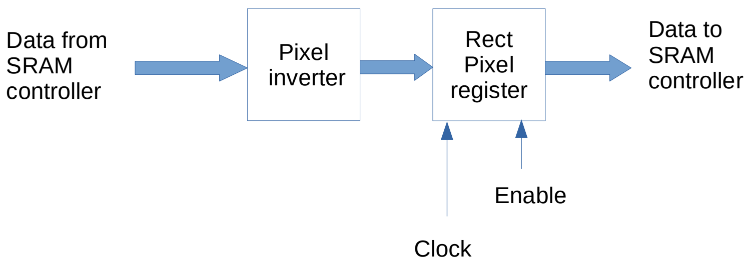
Data flow part

- Separate data flow sections
  - Concurrent operation
- Control implicit in pictures
  - Control signals not drawn



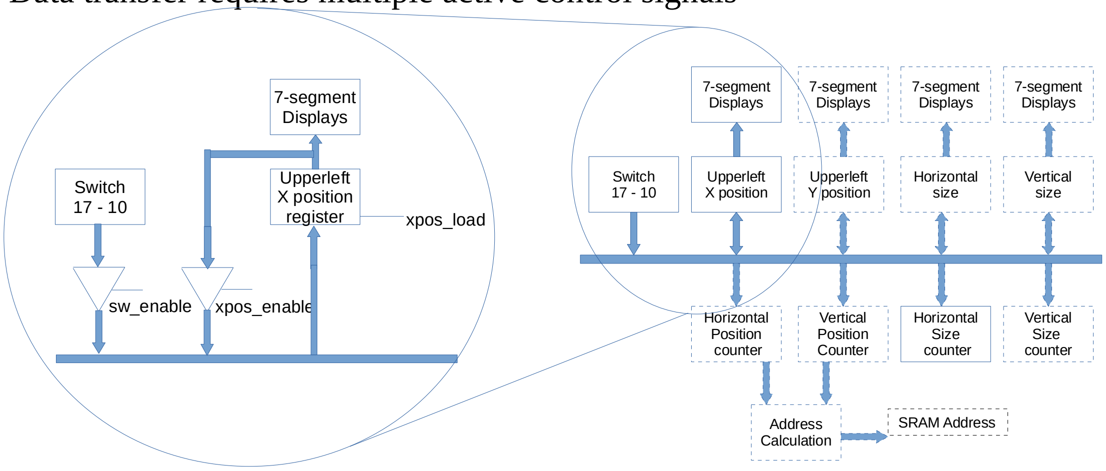
Control signal details

- There are more details hidden in the data flow description
- Implicit clock, enable and other control signals



Multiple control signals necessary

- Data transfer requires multiple active control signals



Large number of control signals

Bit position	Function
28	StoreRectData Store the inverted value of the latest data read from SRAM
27	ReadSW17_10 Read the settings of the switch 17 downto 10
26	Rect-RD Start a read from the SRAM
25	Rect-WR Start a write of the Rectangle data register into SRAM
24	NextHorPos Increment horizontal position counter
23	NextVertPos Increment vertical position counter
22	DecHorCnt Decrement horizontal counter
21	DecVertCnt Decrement vertical counter
20	SetHorPos Set the value of the horizontal position counter
19	SetVertPos Set the value of the vertical position counter
18	SetHorCnt Set the value of the horizontal counter
17	SetVertCnt Set the value of the vertical counter
16	LoadUpperLeftX Set the display showing upper left X value
15	LoadUpperLeftY Set the display showing upper left Y value
14	LoadSizeX Set the display showing the horizontal size
13	LoadSizeY Set the display showing the vertical size
12	ReadUpperLeftX Read the value from the upper left X display register
11	ReadUpperLeftY Read the value from the upper left Y display register
10	ReadSizeX Read the value from the horizontal size display register
9	ReadSizeY Read the value from the vertical size display register
8-5	condition select.
4-0	jump address loaded into the microprogram address counter if the condition is true

Control machine contains large number of states

- Selection of coordinates
  - Upper left x, y of rectangle
  - Width, hight of rectangle
- Update coordinate value
- Loop: read data, invert, write data, increment adress
- Address should run line by line, if necessary increment vertical address and restart horizontal address
  - Total adress calculated automatically as y*1024+x

Partitioning of Finite State Machine (FSM)

- Main problem: Large set of possible sequences
- Dedicated FSM
  - Complex to design
  - Hard to modify
  - Efficient
- Alternative: Microcoded FSM
  - Structured, simple to design and modify
  - Large overhead for small state machines

Microcoded FSM

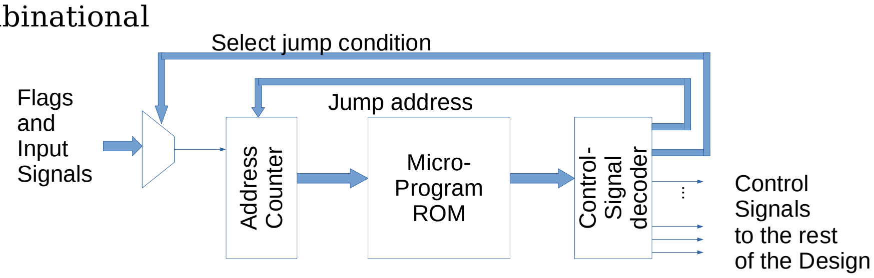
- Some applications requires a longer or complex control step sequence
  - E.g. controller for a microprocessor datapath
- Some applications are too simple for a microprocessor design
  - Datapath control
- Some microprocessors contains a microprogrammed controller
  - Allow patching of processor
  - E.g.; 68000 processor family

Creating a microcontroller

- Simple control machine
  - ROM + register (FSM based on lookup table)
- Replace register with a counter
  - Next state usually corresponds to the next adress i the Lookup table
  - Remove need for an adress to be specified in every control word
  - Possible jump controlled by special control bit
- Conditional jump
  - Control selects condition input, forcing adress load if active

Basic structure

- Expected sequence stored in ROM
  - Once combination/state in each adress
- One clocked block
  - Address counter
  - All other combinational



Behavior

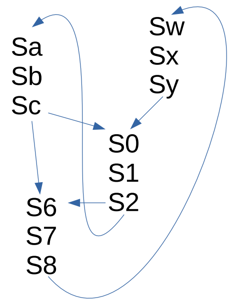
- Moore machine only
  - Control outputs never directly dependent on input
- Conditional jump limited
  - Adress +1 or one branch possible in current structure
    - Corresponds to single if-else
  - Possible to expand hardware to support multichoice branching
    - Corresponds to a case-statement

Timing

- Expected signal update sequence
  - Clock edge -> new adress
  - New adress -> new control values
  - New control values -> new next address
- Some control signals affect outputs in the datapath directly
  - Example: output enable signals
- Some signals affect values on following clock edge
  - Example: register load
- To move a value between a register to another
  - In the same clock cycle enable the output register and the load enable of the reciever register
  - The reciver register will contain the new value at the start of the next clock cycle

Programming

- Sequences are simple to create
  - Signal sequence, auto increment counter
- Branching possible
  - Number of concurrent branch adresses may be limited
- Combine sequences
  - All sequences stored in one ROM



Addr	contents
0	S0
1	S1
2	S2
3	Sa
4	Sb
5	Sc
6	S6
7	S7
8	S8
9	Sw
10	Sx
11	Sy

Control signals

- Current example have long propagation delay (ROM lookup etc.)
- Current example may have glitches on control signals
  - Should not be a problem if design is fully synchronous
- Additional registers may be added
  - ROM output / control bit decode
  - Will delay control signals relative to branching!

Control word (ROM output)

- Can be split into different sections
  - Individual control pins
  - Branch selection
  - Branch address
- Individual bits controls data-path of the system
- Branch related bits control sequence in controller

Branch

- Branch selection
  - Encoded selection possible
- Branch implemented as address register load
- Branch may be done based on more than one input bit
  - Example: microcontroller in microprocessor branch on status bit combinations such as zero or negative

Branch, cont.

- More specialized version possible if important
  - Select 1 out of N (e.g., decode OPCODE in a processor)
  - Dedicated hardware that compute start adress (small ROM)

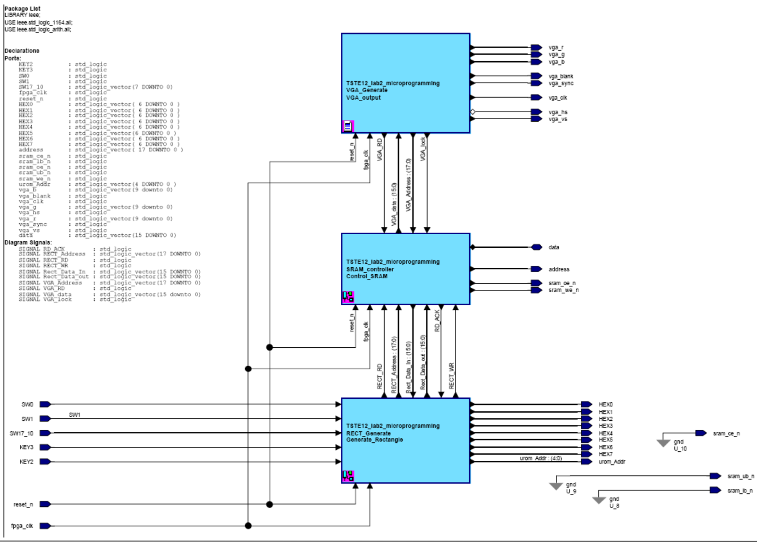
Design steps

- Partition into data flow and FSM
  - Indicate control signals
- Create FSM graph
  - Limit branching
  - Define reset state
- Find sequences
  - What should happen in which order
  - Initially ignore if things can be done in parallel

Design steps, cont.

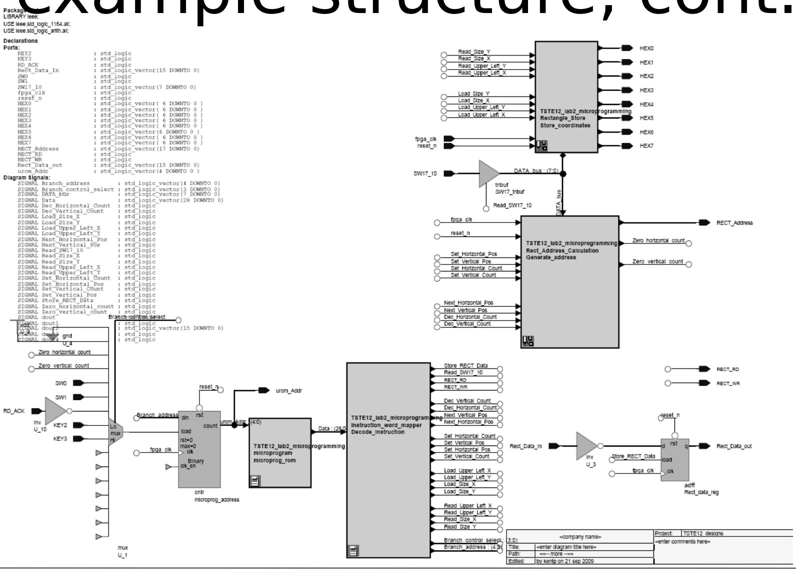
- Compact sequences
  - Datapaths may support multiple activities at once
- Possibly find repeated sequences
  - Sequence with same controls and same end
- Assign addresses to sections of code, adjust branch addresses
- Translate to binary contents in memory

Lab example structure



10/06/2024 23:46

Lab example structure, cont.



Lab example, ROM

- Replace contents in ROM
- Use comment text as help
- Keep at least one ’1’ in each column of the control signals
  - Not necessary for branch adress or jump condition
  - May get synthesis errors if not included

```
ARCHITECTURE behav OF microprogram IS

SUBTYPE memword is bit_vector(28 downto 0);
TYPE memarray is array (0 to 31) of memword;

CONSTANT microprogmem : memarray :=
--                  LL  RR
--  S               oo  ee
--  t               aa  aa
--  oR    N         dd  dd    jmp
--  re   Ne D S S UU  UU
--  ea   exDeSeSe ppLLppRR   c
--  Rd   xtecetet ppooppee   o
--  eSRR tVcVtVtV eeaaeeaa   n
--  cWee HeHeHeHe rrddrrdd   d
--  t1cc orororor LLSSLLSS   i
--  D7tt rtrtrtrt eeiieeii   t
--  a--- PPCCPPCC ffzzffzz   i
--  t1RW oonnoonn tteettee   o  branch
--  a0DR ssttsstt XYXYXYXY   n    Addr
( B"0000_00000000_00000000_0000_00000",   -- 0
  B"0000_00000000_00000000_0000_00000",   -- 1
  B"0000_00000000_00000000_0000_00000",   -- 2
  B"0000_00000000_00000000_0000_00000",   -- 3
  B"0000_00000000_00000000_0000_00000",   -- 4
```

Lab example task

- Key press detection
  - Multiple branch, wait for activation
- Switch setting detection
  - Multiple input branch
- Load/store coordinate info
  - Multiple load/store?
- Memory access
  - Wait for acknowledge
- Wait for key release after completing task

Example microcode use in processor

- Material from “TSEA83 Computer Hardware and Architecture” (by Michael Josefsson)
  - Small simple microprocessor design
    - Programming model is only accumulator, stack, program counter, index register, memory
  - Shows fetch, decode and execute of one instruction
    - Load accumulator with constant $12.
  - www.isy.liu.se/edu/kurs/TSTE12/forelasning/mikroprogram.pdf

Assembly programmers model of the microprocessor example

- Three registers accessible: A, PC, SP
  - PC = program counter, SP = stack pointer, A = ackumulator
- Addressing modes: 6 possible
  - Immediate (next byte is the value)
  - Absolute (next byte is adress to value)
  - ....
- Instructions: initially only LDA, STA and ADD
  - LDA: load register A from memory
  - STA: store register A into memory
  - ADD: add a value from memory to the current value of A

Next time

- Microprocessor structure
- Assembly level programming
- C-level hardware access