ΗΥ-225: Οργάνωση Υπολογιστών
Άνοιξη 2004

Τμ. Επ. Υπολογιστών
© Πανεπιστήμιο Κρήτης

# Βιβλιοθήκη Δομικών Στοιχείων σε Verilog

γιά τις Ασκήσεις 10-13 της Υλοποίησης Επεξεργαστή.

Γιά τις σειρές ασκήσεων 10 έως και 13 που αφορούν την υλοποίηση του επεξεργαστή θα χρησιμοποιήσετε τη βιβλιοθήκη έτοιμων δομικών στοιχείων που βρίσκεται στο αρχείο

**`~hy225/verilog/lib/lib10.v`**

και η οποία περιέχει τα παρακάτω modules.

## 1. RegLd (Καταχωρητής)

`module RegLd (q, d, lden, clk);`

Είναι ο τυπικός ακμοπυροδότητος καταχωρητής, τύπου D, με επίτρεψη φόρτωσης (load enable).

Παράμετροι - Πόρτες:

N Παράμετρος: το πλάτος του καταχωρητή, σε bits.

q Εξοδος (N bits): έξοδος δεδομένων.

d Είσοδος (N bits): είσοδος δεδομένων.

lden Είσοδος (1 bit): έλεγχος επίτρεψης φόρτωσης (load enable) θετικής πολικότητας (1: φόρτωσε, 0: κράτα τα παλαιά περιεχόμενα).

clk Είσοδος (1 bit): ρολόϊ θετικής ακμοπυροδότησης. Οι είσοδοι **d** και **lden** πρέπει να έχουν σταθεροποιηθεί τουλάχιστο 0.2 ns (χρόνος προετοιμασίας, set-up time) πριν τη θετική ακμή του ρολογιού. Ο χρόνος κρατήματος (hold time) είναι 0.1 ns. Η έξοδος **q** αλλάζει τιμή το ταχύτερο μετά από 0.1 ns και το αργότερο μέτα από 0.2 ns (ελάχιστη και μέγιστη καθυστέρηση εξόδου) από τη θετική ακμή του ρολογιού (όταν lden=1). Παρατηρήστε ότι οι παραπάνω χρόνοι ικανοποιούν την παρακάτω συνθήκη (συνήθη σε ακολουθιακά κυκλώματα): η ελάχιστη καθυστέρηση εξόδου (η οποία ονομάζεται και contamination delay) είναι μεγαλύτερη ή ίση από το χρόνο κρατήματος.

Παράδειγμα χρήσης:

```
RegLd #32 pcreg (pc, nxtpc, pcld, clk);
```

## 2. Mux2, Mux4, Mux8 (Πολυπλέκτες)

```
module Mux2 (out, in0, in1, sel);
module Mux4 (out, in0, in1, in2, in3, sel);
module Mux8 (out, in0, in1, in2, in3, in4, in5, in6, in7, sel);
```

Πολυπλέκτες 2-σε-1, 4-σε-1, και 8-σε-1 αντίστοιχα.

Παράμετροι - Πόρτες:

N Παράμετρος: το πλάτος του πολυπλέκτη, σε bits.

out Εξοδος (N bits): πολυπλεγμένη έξοδος. Η καθυστέρηση της εξόδου, σε σχέση με την τελευταία αλλαγή της εισόδου επιλογής (sel) ή της επιλεγμένης (με βάση το sel) εισόδου δεδομένων (inX) είναι το πολύ 0.2 ns στο Mux2, 0.3 ns στο Mux4, και 0.4 ns στο Mux8.

in0 Είσοδος (N bits): είσοδος που επιλέγεται όταν sel=0.

in1 Είσοδος (N bits): είσοδος που επιλέγεται όταν sel=1.

... ...

in7 Είσοδος (N bits): είσοδος που επιλέγεται όταν sel=7.

sel Είσοδος (1 bit στο Mux2, 2 bits στο Mux4, 3 bits στο Mux8): είσοδος επιλογής.

Παράδειγμα χρήσης:

```
Mux2 #32 muxaddr (ma, pc, ALUout, IorD);
```

## 3. ALU (Αριθμητική-Λογική Μονάδα)

`module ALU (out, inA, inB, op);`

Απλή αριθμητική/λογική μονάδα. Εκτελεί πρόσθεση ή αφαίρεση ακεραίων (συμπλήρωμα ως προς 2, 2's complement), ή τις λογικές πράξεις AND/OR.

Παράμετροι - Πόρτες:

N Παράμετρος: το πλάτος της αριθμητικής/λογικής μονάδας, σε bits.

out Εξοδος (32 bits): έξοδος δεδομένων. Η καθυστέρηση της εξόδου, σε σχέση με την τελευταία αλλαγή των εισόδων (δεδομένα ή επιλογή), είναι το πολύ 1.5 ns.

inA Είσοδος (32 bits): πρώτη είσοδος δεδομένων.

inB Είσοδος (32 bits): δεύτερη είσοδος δεδομένων.

op Είσοδος (2 bits): είσοδος επιλογής πράξης:

- 00 πρόσθεση: out = inA + inB.
- 01 αφαίρεση: out = inA - inB.
- 10 bitwise OR: out = inA **OR** inB.
- 11 bitwise AND: out = inA **AND** inB.

Παράδειγμα χρήσης:

```
ALU #32 alu0 (ALUarith, ALUinA, ALUinB, ALUop);
```

## 4. Memory (Μνήμη)

`module Memory (ren, wen, addr, din, dout);`

Γρήγορη ασύγχρονη μνήμη εντολών/δεδομένων.

Πόρτες:

ren Είσοδος (1 bit): ενεργοποίηση ανάγνωσης (read enable).

wen Είσοδος (1 bit): ενεργοποίηση εγγραφής (write enable).

addr Είσοδος (32 bits): διεύθυνση προσπέλασης. Η διεύθυνση αναφέρεται σε bytes (byte-addressable). Δεδομένου ότι εμείς έχουμε μόνο λέξεις στην υλοποίησή μας, τα δύο LS bits της διεύθυνσης αυτής πρέπει να είναι μηδέν, και η πραγματική προσπέλαση γίνεται με τα 30 MS bits. Για πρακτικούς λόγους, κατά την προσομοίωση, υλοποιούνται μόνο οι πρώτες 1024 λέξεις της μνήμης. Επομένως, τα 20 MS bits της διεύθυνσης πρέπει να είναι μηδέν.

din Είσοδος (32 bits): είσοδος δεδομένων προς εγγραφή.

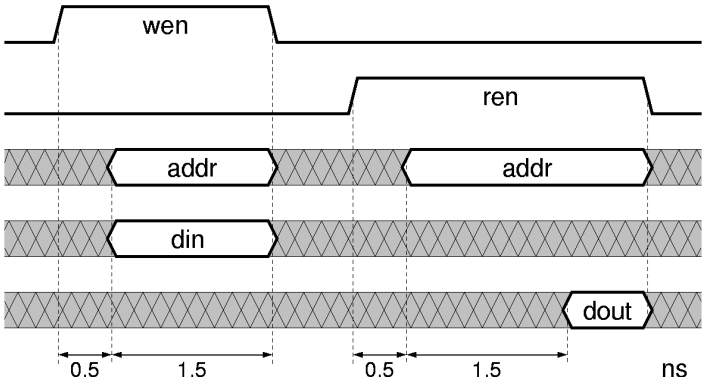
dout Εξοδος (32 bits): εξοδος δεδομένων ανάγνωσης.

Παράδειγμα χρήσης:

```
Memory mem (memRd, memWr, ma, B, md);
```

Η μνήμη είναι ασύγχρονη, δηλαδή ο χρονισμός των προσπελάσεων σε αυτήν **δεν** καθορίζεται από το ρολόϊ του επεξεργαστή (δεν έχει σαν είσοδο το ρολόϊ που έχουν οι ακμοπυροδότητοι καταχωρητές). Για να λειτουργήσει η μνήμη σωστά, ακολουθείται το πρωτόκολλο χρονισμού που φαίνεται στο σχήμα.

**Εγγραφή στη Μνήμη**: γίνεται με βάση το σήμα wen --γιά κάθε εγγραφή:




- Σηκώνουμε το wen.
- Το πολύ σε χρόνο 0.5 ns πρέπει να έχουν σταθεροποιηθεί η διεύθυνση των δεδομένων που θα γράψουμε (addr) και τα ίδια τα δεδομένα (din).
- Για ελάχιστο χρόνο 1.5 ns απο εκεί και πέρα, τα wen, addr, και din πρέπει να παραμείνουν όπως είναι.

**Ανάγνωση απο τη Μνήμη** : γίνεται με βάση το ren --γιά κάθε ανάγνωση:

- Σηκώνουμε το ren.
- Το πολύ σε χρόνο 0.5 ns πρέπει να έχει σταθεροποιηθεί η διεύθυνση των δεδομένων που θα διαβάσουμε (addr).
- Μετά απο άλλα 1.5 ns, η μνήμη θα βάλει τα δεδομένα στο dout. Τα δεδομένα θα παραμείνουν εκεί όσο το ren παραμένει 1 και η διεύθυνση παραμένει σταθερή.

**Προσοχή**: Αν τυχόν χρειάζεται να κάνω δύο διαδοχικές εγγραφές, πρέπει, αφού ολοκληρωθεί η πρώτη, το wen να πέσει και μετά να ξανασηκωθεί για την δεύτερη. Ομοίως και με δύο διαδοχικές αναγνώσεις και το ren.

**Αρχικοποίηση της Μνήμης**: γίνεται με την εντολή "`$readmemh`" της Verilog. Παραδείγματος χάριν, το test bench μπορεί να λέει:

```
$readmemh("memory.hex", dat0.mem.data);
```

όπου, "dat0" το όνομα του datapath σας μέσα στο test bench, "mem" το όνομα της μνήμης σας μέσα στο datapath, και "data" είναι το όνομα του εσωτερικού πίνακα δεδομένου μέσα στη μνήμη. Το αρχείο "memory.hex", που πρέπει να βρίσκεται στο directory στο οποίο τρέχετε τη Verilog και πρέπει να είναι της μορφής:

```
@0    00000000
@4    20100009
@8    00000000
@C    00000000   ... κλπ...
```

Η αριστερή στήλη, μετά το "@", έχει τη διεύθυνση της μνήμης (στο δεκαεξαδικό σύστημα), ενώ η δεξιά στήλη έχει τα δεδομένα (στο δεκαεξαδικό σύστημα, επίσης) που θα φορτωθούν στην αντίστοιχη θέση μνήμης. Στο παραπάνω παράδειγμα, οι λέξεις στις διευθύνσεις 0, 8, και 12 είναι μηδέν, ενώ στη διεύθυνση 4 βρίσκεται η τιμή 32'h20100009. Αν θεωρήσουμε ότι στη διεύθυνση 4 είναι αποθηκευμένη μιά εντολή, τότε αυτή είναι η "`addi $16, $0, 9`". Οι μη αρχικοποιημένες θέσεις μνήμης θα περιέχουν "x".

Εναλλακτικά, υπάρχει η εντολή "**`$readmemb`**", με ακριβώς την ίδια σύνταξη. Γιά αυτήν, το αρχείο πρέπει να έχει την ίδια μορφή όπως παραπάνω, με τις διευθύνσεις στην αριστερή στήλη πάλι στο δεκαεξαδικό, όμως, εδώ, η δεξιά στήλη θα έχει τα δεδομένα στο δυαδικό σύστημα. Έτσι, η εντολή addi στη διεύθυνση 4 θα γραφόταν "`@4 00100000001000000000000000001001`", ή ακόμα καλύτερα, αφού η Verilog επιτρέπει διαχωριστικά underscores:

```
@4   001000_00000_10000_0000000000001001
```

## 5. RegFile (Αρχείο Καταχωρητών)

`module RegFile (raA, raB, wa, wen, wd, rdA, rdB);`

Αρχείο τριανταδύο 32-μπιτων καταχωρητών, με 2 πόρτες ασύγχρονης ανάγνωσης και 1 πόρτα ασύγχρονης εγγραφής.

Πόρτες:

raA Είσοδος (5 bits): διεύθυνση ανάγνωσης πρώτης πόρτας. Ανάγνωση απο τη διεύθυνση 0 δίνει πάντα 0.
raB Είσοδος (5 bits): διεύθυνση ανάγνωσης δεύτερης πόρτας. Ανάγνωση απο τη διεύθυνση 0 δίνει πάντα 0.
wa Είσοδος (5 bits): διεύθυνση εγγραφής. Εγγραφή στη διεύθυνση 0 δεν έχει αποτέλεσμα.
wen Είσοδος (1 bit): ενεργοποίηση εγγραφής (write enable).
wd Είσοδος (32 bits): είσοδος δεδομένων προς εγγραφή.
rdA Εξοδος (32 bits): έξοδος δεδομένων ανάγνωσης πρώτης πόρτας.
rdB Εξοδος (32 bits): έξοδος δεδομένων ανάγνωσης δεύτερης πόρτας.

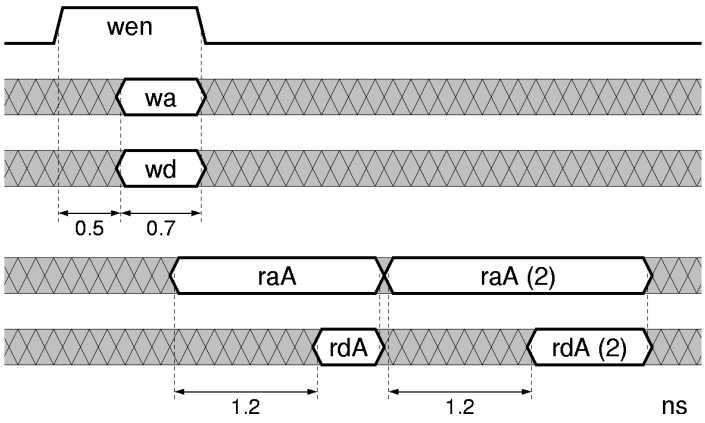
Παράδειγμα χρήσης:

```
RegFile rf (rs, rt, rWA, regWr, rWD, regA, regB);
```

Και το Register File, όπως και η μνήμη, είναι ασύγχρονη. Οι αναγνώσεις γίνονται "συνδυαστικά", απλώς τοποθετώντας τις διευθύνσεις προς ανάγνωση και παίρνοντας μετά απο λίγο τα αποτελέσματα. Η εγγραφή γίνεται όπως και στη μνήμη, με χρήση του wen. Αναλυτικότερα, το πρωτόκολλο φαίνεται στο σχήμα.

**Εγγραφή στο Αρχείο Καταχωρητών** : γίνεται με βάση το σήμα wen --γιά κάθε εγγραφή:

- Σηκώνουμε το wen.
- Το πολύ σε χρόνο 0.5 ns πρέπει να έχουν σταθεροποιηθεί η διεύθυνση των δεδομένων που θα γράψουμε (wa) και τα ίδια τα δεδομένα (wd).
- Για ελάχιστο χρόνο 0.7 ns απο εκεί και πέρα, τα wen, addr, και din πρέπει να παραμείνουν ως έχουν.




**Ανάγνωση από το Αρχείο Καταχωρητών**:
γίνεται απλώς με τη διεύθυνση του καταχωρητή. Στο σχήμα φαίνεται ένα παράδειγμα για την πρώτη πόρτα, αλλά τα ίδια ακριβώς ισχύουν και γιά την δεύτερη. Όταν αλλάζει η διεύθυνση, μετά από 1.2 ns (το πολύ) εμφανίζεται στην έξοδο η τιμή του καταχωρητή. Προσοχή: η έξοδος χαλάει αμέσως μόλις αλλάξει η διεύθυνση. Εδώ, αντίθετα απο τη μνήμη, μπορούμε να διαβάζουμε συνεχώς τιμές από μιά πόρτα, απλώς αλλάζοντας διαδοχικά τη διεύθυνση. Αναγνώσεις μπορούν να γίνονται και ταυτόχρονα με εγγραφή, αρκεί να αφορούν διαφορετικό καταχωρητή --αν διαβάζουμε έναν καταχωρητή την ίδια ώρα που τον γράφουμε είναι ασαφές πότε εμφανίζονται τα νέα δεδομένα στην έξοδο.

**Αρχικοποίηση** του αρχείου καταχωρητών θα μπορούσε να γίνει όπως παραπάνω γιά τη μνήμη, όμως το σωστό είναι να **μην** αρχικοποιήστε τους καταχωρητές "τεχνητά", μέσω Verilog, αλλά το πρόγραμμα που θα τρέξει να αρχικοποιεί μόνο του, έναν-έναν τους καταχωρητές που θα χρησιμοποιήσει.

**Πρόσβαση στο Εσωτερικό, γιά Σκοπούς Αποσφαλμάτωσης**:
Τα κυκλώματα του επεξεργαστή προσπελαύνουν το αρχείο καταχωρητών μόνο μέσω των παραπάνω εγγραφών και αναγνώσεων. Εάν πάντως ο χρήστης, γιά σκοπούς αποσφαλμάτωσης (debugging) και μόνο, θέλει να έχει πρόσβαση στα περιεχόμενα συγκεκριμένων καταχωρητών --π.χ. να διαβάσει το παρόν περιεχόμενο του καταχωρητή $5-- μπορεί να το κάνει ως εξής (π.χ. μέσα από το test bench): "`dat0.rf.data[5]`", όπου "dat0" το όνομα του datapath μέσα στο test bench, και "rf" το όνομα του register file μέσα στο datapath. (Αν θέλουμε τα bits [8:5] του $5, δεν μπορούμε να πούμε απλώς "dat0.rf.data[5][8:5]", αλλά πρέπει να πούμε "wire [31:0] r5; assign r5 = dat0.rf.data[5]; r5[8:5]").

## 6. NOR32

`module NOR32 (out, in32);`

Πύλη NOR 32 εισόδων. Οι είσοδοι έρχονται όλες μαζί, με τη μορφή ενός bus των 32 bits.

Πόρτες:

out Εξοδος (1 bit): το λογικό NOR των 32 bits εισόδου. Η καθυστέρηση της εξόδου, είναι 0.3 ns (το πολύ).
in32 Είσοδος (32 bits): τριανταδύο bits εισόδου.

Παράδειγμα χρήσης:

```
NOR32 zerologic (zero, ALUarith);
```

---



Originally written by S. Lyberis and G. Sapountzis;
last updated: 22 Apr. 2004, by M. Katevenis.