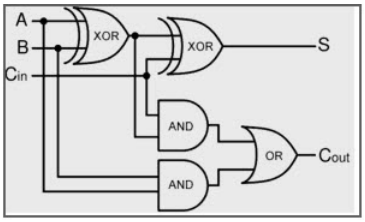
Σύγχρονες Αρχιτεκτονικές Υπολογιστών

Εργαστήριο 1

Μέρος Α: Ιεραρχική σχεδίαση Verilog και δημιουργία Testbench

1. Υλοποίηση και προσομοίωση 1-bit full-adder στο Xilinx Vivado

i) Να γραφεί Verilog module 3-εισόδων-2-εξόδων, έστω `FA (Cout, S, A, B, Cin)`, που να υλοποιεί τη λογική ενός 1-bit πλήρους αθροιστή (1-bit full adder, βλ. διαφάνειες μαθήματος). Πιο συγκεκριμένα, να υλοποιηθεί Verilog module που να υλοποιεί την παρακάτω κυκλωματική περιγραφή σε επίπεδο πυλών.



ii) Να δημιουργηθεί Verilog testbench που να τροφοδοτεί ανά 10 ns τις εισόδους του module `FA (Cout, S, A, B, Cin)` με κατάλληλες τιμές ώστε να προσομοιωθεί καθολικά (για όλους τους δυνατούς συνδυασμούς εισόδων) και να επαληθευτεί η ορθή λειτουργία του κυκλώματος `FA (Cout, S, A, B, Cin)`.

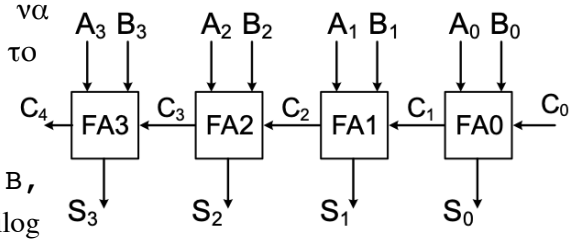
A	B	C_{in}	S	C_{ou}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Σημείωση 1: Για την υλοποίηση του module `FA (Cout, S, A, B, Cin)` να χρησιμοποιηθούν οι πρωτογενείς λογικές πύλες της Verilog.

Σημείωση 2: Η επαλήθευση ορθής λειτουργίας του FA θα γίνει με βάση τον ακόλουθο πίνακα αληθείας.

2. Υλοποίηση 8-bit αθροιστή μη προσημασμένων ακεραίων

i) Να υλοποιηθεί σε Verilog κύκλωμα το οποίο να υπολογίζει το άθροισμα δύο μη προσημασμένων ακεραίων αριθμών των 8-bit. Το κύκλωμα του αθροιστή θα πρέπει να δέχεται ως είσοδο δύο μη προσημασμένους ακεραίους αριθμούς των 8-bit και να υπολογίζει στη έξοδο το 8-bit άθροισμα τους καθώς και το 1-bit κρατούμενο εξόδου. Για παράδειγμα, το σχηματικό διάγραμμα ενός 4-bit αθροιστή δίνεται στο ακόλουθο σχήμα. Να επαναχρησιμοποιηθεί το `FA (Cout, S, A, B, Cin)` από το Μέρος και να γίνει χρήση ιεραρχικής Verilog



σχεδίασης.

ii) Να υλοποιηθεί ο 8-bit αθροιστής με χρήση DataFlow Verilog.
iii) Να δημιουργηθεί Verilog testbench που να επαληθεύει καθολικά/εξαντλητικά (για όλους τους δυνατούς συνδυασμούς εισόδων) την ορθή λειτουργία του 8-bit αθροιστή. Η προσομοίωσή να ελεγχθεί σε επίπεδο κυματομορφής καθώς και να τυπώνει κατάλληλα μηνύματα με χρήση της συνάρτησης `$monitor ("format_string", parameter1, parameter2, ... );` της Verilog.

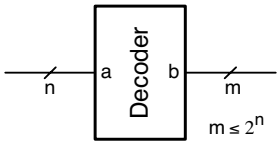
Σημείωση 3: $monitor: Η συνάρτηση $monitor εμφανίζει τις τιμές των παραμέτρων της **ΚΑΘΕ** φορά από οποιαδήποτε παράμετρο αλλάζει την τιμή.

iii) Με χρήση των εργαλείων RTL Analysis, να γίνει επαλήθευση της σχεδίασης σε επίπεδο σχηματικού διαγράμματος. Σημείωση: Για να εμφανιστεί το σχηματικό διάγραμμα toy Verilog module, **Flow Navigator > RTL Analysis > Elaborated Design**) και πατάμε **Schematic**.

iv) Από την καρτέλα **Flow Navigator > Synthesis > Synthesized Design > Schematic**, να ελέγξετε το αντίστοιχο post-synthesis σχηματικό. Πόσοι πόροι του FPGA χρησιμοποιούνται;

Μέρος Β: Αποκωδικοποιητές, πολυπλέκτες και συγκριτές σε Verilog

Β1. Αποκωδικοποιητές και παραμετρική Verilog-HDL



Ένας αποκωδικοποιητής είναι κύκλωμα που μετατρέπει σύμβολα από έναν κωδικό σε άλλο. Ωστόσο, όταν χρησιμοποιείται από μόνος του, ο όρος αποκωδικοποιητής σημαίνει έναν αποκωδικοποιητή από τη δυαδική σε μια one-hot κωδικοποίηση (δηλ. το πολύ ένα bit μπορεί να είναι 1 κάθε φορά και κάθε bit αντιπροσωπεύει ένα σύμβολο). Αυτό μετατρέπει ένα σύμβολο από έναν δυαδικό κώδικα (όπου κάθε μοτίβο bit αντιπροσωπεύει ένα σύμβολο) σε έναν one-hot κώδικα. Το σχηματικό σύμβολο για έναν αποκωδικοποιητή n → m φαίνεται στο παρακάτω σχήμα.

Το σήμα εισόδου a είναι ένα δυαδικό σήμα n-bit και το σήμα εξόδου b είναι ένα one-hot m-bit σήμα ($m \leq 2^n$). Παρακάτω φαίνεται ο πίνακας αλήθειας για έναν αποκωδικοποιητή 3 → 8. Θεωρώντας τόσο την είσοδο όσο και την έξοδο ως δυαδικούς αριθμούς, τότε εάν η είσοδος έχει τιμή i, η έξοδος έχει τιμή 2^i.

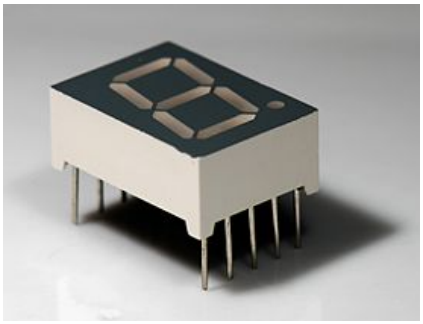
bin	ohout
000	00000001
001	00000010
010	00000100
011	00001000
100	00010000
101	00100000
110	01000000
111	10000000

Ζητούμενα: i) Να υλοποιηθεί σε dataflow Verilog-HDL το συνδυαστικό κύκλωμα ενός αποκωδικοποιητή 3 → 8 και να επαληθευτεί η ορθή λειτουργία του.

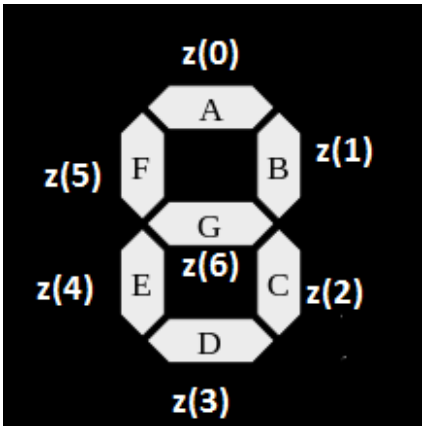
ii) Στη συνέχεια κάνοντας χρήση των δυνατοτήτων παραμετροποίησης που παρέχονται στη Verilog-HDL (βλέπε Παράρτημα Α) να τροποποιηθεί και να παραμετροποιηθεί κατάλληλα η προηγούμενη dataflow Verilog-HDL ώστε να δημιουργηθούν στιγμιότυπα αποκωδικοποιητή 4 → 16 και 6 → 36.

Β2. Αποκωδικοποιητής 7-segment display

Μια οθόνη επτά τμημάτων (7-segment display) απεικονίζει ένα μόνο δεκαδικό ψηφίο φωτίζοντας ένα υποσύνολο επτά τμημάτων που εκπέμπουν φως. Τα τμήματα είναι διατεταγμένα με τη μορφή του αριθμού «8» όπως φαίνεται στο ακόλουθο σχήμα.



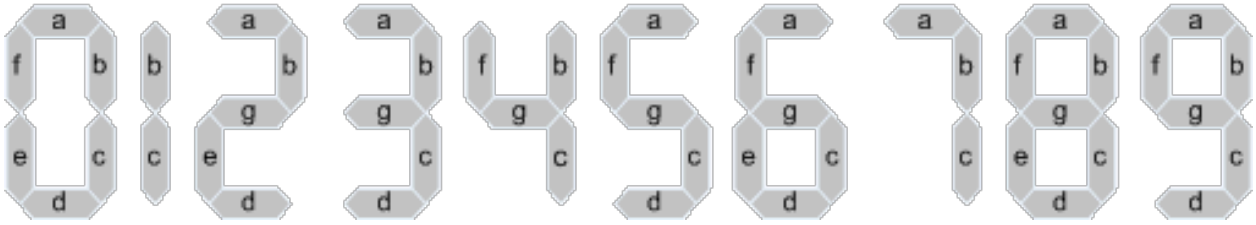
Ένας αποκωδικοποιητής επτά τμημάτων είναι μια λειτουργική μονάδα που δέχεται ένα σήμα εισόδου τεσσάρων δυαδικών κωδικών, bin [3: 0] και δημιουργεί ένα σήμα εξόδου επτά-bit, z [0: 6] που υποδεικνύει ποια τμήματα του 7-segment display θα ενεργοποιηθούν. Η οθόνη πρέπει να φωτίζεται για να εμφανίζει τον αριθμό που κωδικοποιείται στην είσοδο.



Για παράδειγμα, έχουμε σαν είσοδο τον δυαδικό κώδικα "4", δηλ. στον αποκωδικοποιητή 7-segment display εισάγουμε bin=0100, η έξοδος είναι z = 0110011 που δείχνει ότι τα τμήματα 0, 2, 5 και 6 φωτίζονται για να εμφανιστεί το "4".

Ζητούμενα: i) Να υλοποιηθεί σε Verilog-HDL το κύκλωμα ενός αποκωδικοποιητή 7-segment display. Να γίνει χρήση του dataflow επιπέδου σχεδίασης στη Verilog.

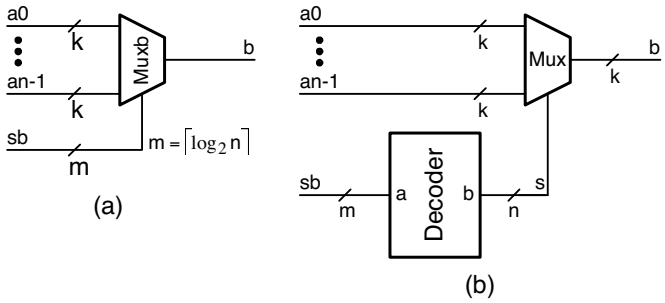
ii) Να επαληθευτεί η ορθή λειτουργία του κυκλώματος μέσω κατάλληλου testbench.



B3. Πολυπλέκτης με δυαδικό σήμα επιλογής

Οι πολυπλέκτες λειτουργούν ως προγραμματιζόμενοι διακόπτες/επιλογείς που επιλεγουν ένα από τα σήματα δεδομένων εισόδου `n` υπό τον έλεγχο του σήματος επιλογής. Οι πολυπλέκτες χρησιμοποιούνται συνήθως σε ψηφιακά συστήματα ως επιλογείς δεδομένων. Για παράδειγμα, ένας πολυπλέκτης στην είσοδο ενός ALU επιλέγει την πηγή δεδομένων για να τροφοδοτήσει το ALU και ένας πολυπλέκτης στις γραμμές διευθύνσεων μιας μνήμης RAM επιλέγει την πηγή δεδομένων για να παρέχει μια διεύθυνση μνήμης σε κάθε κύκλο.

Σε πολλές περιπτώσεις, είναι επιθυμητό να υπάρχει ένας πολυπλέκτης με δυαδικό σήμα επιλογής. Αυτό μπορεί να οφείλεται στο γεγονός ότι το σήμα επιλογής μας είναι σε δυαδικό και όχι one-hot, ή επειδή πρέπει να μεταδώσουμε το σήμα επιλογής μας σε μεγάλη απόσταση (ή μέσω ενός περιορισμένου αριθμού ακίδων) και θέλουμε να εξοικονομήσουμε καλωδίωση. Το παρακάτω σχήμα (α) δείχνει το σύμβολο για έναν πολυπλέκτη δυαδικής επιλογής. Αυτό το κύκλωμα παίρνει ένα σήμα δυαδικής επιλογής `sb` μεγέθους `m = ⌈log2 n⌉` bit, και επιλέγει ένα από τα σήματα εισόδου `ai` σύμφωνα με τη δυαδική τιμή του `sb` - δηλαδή, εάν `sb = i` τότε επιλέγεται το `ai`. Μπορούμε να υλοποιήσουμε ένα πολυπλέκτη με δυαδικό σήμα επιλογής χρησιμοποιώντας έναν αποκωδικοποιητή m → n για την αποκωδικοποίηση του δυαδικού σήματος επιλογής `sb` σε ένα σήμα επιλογής one-hot και στη συνέχεια χρησιμοποιούμε έναν κανονικό πολυπλέκτη (με επιλογή one-hot) για να επιλέγει την επιθυμητή είσοδο.

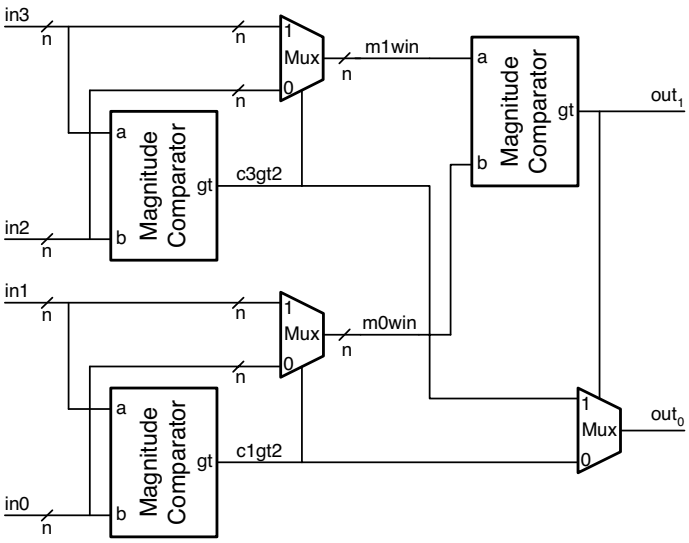


Ζητούμενα: *Να υλοποιηθεί σε Verilog-HDL ένας* `4:1` *παραμετρικός πολυπλέκτη των* `k-bit` *με δυαδικό σήμα επιλογής. Να γίνουν δύο υλοποιήσεις:* ***i)*** *με χρήση του ternary-operator της dataflow Verilog και* ***ii)*** *με χρήση* 2 → 4 αποκωδικοποιητή και `4:1` *πολυπλέκτη με one-hot σήμα επιλογής.* ***iii)*** *Να δημιουργήσετε κατάλληλα στιγμιότυπα για 32-, και 64-bit εισόδου αντίστοιχα, να προχωρήσετε σε σύνθεση και να ελέγξετε την κατανάλωση πόρων του FPGA.*

B4. Κύκλώματα διαιτησίας με προτεραιότητα

Ζητούμενα: Με χρήση **συμπεριφορικής Verilog** και **always@ block** δομών να υλοποιηθεί ένα συνδυαστικό κύκλωμα διαιτησίας με προτεραιότητα (priority arbiter) 4 εισόδων. Το κύκλωμα θα πρέπει να δέχεται τέσσερις εισόδους των 8-bit η κάθε μια και θα εξάγει τον δείκτη της εισόδου με την υψηλότερη τιμή. Σε περίπτωση ισοπαλίας, το κύκλωμα εξάγει τον χαμηλότερο δείκτη που έχει την υψηλότερη τιμή. Για παράδειγμα, ας υποθέσουμε ότι οι τέσσερις είσοδοι είναι: 28, 32, 47 και 19. Το κύκλωμα διατησίας θα εξάγει 2 επειδή η είσοδος 2 έχει την υψηλότερη τιμή, 19. Εάν οι τέσσερις είσοδοι είναι: 17, 23, 19, 23, ο arbiter θα εξάγει 1 λόγω των δύο εισόδων (1 και 3) με την υψηλή τιμή 23, η είσοδος 1 έχει τον χαμηλότερο δείκτη.

Κυκλώματα διαιτησίας με προτεραιότητα χρησιμοποιούνται ευρέως σε network routers όπου το επόμενο πακέτο για αποστολή επιλέγεται σύμφωνα με μια πολιτική ποιότητας υπηρεσίας (Quality-of-Service - QoS) που δίνει σε κάθε πακέτο προς αποστολή μια συγκεκριμένη βαθμολογία. Το πακέτο με την υψηλότερη βαθμολογία αποστέλλεται πρώτα. Η κυκλωματική υλοποίηση του κύκλωματος διαιτησίας με προτεραιότητα, που φαίνεται στο ακόλουθο σχήμα στηρίζεται σε ένα «δέντρο» διαδοχικών συγκρίσεων μεταξύ των εισόδων. Στο πρώτο επίπεδο, οι είσοδοι 0 και 1 και οι είσοδοι 2 και 3 συγκρίνονται. Στο δεύτερο επίπεδο η σύγκριση πραγματοποιείται μεταξύ των εισόδων που προκρίθηκαν στο πρώτου επιπέδου.



Ο κάθε συγκριτής[1] υπολογίζει το σήμα cxgty που είναι 1 εάν inx > iny. Εάν in1 < in0 ή είναι ισοπαλία, αυτό το σήμα είναι 0, υποδεικνύοντας ότι το in0 έχει κερδίσει τον αγώνα. Μια παρόμοια σύγκριση γίνεται μεταξύ in3 και in2. Για να επιλεγούν οι διαγωνιζόμενοι για τον δεύτερο γύρο, χρησιμοποιούνται δύο πολυπλέκτες 2:1. Κάθε πολυπλέκτης επιλέγει τον αντίστοιχο νικητή από κάθε συγκριτή του πρώτου επιπέδου χρησιμοποιώντας το αποτέλεσμα της σύγκρισης ως σήμα επιλογής. Ένας τρίτος συγκριτής εκτελεί το δεύτερο επίπεδο - συγκρίνοντας την έξοδο των δύο νικητών από τους πολυπλέκτες.

[1] Το module **MagnitudeCompatator()** υλοποιεί ένα συγκριτή μεγέθους 2 αριθμών.

Μέρος Γ: Verilog για ακολουθιακά ψηφιακά κυκλώματα και μηχανές πεπερασμένων καταστάσεων

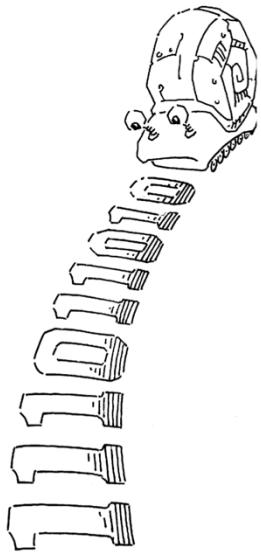
Γ1. Μοντελοποίηση και υλοποίηση ακολουθιακών στοιχείων Flip-Flop και Latches στη Verilog-HDL

Εισαγωγή: Στη Verilog-HDL, η κάθε αλλαγή/μετάβαση στις τιμές των καλωδίων και των μεταβλητών ορίζει ένα έμμεσο συμβάν (event) συγχρονισμού το όποιο μπορεί να ενεργοποιήσει την εκτέλεση κάποιου **always@ block**. Το κάθε event μπορεί επίσης να οριστεί βάσει της κατεύθυνσης της μετάβασης, `negedge` και `posedge` events. Ένα `negedge` event είναι όταν υπάρχει μετάβαση από 1 σε X, Z ή 0 και από X ή Z σε 0. Ένα `posedge` event ορίζεται όταν υπάρχει μετάβαση από 0 σε X, Z ή 1 και από X ή Z σε 1. Η μετάβαση από την ίδια κατάσταση στην ίδια κατάσταση δεν θεωρείται event. `Posedge` και `negedge` events μπορούν να χρησιμοποιηθούν στη λίστα ευαισθησίας των **always@ block** προκειμένου να υλοποιήσουν ακμοπυροδότητα κυκλώματα. Το παρακάτω παράδειγμα δείχνει την χρήση του `posedge` event στην υλοποίηση ενός ακμοπυροδότητου D-flip-flop.

```
1
2  module flop (input  clk,
3               input  d,
4               output reg q);
5    always @ (posedge clk)
6      begin
7          q <= d;   // when clk rises copy d to q
8      end
9  endmodule
```

Ζητούμενα: i) Να υλοποιήσετε σε Verilog-HDL ένα καταχωρήτη των 16-bit με D-Flip-Flop ο οποίος να υποστηρίζει α) ασύγχρονο και β) σύγχρονο μηδενισμό (reset).

Γ2: Το «χαμογελαστό σαλιγκάρι»



Έστω ένα ρομποτικό σαλιγκάρι το οποίο έχει για εγκέφαλο ένα ολοκληρωμένο κύκλωμα που υλοποιεί σε ακολουθιακή λογική μια μηχανή πεπερασμένων καταστάσεων (FSM). Αυτή η μηχανή πεπερασμένων καταστάσεων κάνει το ρομποτικό σαλιγκάρι να κινείται από αριστερά προς τα δεξιά κατά μήκος μιας ταινίας που περιέχει μια ακολουθία 1 και 0. Σε κάθε κύκλο ρολογιού το σαλιγκάρι κινείται στο επόμενο κομμάτι. Το σαλιγκάρι «χαμογελάει» όταν τα τελευταία 4 bits που έχει ανιχνεύσει είναι, από αριστερά προς τα δεξιά, "1101".

Ζητούμενα: Να σχεδιάσετε το διάγραμμα καταστάσεων που υλοποιεί τον τεχνητό εγκέφαλο του ρομποτικού σαλιγκαριού σύμφωνα με τις παραπάνω προδιαγραφές και στη συνέχεια να υλοποιήσετε σε Verilog-HDL το αντίστοιχο ακολουθιακό κύκλωμα και να προσομοιωθεί η λειτουργία του. Θεωρήστε ότι το ολοκληρωμένο κύκλωμα του τεχνητού εγκεφάλου διαθέτει είσοδο A που «διαβάζει/δειγματοληπτεί» συνεχώς το κομμάτι κάτω από τις κεραίες του σαλιγκαριού και έξοδο Y που γίνεται «1» όταν χαμογελά το σαλιγκάρι.

Παράρτημα Α: Παραμετροποίηση Verilog-HDL

Το παρακάτω module αποκωδικοποιητή εισάγει τη χρήση παραμέτρων Verilog. Το module χρησιμοποιεί παραμέτρους `n` και `m` για να επιτρέψει σε αυτόν τον τύπο module να χρησιμοποιείται για την δημιουργία στιγμιότυπων αποκωδικοποιητών αυθαίρετου μεγέθους εισόδου και πλάτους εξόδου. Στην περιγραφή του module, η παράμετρος δήλωσης `n = 2`; δηλώνει ότι το `n` (το πλάτος σήματος εισόδου) είναι μια παράμετρος με μια προεπιλεγμένη τιμή 2. Ομοίως `m` (το πλάτος σήματος εξόδου) είναι μια παράμετρος με μια προεπιλεγμένη τιμή 4. Εάν δημιουργούμε στιγμιότυπο του module ως συνήθως, η μονάδα θα δημιουργηθεί με τις προεπιλεγμένες τιμές για όλες τις παραμέτρους. Για παράδειγμα, ο ακόλουθος κώδικας δημιουργεί έναν αποκωδικοποιητή 2 → 4 καθώς οι προεπιλεγμένες τιμές είναι n = 2 και m = 4.

```
//-----------------------------------------------------------------------
// n -> m  Decoder
// a - binary input   (n bits wide)
// b - one hot output (m bits wide)
//-----------------------------------------------------------------------
module Dec(a, b) ;
  parameter n=2 ;
  parameter m=4 ;

  input  [n-1:0] a ;
  output [m-1:0] b ;

  wire [m-1:0] b = 1<<a ;
endmodule
```

Μπορούμε να παρακάμψουμε τις προεπιλεγμένες τιμές παραμέτρων όταν δημιουργούμε μια μονάδα. Η γενική μορφή για μια τέτοια δημιουργία στιγμιότυπου του module με διαφορετική παραμετροποίηση είναι:

```
<όνομα μονάδας> # (<λίστα παραμέτρων>) <όνομα στιγμιότυπου> (<λίστα θυρών>);
```

Για παράδειγμα, για να δημιουργηθεί ένας αποκωδικοποιητής 3 → 8, ο κατάλληλος κωδικός Verilog είναι: `Dec # (3,8) dec38 (a, b);`