

## PREZENTAREA PROGRAMULUI "MAXPLUS II"

Programul MaxPlusII este un produs al companiei Americane Altera și este dedicat lucrului cu circuitele logice programabile dezvoltate de către această companie. Programul permite realizarea circuitului logic (schematic sau într-un limbaj hardware VHDL, Verilog, AHDL) compilarea circuitului pentru o anumită arie programabilă, simularea circuitului și programarea ariei.

### **Ce sunt ariile logice programabile?**

Un circuit logic programabil (PLD – Programmable Logic Device) este un circuit integrat folosit pentru a obține un circuit digital reconfigurabil, sau configurabil după aplicație. Spre deosebire de circuitele logice clasice (porți, numărătoare, regiștri), care au o funcție fixă, un PLD nu are o funcție predefinită în momentul fabricației. Înainte de a fi folosit el trebuie programat.

Înainte ca PLD să fie inventate, memoriile erau folosite pentru a crea circuite logice combinaționale arbitrare. Considerând o memorie cu  $m$  intrări de adresă și  $n$  ieșiri, avem la dispoziție  $2^m$  locații cu  $n$  biți fiecare. Dacă adresele sunt comandate de către  $m$  semnale independente (intrările funcțiilor logice de implementat), iar în locațiile memoriei se află valorile celor  $n$  funcții, structura se comportă ca un circuit logic combinațional cu  $m$  intrări și  $n$  ieșiri. Deoarece memoriile nu au regiștri de intrare sau ieșire, nu pot fi folosite pentru realizarea de circuite secvențiale.

De-a lungul timpului s-au inventat multe tipuri de PLD: PAL, GAL, CPLD, FPGA.

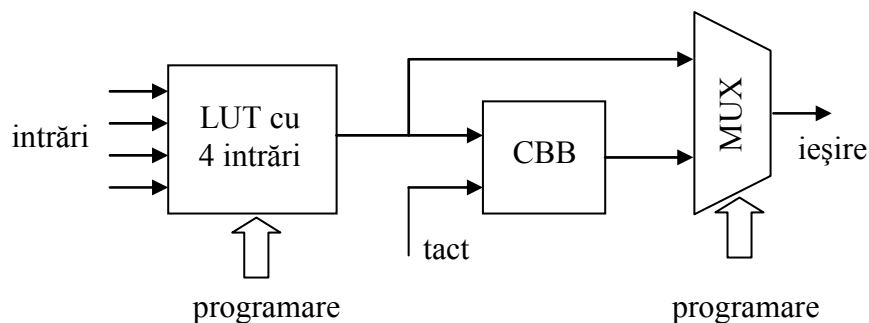
PAL – Programmable Array Logic (Arie Logică programabilă) sunt bazate pe memorii PROM (Memorie programabilă de tip read-only) mici plus o logică adițională de ieșire folosită pentru implementarea unor funcții logice particulare într-un singur integrat.

GAL – Generic Array Logic (Arie logică generică) sunt o variantă evoluată a PAL-urilor, ce pot îngloba funcții cât mai multe PAL-uri la un loc și care permit și reprogramabilitatea.

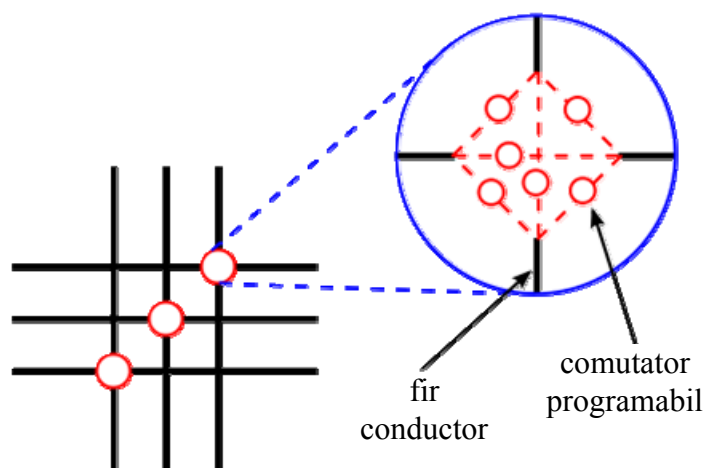
CPLD – Complex Programmable Logic Device (Circuit Logic programabil complex) are un grad de complexitate mai mare decât GAL. Circuitul este realizat din celule (Macro Cell) ce permit implementări

normal disjunctive ale expresiilor logice. Permit memorarea configurației după oprirea alimentării și conțin un număr mare de porți logice.

FPGA – Field Programmable Gate Array (Arii logice programabile în circuit) sunt cele mai evoluat din punct de vedere a flexibilității și mărimii. Circuitul se bazează pe blocuri logice (Logic Blocks) a căror structură este prezentată în figura 1a.



a)



b)

Figura 1. FPGA: a) Bloc logic în FPGA; b) matrice de comutatoare

LUT înseamnă Look Up Table – Tabel de căutare, memorie, CBB este un circuit basculant bistabil, iar MUX este un multiplexor. Acest gen de celule pot implementa atât circuite combinaționale prin preluarea ieșirii de la cea a LUT, fie secvențele prin preluarea ieșirii de la CBB. Celulele pot fi conectate între ele după dorință prin intermediul unei matrici de comutatoare (Figura 1b) pe lângă faptul că ele însele pot fi programate. În plus ieșirile și

intrările în arie pot fi programate după dorință la oricare din pinii disponibili ai circuitului.

### MaxPlusII

MaxPlusII este un soft gratuit ce poate fi încărcat liber de pe site-ul companiei ([www.altera.com](http://www.altera.com)), licența de funcționare acordându-se gratuit după o înregistrare prealabilă.

Programul se instalează de regulă pe C:\maxplus2\ și își mai creează un director C:\max2work\. Prima deschidere a programului ne prezintă o fereastră ca în figura 2.

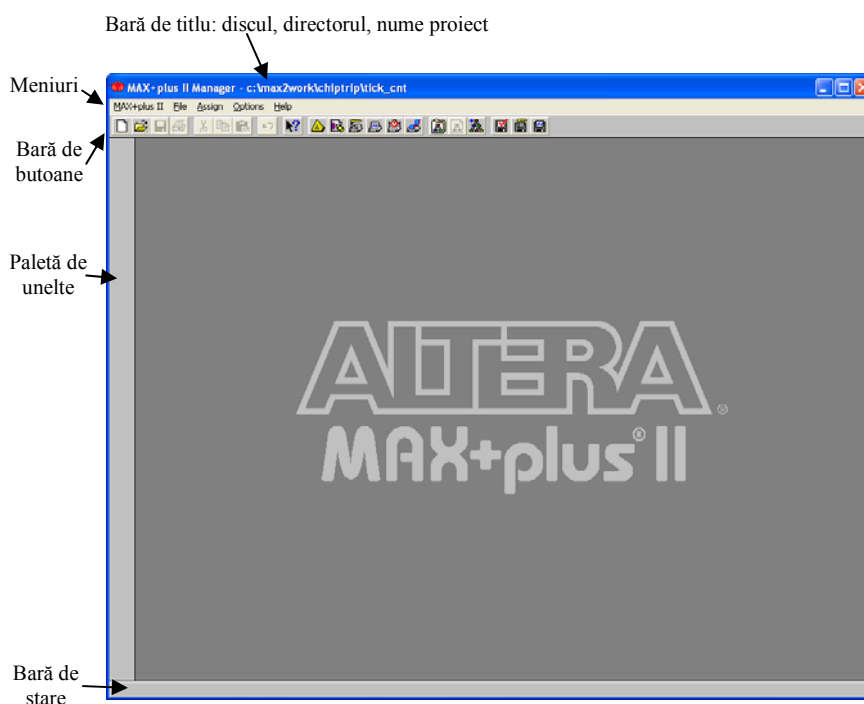
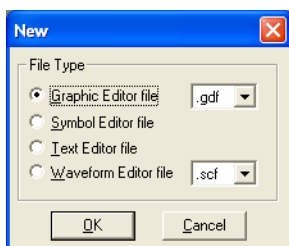


Figura 2. Fereastra principală a MaxPlusII

Pe parcursul laboratorului vom lucra cu 2 tipuri de fișiere: gdf – grafic editor file (fișier de editare grafică) și scf – waveform editor file (fișier de editare a formelor de undă). Pentru înțelegere vom crea un proiect simplu.

Din meniul *File* sau din butonul  de pe bara de butoane se deschide un document nou. Se deschide o fereastră de dialog:



din care vom selecta editorul grafic ca în figură și apoi se apasă *OK*. Fereastra principală se populează cu o foaie albă, iar Paleta de unelte și o serie de butoane de pe bara cu butoane se activează (figura 3). Plimbând mouse-ul deasupra lor, în bara de stare se afișează funcția lor.

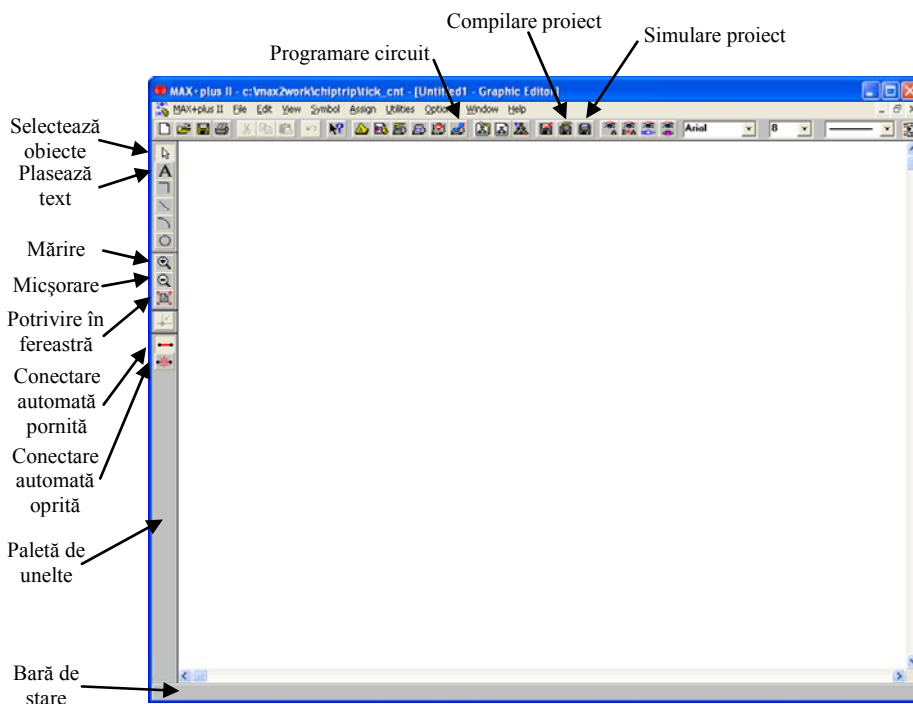
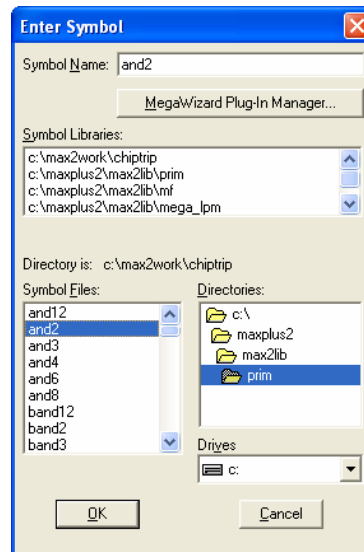
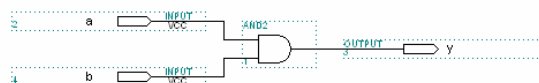



Figura 3. S-a deschis un nou fișier grafic

Pe foaia albă se face dublu click cu mouse-ul, sau se face click dreapta și se alege *Enter Symbol* și se deschide o fereastră de dialog:

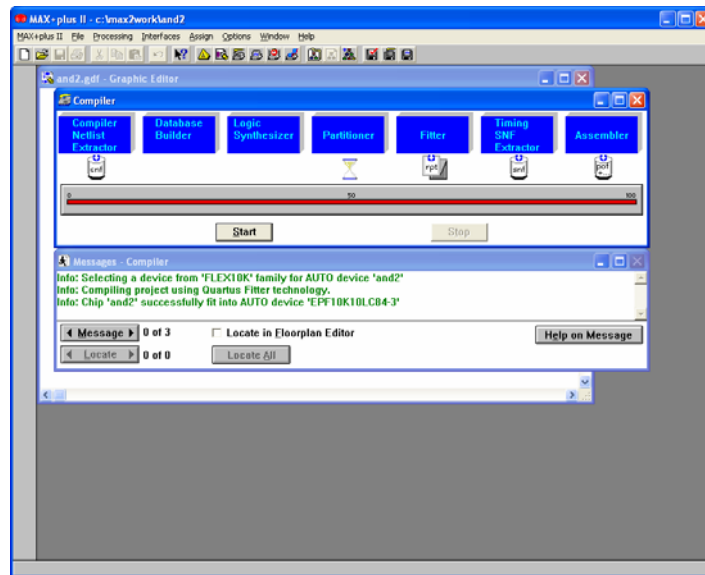


De aici se pot alege componentele, fie prin tastarea numelui în zona *Symbol Name* dacă se cunoaște numele, fie prin alegerea dintr-o bibliotecă (dublu click pe una din bibliotecile din zona *Symbol Libraries*, și apoi alegerea simbolului din zona *Symbol Files*) și se încheie cu *OK*. Se alege o poartă ȘI cu 2 intrări al cărei nume este "and2". Imediat după apăsarea *OK* poarta apare pe foaie. Circuitului trebuie să îi adăugăm intrări și ieșiri. Acestea sunt tot componente și se numesc "input" și "output". Se procedează ca și în cazul porții. Sunt necesare 2 intrări și o ieșire. Acestea se denumesc făcându-se click dreapta și alegând "Enter pin name" și apoi tastând numele dorit. Acesta **trebuie** să conțină doar litere și cifre. Cele două intrări se denumesc *a* și *b*, iar ieșirea *y*. Cursorul se transformă automat pentru trasarea legăturilor între componente. Se leagă cele două componente input la cele două intrări ale porții și componenta output la ieșirea acesteia, ca mai jos.

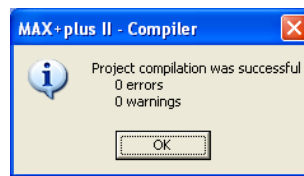


În acest moment circuitul este gata. Fișierul se salvează (File, Save sau butonul ). Automat i se pune extensia .gdf. Apoi din meniul File, Project, Name, sau "Set Project to Current File" se alocă fișierul unui proiect. **Toate fișierele proiectului trebuie să se afle în același loc pe disc și să se cheme la fel.**

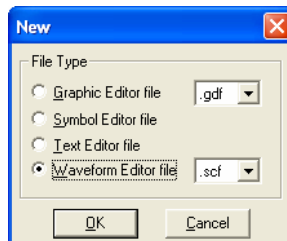
Se apasă butonul de compilare  și se compilează circuitul:



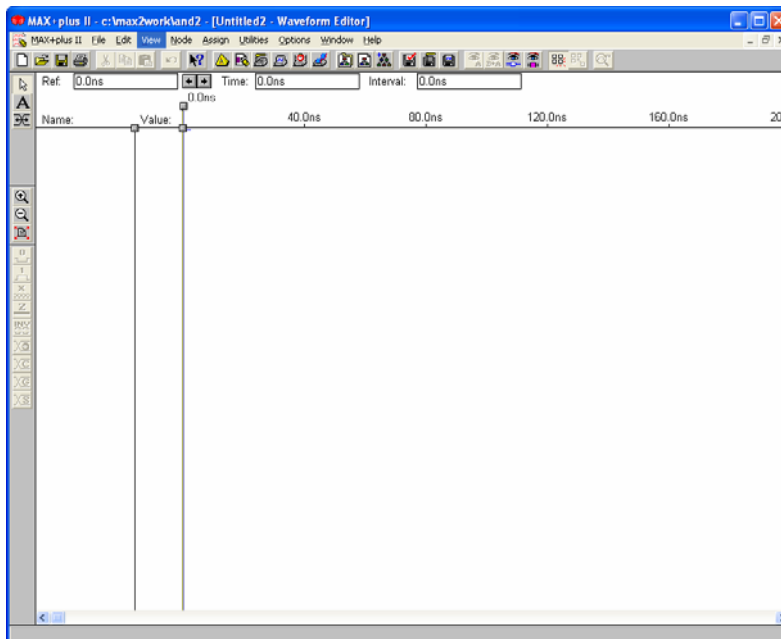
Programul deschide compilatorul, face compilarea, returnează mesaje de eroare sau de atenționare în fereastra "Message window" și apoi anunță încheierea procesului:



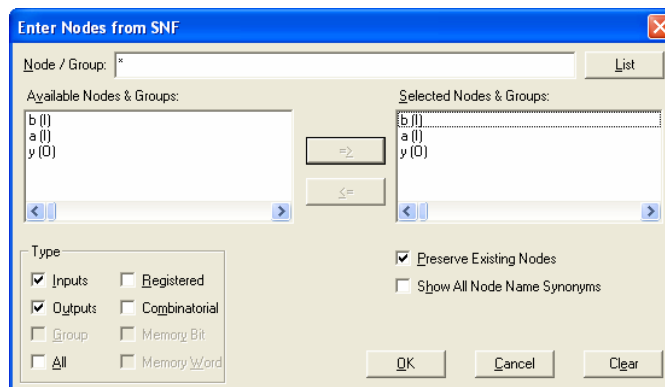
Dacă procesul nu s-a încheiat cu succes, atunci se citesc mesajele de eroare și se procedează la remedierea lor. Există și un buton "Help on message" care deschide o fereastră de ajutor. Dacă procesul s-a încheiat cu succes, se închide compilatorul și se trece la etapa de simulare prin deschiderea unui nou fișier, de editare a formelor de undă de data asta:



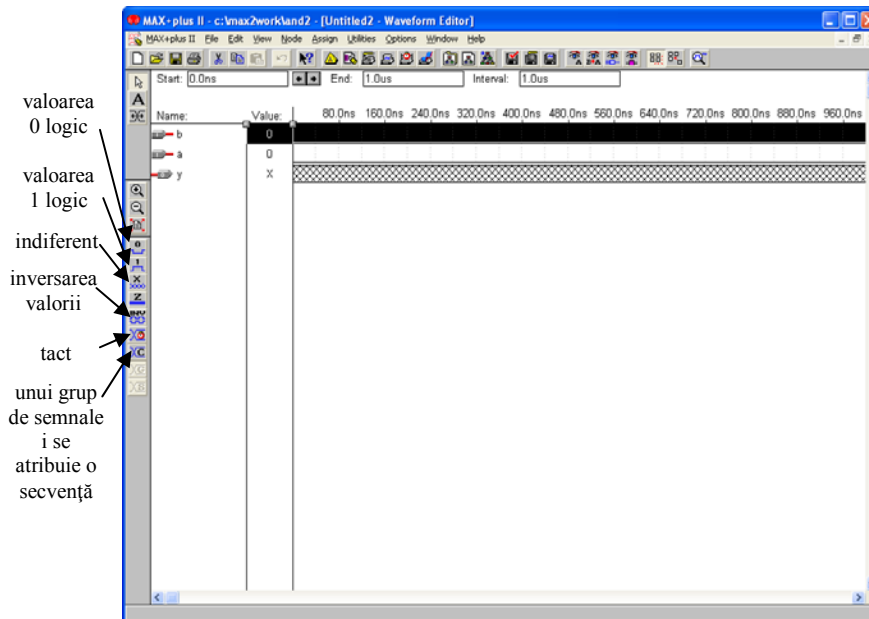
Fereastra se populează cu o foaie albă de tip "editor file":



Se pot introduce semnalele câte unul sau toate odată. Se face click dreapta sub name și se alege fie "Insert node" fie "Enter nodes from snf". Alegem opțiunea a doua deoarece e mai comodă, mai ales atunci când sunt multe semnale. Se deschide o nouă fereastră de dialog:

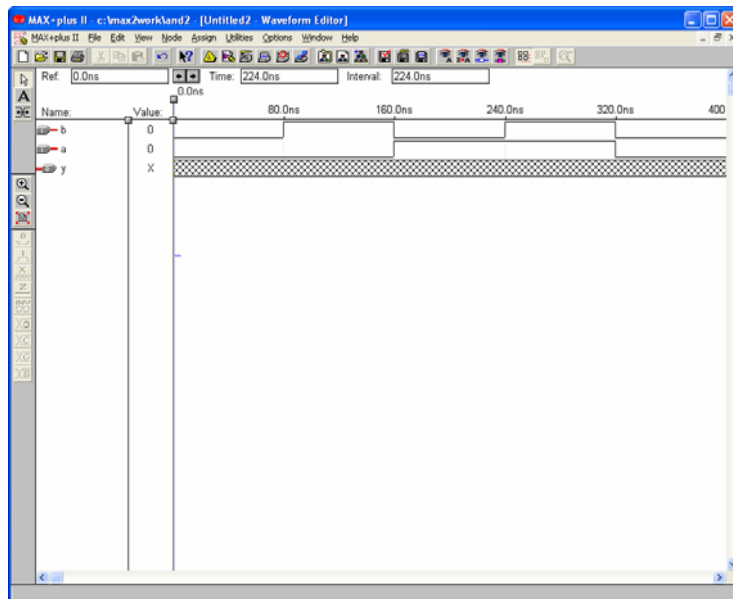



se apasă butonul list, apoi  $\Rightarrow$  și OK. Pe foaia fișierului formelor de undă au apărut cele 3 semnale:

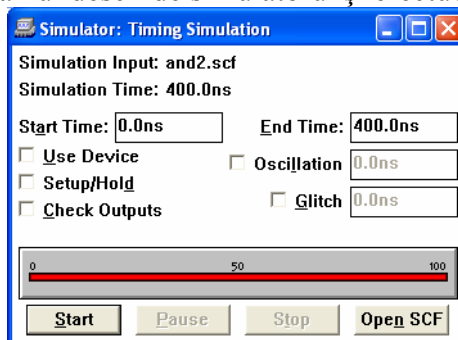


Semnalelor de intrare le vom da valori, iar cele de ieșire, care acum sunt hașurate, vor rezulta din simulare. Dacă faceți click pe fiecare din semnale, Paleta de unelte se activează și puteți alege diferite valori sau succesiuni de valori pentru formele de undă. Pentru cazul nostru alegem un semnal de tip tact, pentru intrarea *a* multiplicat cu 1, iar pentru *b* multiplicat cu 2. Pentru *n* intrări există  $2^n$  stări distincte. Deci în cazul nostru avem  $2^2=4$  stări. Pentru fiecare intrare în plus tactul se multiplică cu 2 față de precedentă. În acest fel vom obține toate stările posibile în ordinea lor naturală fără să se repete. Funcție de valoarea gridului se fixează și durata unei stări: *Options, Grid size*. În cazul nostru îl alegem de 80ns. Timpul total de simulare trebuie să fie cel puțin egal cu numărul de stări înmulțit cu durata unei stări:  $4 \times 80\text{ns} = 320\text{ns}$ . Astfel din meniul *File, End time* fixăm timpul final la 400ns, adică o stare mai mult decât necesar. Am ales astfel deoarece circuitele introduc întârzieri și astfel vom putea vedea și răspunsul complet la ultima stare.

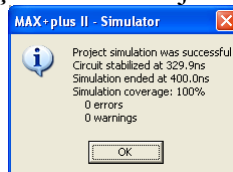




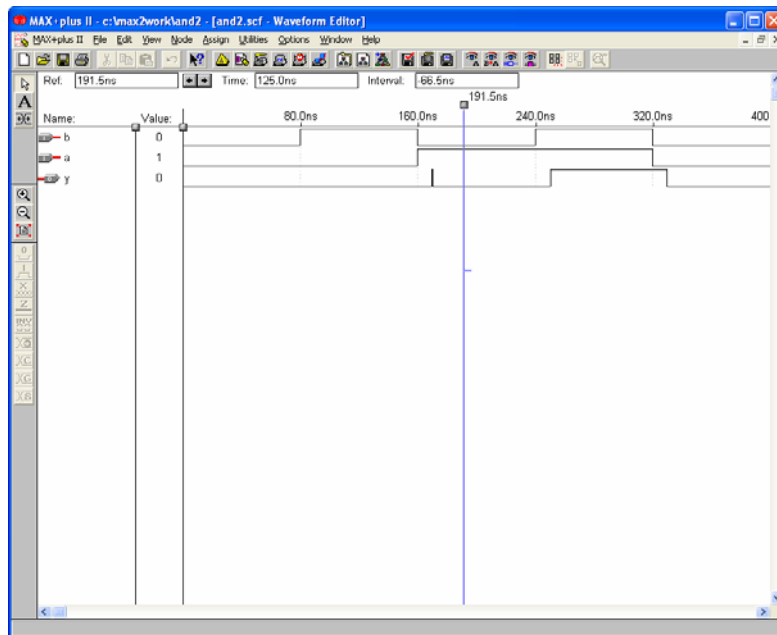
În acest moment putem porni simularea prin apăsarea butonului . Programul cere să salvăm fișierul, pe care l-am denumit deja precum fișierul *gdf* doar că are extensia *scf* și pe care sugerează să îl salvăm în același loc. Se apasă OK. Programul deschide simulatorul și efectuează simularea:



Dacă nu sunt erori atunci afișează un mesaj de sfârșit:

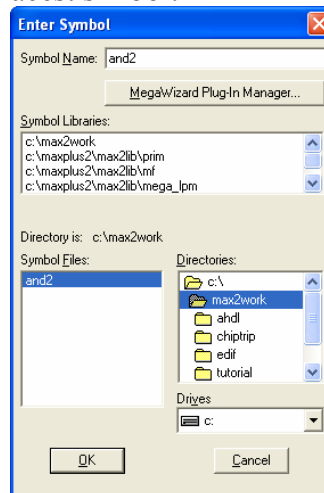


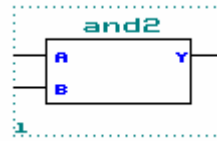
și după ce se apasă OK și se închide simulatorul, observăm că în fereastra formelor de undă a apărut și forma de undă a ieșirii:



Cu ajutorul unui cursor ne putem plimba pe forma de undă, iar în coloana „Value” vom găsi valorile semnalelor în dreptul cursorului. Se observă și întârzierea semnalului de ieșire față de cele de intrare. Se urmărește corespondența între valorile intrărilor și rezultatul de la ieșire.

Odată funcțional, circuitului i se poate atașa un simbol, care ulterior poate fi folosit în alte circuite. Cu foaia schemei în față, din meniul *File*, *Create Default Symbol* se creează acest symbol. Pe o foaie nouă de fișier schematic *gdf* se introduce acest simbol:





Acesta poate fi folosit ulterior în alte circuite cu condiția ca toate fișierele proiectului în care a fost creat să se găsească în locul în care este noul proiect. Dacă dați dublu click pe acest simbol, programul va deschide schema anterioară.

Această facilitate de a crea simboluri pentru circuite permite lucrul mai facil cu scheme mari.