Multiplicator de numere complexe

Funcționare

Circuitul realizează înmulțirea a două numere complexe reprezentate sub forma algebrică.

Părțile reale și imaginare ale operanzilor sunt numere întregi reprezentate pe 8 biți, în complement față de 2.

Dacă se notează:

$$z_1 = x_1 + i * y_1$$

$$z_2 = x_2 + i * y_2$$

atunci rezultatul este:

$$z_1 * z_2 = r = x_r + i*y_r,$$

unde:

$$x_r = x_1 * x_2 - y_1 * y_2$$

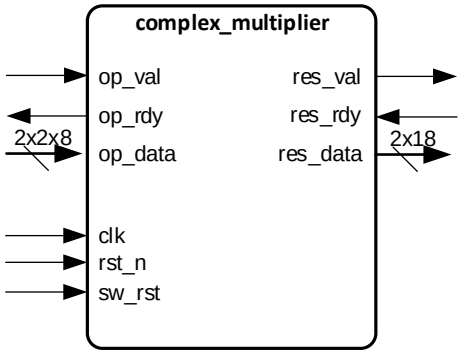
$$y_r = x_1 * y_2 + x_2 * y_1$$

Circuitul este sincron și are un semnal de reset asincron activ în 0.

Circuitul are un semnal de reset sincron activ în 1.

Interfețele cu operanzii și cu rezultatul sunt de tip "*valid-ready*".

Simbol



Descrierea porturilor

Nume port	Descriere	Lățime [biți]	Direcție
Semnale globale			
clk	Semnal de ceas.	1	I
rst_n	Reset asincron activ în 0. Activarea acestui semnal, indiferent de ceas, determină aducerea imediată a multiplicatorului în starea inițială.	1	I
sw_rst	Reset sincron activ în 1. Activarea acestuia pentru o perioadă de ceas, sincron cu ceasul, determină aducerea multiplicatorului în starea inițială, la următorul front activ al semnalului de ceas. Semnalul are o lățime minimă de o perioadă de ceas.	1	I
Interfață cu operatorii			
op_val	Operanzi valizi.	1	I
op_rdy	Este permisă primirea operanzilor.	1	O
op_data	Operanzi (x_1, y_1, x_2, y_2)	2 * 2 * 8	I
Interfață cu rezultatul			
res_val	Rezultat valid.	1	O
res_rdy	Este acceptat rezultatul.	1	I
res_data	Rezultatul (x_r, y_r)	2 * 18	O

Resurse

Se presupune că este disponibil un modul care realizează multiplicarea a două numere întregi, pozitive, reprezentate pe 8 biți.

```
module uint8_mult (
input   [8  -1:0] op1    ,
input   [8  -1:0] op2    ,
output  [16 -1:0] result
);

assign result = op1 * op2;

endmodule // uint8_mult
```

Cerințe

a) Proiectați multiplicatorul de numere complexe utilizând o singură instanță **uint8_mult**.

b) Proiectați multiplicatorul de numere complexe utilizând două instanțe **uint8_mult**.

c) Proiectați multiplicatorul de numere complexe utilizând patru instanțe **uint8_mult**.

Livrabile

Pentru fiecare din cerințele a)-c) menționate se va livra:

1) Desen cu structura proiectată.
2) Model Verilog.
3) Forme de undă rezultate din simulare pentru simularea cazului:

$$(2 + i*3) * (4 + i*2) = 2 + i*16$$

Verificare

Modelele a)-c) vor fi verificate într-un mediu de testare unic.

Mediul de testare va cuprinde module de verificare automată a rezultatelor.

Se vor verifica următoarele înmulțiri:

$$(2 + i*3) * (4 + i*2) = 2 + i*16$$
$$(3 + i*3) * (4 + i*2) = 6 + i*18$$

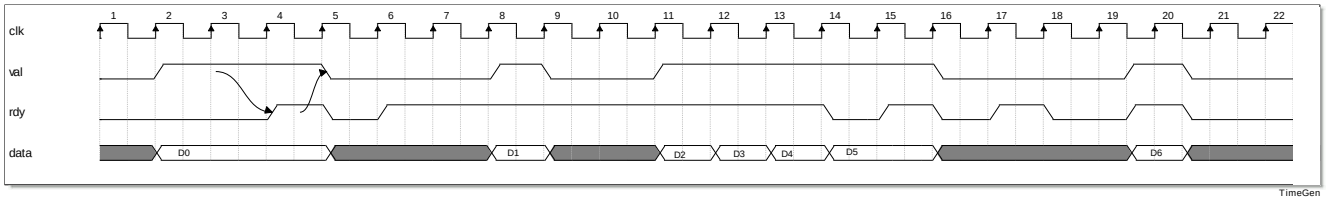
Se vor implementa facilități de verificare punctuală a unor înmulțiri, specificate de către proiectant.

Se vor verifica înmulțirea cazurilor de numere extreme.

Suplimentar, se va implementa posibilitatea de verificare a înmulțirii unor operatori aleatorii.

Protocolul “valid-ready”

Forme de undă caracteristice interfeței „**valid-ready**”.



Pe o interfață valid-ready datele sunt transferate unidirecțional, de la modulul emițător la modulul receptor.
Modulul receptor activează semnalul *rdy=1* dacă este capabil de a primi date.
Modulul emițător activează *val=1* dacă are date de transmis.
Transmiterea efectivă a datelor între modulele emițător și receptor se face în cazul în care *val=rdy=1*.
Modulul receptor are dreptul să activeze sau să dezactiveze orcând semnalul *rdy*.
Dacă *val=1* când *rdy=1*, emițătorul are obligația de a prelua datele.
Modulul emițător are dreptul să activeze oricând *vld=1*, însă poate dezactiva semnalul *vld* doar dacă *rdy=1*.
Simultan cu activarea *val=1*, emițătorul trebuie să depună datele *data*. Datele se mențin neschimbate de către emițător până la activarea *rdy=1*.
Schimbarea datelor când *val=1* și *rdy=0* reprezintă o eroare de protocol. O eroare de protocol este și inactivarea *vld* (trecerea din 1 în 0) când *rdy=0*.