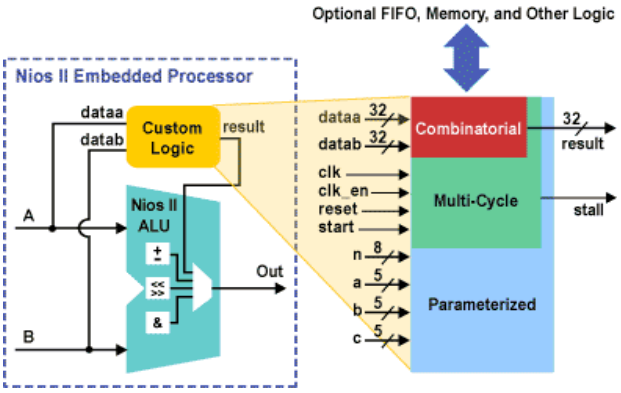
Spécialisation d'instructions sous NIOS-II

Daniel Etiemble
de@lri.fr

Spécialisation d'instructions avec NIOS II CPU



- La spécialisation d'instructions (customization) concerne essentiellement les instructions combinatoires et multi-cycles

Squelette programme C (1)

```
#include <stdio.h>
#include <stdlib.h>
#include "system.h"
#include "alt_types.h"
#include <time.h>
#include <unistd.h>
#include <sys/alt_timestamp.h>

// définition des instructions "ajoutées".
#define neg(a)          __builtin_custom_ini(4,a)
#define min32(a,b) __builtin_custom_inii(0,a,b)
#define min8(a,b) __builtin_custom_inii(1,a,b)
#define decg(a,b) __builtin_custom_inii(2,a,b)
#define decd(a,b) __builtin_custom_inii(3,a,b)
// etc...
```


Squelette programme C (2)

```
typedef union s32{
  unsigned char char_word[size][size];
  int int_word[size][size/4];
} s32 ;

//variables pour mesure de temps.
alt_u32 num_ticks ;
alt_u32 time1, time2, timer_overhead;

int main() {
//Initialisation du timer
  if(alt_timestamp_start() < 0) {
    printf("Timer init failed \n");
    exit(0);  }

// Nombre de cycles d'horloge liés à la prise de temps

    time1 = alt_timestamp();
    time2 = alt_timestamp();
    timer_overhead = time2 - time1;

for (k=0; k<5; k++) {
    time1 = alt_timestamp();

  //mesure de temps sur le programme (le programme est exécuté 5 fois)

    time2 = alt_timestamp();
num_ticks = time2 - time1 - timer_overhead; }

 return 0;
}
```

VHDL pour instruction « spécialisée » : entité

```
entity operation_instruction specialisee is
   port
   (
        dataa   :     in      std_logic_vector(31 downto 0);
        datab   :     in      std_logic_vector(31 downto 0);
        result  :     out     std_logic_vector(31 downto 0)

   );
end operation_instruction specialisee ;
```


VHDL pour instruction Min32

```
entity min32 is
   port
   (
        dataa :  in      std_logic_vector(31 downto 0);
        datab:   in      std_logic_vector(31 downto 0);
        result:  out     std_logic_vector(31 downto 0)          );
end min32;

architecture comp of min32 is
begin
result (31 downto 0) <= dataa (31 downto 0)
        when (dataa (31 downto 0) < datab (31 downto 0))
        else datab (31 downto 0);
end comp; -- end of architecture
```

Déclaration de bibliothèques

- Entiers non signés

```
-- IEEE Libraries --
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
--use IEEE.std_logic_signed.all;
```

- Entiers signés

```
-- IEEE Libraries --
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
--use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_signed.all;
```

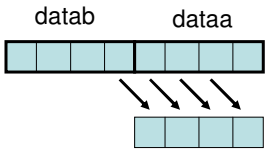

VHDL pour MIN32 SIMD

```
architecture comp of min32_simd is
begin
result (7 downto 0) <= dataa (7 downto 0)
        when (dataa (7 downto 0) < datab (7 downto 0))
        else datab (7 downto 0);
result (15 downto 8) <= dataa (15 downto 8)
        when (dataa (15 downto 8) < datab (15 downto 8))
        else datab (15 downto 8);
result (23 downto 16) <= dataa (23 downto 16)
        when (dataa (23 downto 16) < datab (23 downto 16))
        else datab (23 downto 16);
result (31 downto 24) <= dataa (31 downto 24)
        when (dataa (31 downto 24) < datab (31 downto 24))
        else datab (31 downto 24);
end comp; -- end of architecture
```

VHDL pour décalages gauche et droit

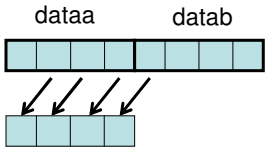
- **Décalage droit**

```
architecture comp of decd is
begin
result (7 downto 0) <=  dataa (15 downto 8);
result (15 downto 8) <= dataa (23 downto 16);
result (23 downto 16) <= dataa (31 downto 24);
result (31 downto 24) <= datab (7 downto 0);
end decd; -- end of architecture
```



- **Décalage gauche**

```
begin
result (7 downto 0) <=  datab (31 downto 24);
result (15 downto 8) <= dataa (7 downto 0);
result (23 downto 16) <= dataa (15 downto 8);
result (31 downto 24) <= dataa(23 downto 16);
end decg; -- end of architecture
```



SIMD – Extension 8 bits vers 16 bits

- Partie basse

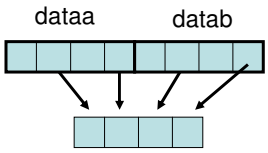
```
architecture comp of B2HL is
begin
result (7 downto 0) <=  dataa (7 downto 0);
result (15 downto 8) <= "00000000";
result (23 downto 16) <= dataa (15 downto 8);
result (31 downto 24) <= "00000000";
end comp; -- end of architecture
```

- Partie haute

```
architecture comp of B2HH is
begin
result (7 downto 0) <=  dataa (23 downto 16);
result (15 downto 8) <= "00000000";
result (23 downto 16) <= dataa (31 downto 24);
result (31 downto 24) <= "00000000";
end comp; -- end of architecture
```

SIMD – Réduction SIMD 16 bits vers 8 bits

- Sans saturation




```
architecture comp of H2B is
begin
    result (15 downto  8) <= datab (23  downto 16);
    result (7  downto  0) <= datab (7   downto  0);
    result (31 downto 24) <= dataa (23 downto 16);
    result (23 downto 16) <= dataa (7  downto 0);
end comp;
```

- Avec saturation

```
architecture comp of H2BS is
begin
    result (15 downto  8) <= datab (23  downto 16) when (datab (31 downto 24) = 0) else "11111111";
    result (7  downto  0) <= datab (7   downto  0) when (datab (15 downto 8) = 0) else "11111111";
    result (31 downto 24) <= dataa (23 downto 16) when (dataa (31 downto 24) = 0) else "11111111";
    result (23 downto 16) <= dataa (7  downto 0) when (dataa (15 downto 8) = 0) else "11111111";
end comp; -- end of architecture
```


Instruction à trouver

```
architecture comp of instr_x  is
signal t1p,t2p,t3p, t4p,t1m, t2m,t3m,t4m, t1,t2,t3,t4 : std_logic_vector (7 downto 0);
signal t5,t6 : std_logic_vector (8 downto 0);
signal t7 :std_logic_vector (9 downto 0);
begin
t1p <= dataa (7 downto 0)- datab (7 downto 0) when (dataa (7 downto 0)> datab (7 downto 0)) else '00000000';
t2p <= dataa (15 downto 8)- datab (15 downto 8)when (dataa (15 downto 8)> datab (15 downto 8)) else '00000000';
t3p <= dataa (23 downto 16)- datab (23 downto 16)when (dataa (23 downto 16)> datab (23 downto 16)) else '00000000';
t4p <= dataa (31 downto 24)- datab (31 downto 24)when (dataa (31 downto 24)> datab (31 downto 24)) else '00000000';
t1m <= datab (7 downto 0)- dataa (7 downto 0)when (dataa (7 downto 0)< datab (7 downto 0)) else '00000000';
t2m <= datab (15 downto 8)- dataa (15 downto 8) when (dataa (15 downto 8)< datab (15 downto 8)) else '00000000';
t3m <= datab (23 downto 16)- dataa (23 downto 16)when (dataa (23 downto 16)< datab (23 downto 16)) else '00000000';
t4m <= datab (31 downto 24)- dataa (31 downto 24)when (dataa (31 downto 24)< datab (31 downto 24)) else '00000000';
t1 <= t1p when (t1p > t1m) else t1m;
t2 <= t2p when (t2p > t2m) else t2m;
t3 <= t3p when (t3p > t3m) else t3m;
t4 <= t4p when (t4p > t4m) else t4m;
t5=('0'&t1)+('0'&t2);
t6=('0'&t3)+('0'&t4);
t7=('0'&t5)+('0'&t6);
result <= '000000000000000000000000'&t7;
end comp; -- end of architecture
```