Rapport technique de stage
Optimisation de division et multiplication par des constantes sous VIVADO HLS

Victor Lezaud

31 août 2018

Sommaire

LIEN VERS LE GITHUB :
https ://github.com/Victorlzd/High_Level_Synthesis_Trainee.git

1 Script

1.1 Build_operators

1.1.1 Explication

La commande Build_operators permet de générer un projet GeCoS contenant les opérateurs que l'on veut. Pour sortir ces opérateurs en c++, il suffit de lancer le script build_operators.cs. Pour choisir les operateurs générés il faut ajouter les méthodes de correspondantes dans la méthode compute de la classe Build_operators, sous le commentaire `//add method call here`. Les méthodes de générations se trouvent dans les classes Div et Mul de operator. Elles prennent en parametre un ProcedureSet (ici il suffit de laisser ps qui est bien définit dans) cette méthode) un entier qui définit la constante et un booléen qui définit le type (true pour `long/double` et false pour `int/float`).

1.1.2 code script build_operators.cs

Ce script génère les opérateurs et les écrit en c++ dans le dossier output :

```
project = Build_operators();

SetBitAccurateBackend("VivadoAP");
SaveGecosProject(project, "project_from_factories.gecosproject");
output(project, "c", "output");
```

1.1.3 Modification dans la classe Build_operators

Pour choisir les opérateurs générés il faut aller dans la classe Build_operators et modifier la methode compute sous le commentaire `// add method call here` le code d'exemple ci-dessous génère :

- diviseur de int par 3
- diviseur de long par 5
- diviseur de float par 7
- diviseur de double par 9
- multiplieur de int par 3
- multiplieur de long par 5
- multiplieur de float par 7
- multiplieur de double par 9

```
public GecosProject compute() {
GecosProject project = GecosUserCoreFactory.project("build_operators");
ProcedureSet ps = File_builder.create_ps();

// add method call here
Div.build_int_div_by_constant(ps, 3, false);
Div.build_int_div_by_constant(ps, 5, true);
Div.build_float_div_by_constant(ps, 7, false);
```

```
Div.build_float_div_by_constant(ps, 9, true);
Mul.build_int_mul_by_constant(ps, 3, false);
Mul.build_int_mul_by_constant(ps, 5, true);
Mul.build_float_mul_by_constant(ps, 7, false);
Mul.build_float_mul_by_constant(ps, 9, true);

return File_builder.add_files(project);
}
```

1.2 Replace_operators

1.2.1 Explication

Cette commande est celle qui est destinée à être untilisée par l'utilisateur final. Elle prend en parametre un projet GeCoS et remplace toutes multiplications et divisions par des constantes pat les opérateurs optimisés. Le projet est ensuite prêt à être retransformer en code C++, la commande ajoute les fichiers "optimized_operators.cpp" et "optimized_operators.h" au projet.

1.2.2 Exemple de script

Voici un exemple de script utilisant Replace_operators :

```
p = CreateGecosProject("example");
AddSourceToGecosProject(p, "input_c/example.c");
CDTFrontend(p);

p = Replace_operators(p);

SetBitAccurateBackend("VivadoAP");
GecosDAGToTreeIRConversion(p);
output(p, "c", "output");
```

2 Resultats

Les resultats ci-dessous ont été obtenu avec le kintex-7 : xc7k160tfbg484-1. La consigne de fréquence est de 400MHz (2.5ns).

2.1 Operateur

2.1.1 Division

Type	Constant	LUT	FF	DSP	BRAM	SRL	Cycles	Frequence
INT	HLS	144	219	4	0	0	9	430MHz
	3	67	93	0	0	0	23	470MHz
	5	97	101	0	0	0	32	450MHz
	6	71	91	0	0	0	23	460MHz
	7	97	101	0	0	0	32	450MHz
	9	114	115	0	0	0	47	430MHz
	10	95	99	0	0	0	32	430MHz
	11	114	115	0	0	0	47	420MHz
LONG	HLS							
	3	114	173	0	0	0	47	400MHz
	5	190	187	0	0	0	65	380MHz
	6	115	171	0	0	0	47	320MHz
	7	190	187	0	0	0	65	380MHz
	9	251	276	0	0	0	96	370MHz
	10	176	183	0	0	0	65	380MHz
	11	251	276	0	0	0	96	370MHz
FLOAT	HLS	713	835	0	0	35	29	460MHz
	2	89	158	0	0	0	5	590MHz
	3	314	352	0	0	0	28	430MHz
	4	138	185	0	0	0	5	600MHz
	5	341	356	0	0	0	35	430MHz
	6	319	351	0	0	0	28	500MHz
	7	340	356	0	0	0	35	450MHz
	9	357	382	0	0	0	50	500MHz
	10	344	366	0	0	0	38	420MHz
	11	356	382	0	0	0	50	500MHz
DOUBLE	HLS	3162	3295	0	0	105	58	350MHz
	2	289	290	0	0	53	8	520MHz
	3	759	885	0	0	6	52	400MHz
	4	396	398	0	0	54	8	500MHz
	5	797	905	0	0	6	68	390MHz
	6	756	878	0	0	6	52	400MHz
	7	799	913	0	0	6	68	360MHz
	9	880	1044	0	0	6	98	340MHz
	10	808	898	0	0	6	68	380MHz
	11	883	1058	0	0	6	98	340MHz

2.2 Stencils

2.2.1 Jacobi-1d-imper

	classic	my_version
LUT	3875	1378
FF	1970	1005
DSP	3	3
BRAM	0	0
SRL	64	0
Cycles	37×10^6	24×10^6
Frequency	114MHz	134MHz

2.2.2 Seidel-2d

	classic	my_version
LUT	4075	1586
FF	2074	1119
DSP	7	7
BRAM	0	0
SRL	65	0
Cycles	1.35×10^9	1.24×10^9
Frequency	119MHz	146MHz

2.2.3 Fdtd-2d

	classic	my_version
LUT	1398	1477
FF	1435	1556
DSP	17	17
BRAM	0	0
SRL	0	0
Cycles	3.65×10^9	3.25×10^9
Frequency	116MHz	117MHz

2.2.4 Jacobi-2d-imper

	classic	my_version
LUT	1118	1743
FF	1052	1072
DSP	18	7
BRAM	0	0
SRL	0	0
Cycles	717×10^6	677×10^6
Frequency	111MHz	79MHz