

کتابچه راهنمای نرم افزار فنی - مهندسی

DIgSILENT PowerFactory
Version 13.0

شرکت برق منطقه ای فارس

معاونت برنامه ریزی و تحقیقات

۱۳۸۴

تمیبه شده کار :

شیراز- فیابان زند- نبش فیابان فلسطین

شرکت برق منطقه ای فارس

تلفن : ۹-۲۳۳۰۰۳۱ (۰۷۱۱)

فاکس : ۲۳۵۹۰۴۷ (۰۷۱۱)

www.frec.co.ir

وزارت نیرو

تماس با مترجمان

مهمربنا کلساز شیرازی mshirazi@frec.co.ir

امیر فرشیان فسایی farshchian@frec.co.ir

حق چاپ و انتشار انحصاری

تمامی این ترجمه در شرکت برق منطقه ای فارس و با همکاری کارشناسان دفتر برنامه ریزی فنی و برآورد بار معاونت برنامه ریزی و تحقیقات تهیه شده است. بنابراین کلیه حقوق این ترجمه متعلق به شرکت برق منطقه ای فارس بوده و هرگونه نسخه برداری بدون کسب اجازه از این شرکت، ممنوع بوده و پیگرد قانونی دارد.

تابستان ۱۳۸۴ - شیراز

شرکت برق منطقه ای فارس

وزارت نیرو

Advanced User's Manual

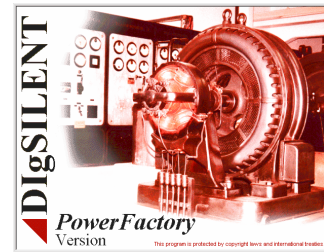
شرکت برق منطقه ای فارس

فهرست مطالب

1	Calculation of Transients	2
1.1	Introduction.....	2
1.2	Algorithms And Modeling.....	3
1.3	Calculation Of Initial Values.....	6
1.4	Run Simulation.....	9
1.5	Result Objects.....	9
1.6	Events.....	10
2	Models For Stability Analysis	12
2.1	System Modeling Approach.....	12
2.2	The Composite Model.....	14
2.3	The Common Model	18
2.4	The Composite Frame.....	20
2.5	The Composite Block Definition.....	21
2.6	Drawing Composite Block Diagrams and Composite Frames	23
2.7	The Block Definition	28
3	Programming Primitive Block Definitions	29
3.1	Modeling and Simulation Tools	31
3.2	DSL Implementation: an Introduction	32
3.3	Defining DSL Models.....	36
3.4	The DIgSILENT Simulation Language (DSL)	36
3.5	DSL Functions.....	47
4	Model Parameter Identification	50
4.1	Target Functions and Composite Frames.....	51
4.2	Creating The Composite Identification Model	53
4.3	Performing a Parameter Identification.....	55
4.4	Identifying Primary Appliances	57
5	The Medinas Monitoring System	60
5.1	Hardware Description	60
5.2	Basic Installatin and Operation	61
5.3	Measurment Principles.....	62
5.4	Configuring the Measurment Process	67
5.5	Performing Measurments	73
5.6	Result Objects.....	75
5.7	The Measurment Toolbar	80
5.8	Triggering.....	80
5.9	The Signal Processing Block Diagrams.....	83
6	Reliability Assessment Functions	89
6.1	Contingency Analysis	89
6.2	Stochastic Reliability Assessment : Basic Theory	95
6.3	Failure Models.....	100
6.4	Generation Reliability Assessment.....	121
6.5	Network Reliability Assessment	124

Chapter 3

Programming Primitive Block Definitions



مدل سازی شبکه به منظور آنالیز پایداری یکی از اصلی ترین مسائل در زمینه آنالیز شبکه قدرت می باشد. بسته به دقت مدل اجرا شده، اعتبار سیگنالهای فشارقوی با دامنه بزرگ، در دسترس بودن پارامترهای شبکه و آزمایشات یا خطاهای اعمال شده، تقریباً می توان هر نتیجه ای را ایجاد کرد و بنابراین باید پارامترهای (آرگومان های) مورد نیاز برای تحقق آنها را یافت.

یک مثال ساده می تواند این حقیقت را روشن کند. یک شبکه قدرت ۱۰ گیگاوات را در نظر بگیرید، سؤالی که مطرح است در مورد انحراف فرکانس حالت کار دائم سیستم است زمانیکه کل بار یک واحد ۲۰۰۰ مگاواتی را از دست داده باشیم، که این بستگی زیادی به ضریب وابستگی فرکانس بارها K_f دارد. فرض نمایید کل افت شبکه ۷ درصد و K_f برابر صفر باشد، آنگاه انحراف فرکانس حالت ماندگار تقریباً ۷۰۰ میلی هرتز خواهد شد. حال با در نظر گرفتن یک ضریب واقعی $K_f = 5\% / \text{Hz}$ ، انحراف فرکانس حالت ماندگار مورد انتظار تنها ۵۹۶ میلی هرتز می باشد. از طرف دیگر، وابستگی فرکانس ممکن است کمی بالاتر یا پایین تر باشد، اما مشخصات غیرخطی راندمان توربین آبی و رفتار غیرخطی سوپاپ بخار می تواند در یک نقطه کار معین از ظرفیت بارگیری واحد، وابستگی شدیدتری به یکدیگر داشته باشند. در نتیجه مادامی که فقط یک یا دو سناریوی بارگیری مختلف در نظر گرفته شود، مقادیر متوسط با مدل های ساده منطقی ممکن است نتایج قابل قبولی را ایجاد نمایند که بوسیله تنظیم کردن فقط تعدادی پارامتر کلیدی مثل ضرایب وابستگی فرکانس بارها یا تنظیمات افت بدست آمده اند.

این مسئله را بخاطر داشته باشید که ساختار های مدل شبکه و تنظیمات پارامتر باید با توجه به معیارهای اصلی زیر محاسبه شوند :

System size (اندازه شبکه) : شبکه های کوچک و بزرگ دارای پارامترهای کلیدی مختلفی هستند. به مثال بالا رجوع کنید، برای یک شبکه کوچکتر قدرت وابستگی فرکانس بارها به طور کلی مطرح نمی باشد در حالی که در شبکه های بزرگ مثل *UCPTE* یا *UPS* وابستگی فرکانس ممکن است به اندازه کل نیاز ظرفیت ذخیره چرخشی شبکه تحت پوشش باشد.

Unit size (اندازه واحد) : رفتار حالت گذرا و حالت ماندگار واحدهای بزرگ در تعیین پاسخ کل شبکه بسیار نقش تأثیرگذارتری نسبت به واحدهای کوچکتر که عملاً تأثیری بر کل شبکه نمی گذارند، دارند.

System structure (ساختار شبکه) : مستقل از شبکه و اندازه واحد، ساختار شبکه پیش از هر عامل دیگری مطرح می باشد. این مسئله را وقتی که شبکه های ضعیف گسترده در طول جغرافیایی یا دارای زیرساختهای اختصاصی آنالیز می شوند، می توان به راحتی درک کرد.

System fault (اختلال شبکه) : عمده ترین مسائل مطرح در مدل سازی شبکه، خطاهای اعمال شده و مسائل مربوطه است که باید مورد نقد و بررسی قرار گیرند. آنالیز میرایی شبکه و تنظیم PSS لزوماً نیازی به در نظر گرفتن دینامیک بویلر ندارد. از طرف دیگر اگر منحنی های دراز مدت یا میان مدت

عناصر وابسته شبکه حذف شده باشند و یا چشم پوشی گردند، بهینه سازی مانور قطع بار و بازیابی فرکانس، نتایج مناسبی را ایجاد نخواهند کرد.

Study purpose (هدف مطالعه): بطور کلی برای شبکه هایی که در مرحله طراحی هستند، مادامیکه هیچ اطلاعات خاص اضافه دیگری موجود نباشد می توان از مدلها و پارامترهای نمونه استفاده نمود. به هر حال در موارد توسعه و گسترش شبکه لازم است تا از مدلهاى دقیق تر استفاده گردد. یعنی جایی که مدل دقیق تر باید بخشی از مشخصات شبکه را تشکیل دهد، به آنالیز مسائل بهره برداری و بهینه سازی عملکرد باید توجه خاصی مبذول گردد. در این موارد، مدل سازی دقیق مؤلفه های مربوطه ضروری و لازم الاجرا می باشد.

به محض این که یک آنالیز و ارائه دقیق مدلهاى سیستم لازم باشد، اولین سوالاتی که بلافاصله مطرح می شوند، عبارتند از :

- چگونه می توان ساختارها و پارامترهای مدل را تعیین کرد؟
- آیا مدلهاى *IEEE* و سایر بلوک دیاگرام های سازندگان تجهیزات شبکه، دقیق و مناسب هستند؟
- چطور می توان اطلاعات موجود را در داخل نرم افزار آنالیز شبکه قدرت استفاده کرد؟

روشی که در این جا ارائه شده و به صورت موفقیت آمیزی در پروژه های مختلف بکار رفته است می توان *'Advanced System Modeling Approach (ASMA)* نامید. کاربردهای نمونه آن عبارتند از :

- آنالیز مشکلات کنترل کننده و عملکردهای غلط آن، مخصوصاً تحت شرایط اغتشاش
- بهینه سازی تنظیمات پارامتر کنترل
- مدل سازی ساختارهای غیر عادی شبکه و مفاهیم کنترل که اغلب در سیستم های صنعتی یافت می شوند.
- انجام مطالعات فاز صفر طراحی و ارائه مشخصات اجزاء و سیستم ها (از قبیل *PSS*، کنترل کننده ولتاژ و *HVDC*)

برای روش *ASMA* مراحل زیر دارای اهمیت بنیادی می باشند :

setup of system models (نصب مدلهاى شبکه): بر اساس معادلات اساسی مهندسی و فیزیکی، معادلات جبری و دیفرانسیل پایه، باید در محدوده دقت مورد احتیاج تنظیم شوند. بعلاوه همه پارامترها مانند ثابت زمانی و ضرایب بهره که می توانند از این معادلات پایه بدست آیند، نیز باید به همان شیوه محاسبه گردند.

performance of system tests (تست های اجرای سیستم): به منظور تعریف همه پارامترهای دیگر بخصوص منحنی های غیرخطی، تست های اجرای شبکه، بهترین روش می باشند. در اکثر موارد، آزمایشات پاسخ فرکانسی اجازه تعیین هر ساختار غیرخطی و پارامترهای آن را نمی دهند. روشهای تست ویژه، که با عملکرد عادی تداخلی نداشته باشند باید به منظور تمرکز بر روی مشخصات حالت ماندگار، ضرایب بهره و ثابت زمانی ها بکار برده شوند. این اندازه گیریها ترجیحاً توسط یک سیستم اندازه گیری دیجیتال حالت گذرای بسیار دقیق انجام می شوند.

system identification (شناسایی شبکه): روشهای شناسایی غیر خطی، چند ورودی و چند خروجی برای روشهای شناسایی شبکه بکار برده می‌شوند. برای نمونه، عدم تطابق بین منحنی‌های اندازه‌گیری شده و شناسایی شده باید کمتر از ۲٪ باشد.

Comparison of measurements and simulations (مقایسه بین اندازه‌گیری‌ها و شبیه‌سازی‌ها): علاوه بر آنالیز زیر شبکه‌ها و مؤلفه‌ها، عملکرد کل شبکه نیز باید با مدل تئوری و برای همه حالت‌های بهره‌برداری مربوطه مقایسه شود.

البته کاربرد روش 'ASMA' به شکل کاملاً جدی و سختگیرانه برای مدلسازی رله‌ها و برای پیچیدگی‌های کمتر یا توابع کنترل دیجیتالی لازم نمی‌باشد، زیرا اینها به صورت واضح مطابق با اسناد آزمایشات تأیید شده و کلی مربوط به خودشان تعریف شده‌اند. اما مستقل از شبکه آنالیز شده، در جایی که نمایش شبکه نمی‌تواند با مدل‌های کلاسیک *IEEE* یا هر مدل استاندارد دیگر تطابق داشته باشد، نیاز اساسی به استفاده از یک روش آسان و قابل انعطاف برای درک تمام مدل‌های اختصاصی وجود دارد.

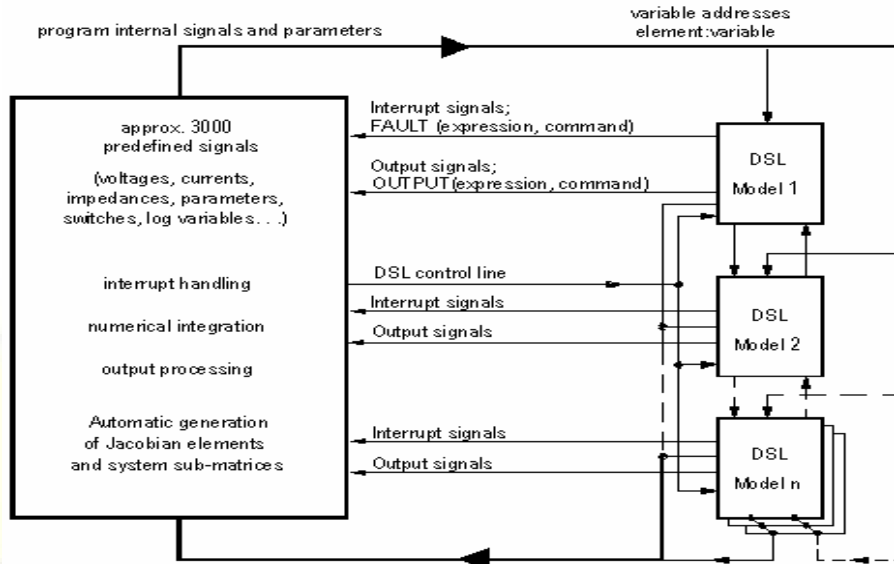
3.1 Modelling and Simulation Tools

همانطور که قبلاً بیان شد، ضروری‌ترین و تعیین‌کننده‌ترین عامل برای دستیابی به نتایج شبیه‌سازی مطمئن، دقت و تکامل مدل‌های ارائه شده برای شناسایی شبکه و اهداف شبیه‌سازی می‌باشد. روشهای انجام این خواسته متغیر بوده و از روشهای کلاسیک و سنتی استفاده از نرم‌افزارها که به عنوان محیط واسطه برای مدل‌های تعریف شده توسط کاربر عمل می‌کردند و در سطح نرم‌افزارهای فترن و C "معمولاً به شکل فهرست‌های اتصال" بودند، تا استفاده از روش‌های بلوکی که بر اساس تهیه بلوک ماکروهای سطح پایین از پیش تعریف شده عمل کرده و به سطح تعریف مورد متصل می‌باشند، گسترده می‌باشند. بعلاوه اغلب ابزارهای شبیه‌سازی مدرن تجارتي همه منظوره موجود، ممکن است برای نمایش ویژه و انعطاف پذیر شبکه استفاده گردند. متأسفانه روش منحنی‌های پخش بار، شبکه‌های الکتریکی خاص را به حد کافی پوشش نمی‌دهد.

به منظور ایجاد یک ابزار شبیه‌سازی و مدل‌سازی قابل انعطاف، که قسمتی از برنامه پایداری را تشکیل می‌دهد، یک سیستم کنترل مبتنی بر زبان شبیه‌سازی ایجاد شده است. که ویژگی‌های اصلی زیر از زبان شبیه‌سازی *DIgSILENT* در آن منظور شده اند:

- ابزار شبیه‌سازی در طبقه یک زبان شبیه‌سازی پیوسته شبکه (*CSSL*) قرار داده می‌شود.
- *DSL* شامل یک توصیف ریاضی کامل از شبکه‌های خطی و غیرخطی پیوسته (زمانی) می‌باشد.
- ابزار شبیه‌سازی که برپایه کنترل مشترک و نمودارهای منطقی بنا شده است به یک زبان بدون رویه سوق داده می‌شود که در آن توالی عناصر را می‌توان به دلخواه انتخاب کرد. به عبارت دیگر، یک مدل *DSL* می‌تواند به یک ارائه گرافیکی تبدیل شود.
- تهیه و ایجاد یک تعریف انعطاف پذیر از ماکروها که می‌تواند: معادلات جبری، عناصر اصلی کنترل مانند *PID* یا *PTn* یا حتی زیر سیستم‌های فیزیکی کامل مثل انواع گروه‌های سوپاپ یا سیستم‌های تحریک کننده را در برگیرد.

- تهیه توابع درونی مختلف مثل: "select" , "lim" , "limits" , "lapprox" , "picdrop" به منظور فراهم کردن یک کنترل کامل بر مدلها.
- ایجاد روش های رسمی مختلف برای شناسایی خطا و اهداف آزمایشی مثل: شناسایی حلقه جبری، گزارش متغیرهای بکار نرفته و تعریف نشده و شرایط اولیه حذف شده.



شکل ۳-۱: ساختار سیستم Digilent DSL

3.2 DSL Implementation: an Introduction

زبان شبیه سازی DigSILENT به منظور تعریف کنترل کننده های دینامیکی جدید که سیگنالهای ورودی را از شبکه قدرت شبیه سازی شده دریافت کرده و با تغییر برخی سیگنال های دیگر عکس العمل نشان می دهد، استفاده می شود.

DSL خودش می تواند به عنوان یک ابزار اضافه شونده به تابع آنالیز گذرای DigSILENT اضافه شود. در زمان شبیه سازی، معادلات نمونه مدل های DSL با آنهایی که رفتار دینامیکی مؤلفه های شبکه قدرت را توصیف می کنند، ترکیب می شوند. سپس این معادلات بصورت توأمان حل گردیده و به یک شبیه ساز گذرای جامع و یکپارچه مجموعه شبکه قدرت و کنترلرهای آن تبدیل می شوند.

توابع واسطه اصلی DSL عبارتند از:

Signal input and output channels (کانل های ورودی و خروجی سیگنال): هر متغیر تعریف شده در برنامه اصلی (در واقع بیش از ۲۵۰۰) و در مدل DSL، می تواند قابلیت خواندن و نوشتن داشته باشد. ویژگیهای آدرس اصلی و فرعی برای امکان دسترسی به هر سیگنال موجود در سیستم یا جمع کردن ساختارهای پیچیده مثل مازول های سخت افزاری تجهیزات دربرگیرنده تجهیزات rack (قفسه) و Function card (کارت تابع)، ذکر شده است.

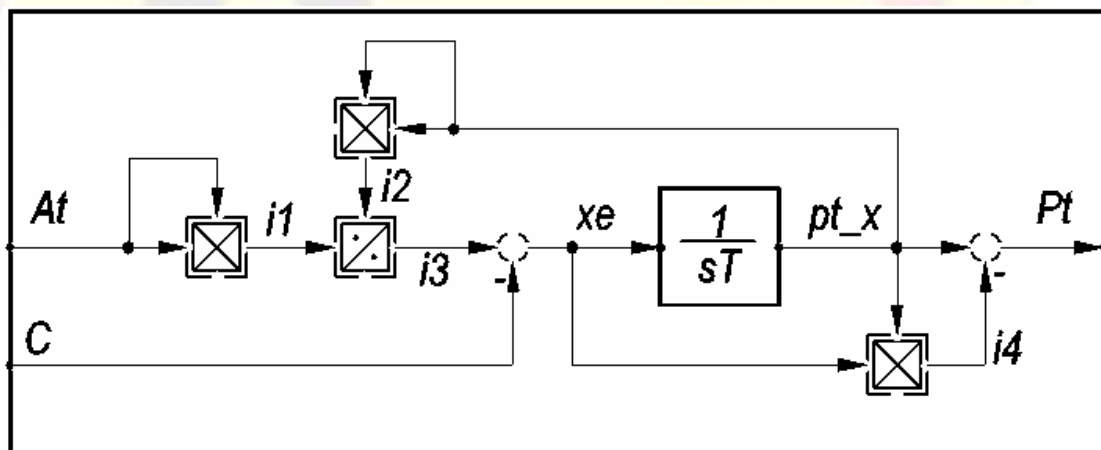
Events (رخدادها): شرایط ارزیابی شده بوسیله مدل‌های *DSL* ممکن است باعث ارسال رخدادها به برنامه مرکزی شوند جایی که آنها باید به نوبت ارزیابی گردند.

Output and Monitoring (خروجی و نظارت): شرایطی که منجر به فعال سازی نمایش پیامهای تعریف شده توسط کاربر در پنجره خروجی می شود.

ساختار مدل *DSL* از طریق مثال به خوبی توصیف گردیده است. مثال استفاده شده مربوط به مدل واحد محرک اولیه یک توربین آبی ساده می باشد. این مدل *DSL* به صورت گرافیکی تعریف شده است، و دارای یک ماکروی جاگذاری شده *DSL* می باشد. آن ماکروی جاگذاری شده یک انتگرالگیر ساده را مدل می کند و با برنامه نویسی آن تعریف شده است. روش اصلی طراحی مدل‌های جدید *DSL* عبارت است از:

۱- یک مجموعه از مدل‌های *DSL* پایه ایجاد شده است. این مدل‌ها شامل کنترل کننده های ابتدایی ساده مثل یک کنترل کننده تأخیری مرتبه اول (*First order time lag*) یا یک کنترل کننده *PID* می باشند. برنامه *Digsilent* به همراه تعداد زیادی از مدل‌های کنترل کننده های ابتدایی برای کاربران ارسال می شود. کنترل کننده های ابتدایی جدید بوسیله برنامه نویسی معادلات دیفرانسیل آنها و تنظیمات سیگنال با استفاده از زبان *DSL* ایجاد می شوند.

۲- کنترل کننده های پیچیده تر به صورت گرافیکی بوسیله ترسیم بلوک دیاگرام آن ایجاد می شوند. چنین بلوک دیاگرامی معمولاً از ارجاعاتی برای مدل‌های دیگر *DSL* استفاده می کند، که بنابراین یک کنترل کننده پیچیده تر را ایجاد می نمایند. ارجاعات کنترل کننده را می توان برای گنجاندن مدل‌های ابتدایی *DSL* در مدل‌های پیچیده نیز استفاده کرد، اما همچنین ممکن است به دیگر مدل‌های پیچیده تعریف شده گرافیکی هم ارجاع داده شوند. بنابراین ممکن است کنترل کننده های بسیار پیچیده در یک روش سلسله مراتبی بوسیله طراحی مدل‌های فرعی و زیرمجموعه های فرعی تر آنها تا جایی که پایین ترین سطح آن را کنترل کننده های ابتدایی *DSL* تشکیل می دهد، طراحی شوند.



شکل ۲-۳: نمودار مدل ساده یک توربین آبی

شکل ۲-۳، نموداری را نشان می دهد که برای تعریف مدل توربین آبی استفاده شده است. وقتی یک *graphics rebuild* اجرا می شود، نظیر آنچه در پنجره خروجی نشان داده شده است، کد *DSL* متوجه بوجود خواهد آمد:

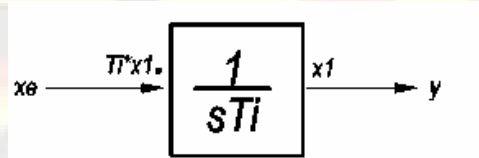
1. model pt = 'pmu - hydro' (At , C; x1;Ti;)

2. $pt - x = 'I. BlkDef' (xe ; x1 ; Ti);$
3. $i3 = i1 / i2$
4. $i1 = At * At$
5. $i2 = pt - x * pt - x$
6. $i4 = xe * pt - x$
7. $xe = i3 - c$
8. $pt = pt - x - i4$

شماره های خطوط برای سهولت عمل اضافه شده اند. تعریف بلوک متناظر نشان می‌دهد که :

Output signals	: pt	سیگنال خروجی
input signal	: At ,C	سیگنال ورودی
State variables	: X1	متغیرهای وضعیت
Parameter	: Ti	پارامتر
Internal variables	:	متغیرهای داخلی

این مثال، مدل یک توربین آبی ساده را با سیگنال های ورودی At و C و سیگنال خروجی Pt توصیف می‌کند.



شکل ۳-۳: نمایش گرافیکی یک مدل DSL یک انتگرالگیر

در شکل ۳-۳، نمایش گرافیکی مدل DSL ابتدایی جاگذاری شده ترسیم شده است. این مدل ابتدایی در توربین آبی بکار رفته است (در سطر ۲ از تعریف توربین). این DSL ابتدایی از یک انتگرالگیر تکی استفاده نموده است و به شکل زیر برنامه نویسی شده است :

1. $model y = 'I' (xe ; x1 ; Ti);$
2. $[Ti] = 's'$
3. $limits (Ti) = (0,)$
4. $inc (x1) = y$
5. $inc (xe) = 0$
6. $x1. = xe / Ti$
7. $y = x1$

خط ۱ با فشار دادن دکمه **Equations** از کادر محاوره ای ویرایش بلوک دیاگرام ایجاد شده است.

خط ۲ تا ۷ به صورت دستی وارد شده اند.

کادر محاوره ای تعریف بلوک برای تنظیم موارد زیر استفاده شده است :

Output signals	: y	Input signals	: xe
State variables	: x1		
Parameter	: Ti		
Internal Variables	:		

۳-۲-۱ قسمتهایی از یک مدل DSL

هر دو مثال *DSL* دو قسمت اصلی هر مدل *DSL*، ابتدایی یا پیچیده را نشان می‌دهند:

۱. تعاریف واسطه

۲. توصیف مدل *DSL*

Interface description (توصیف محیط واسطه): واسطه نام مدل، نام سیگنال‌های ورودی و

خروجی، پارامترهای مدل و متغیرهای حالت را تعریف می‌نماید. این موارد در پنجره خروجی در عنوان مدل نشان داده شده‌اند.

مثال (سطر ۱ از مدل توربین آبی):

`model pt = pmu - hydro (At , C;x1 ; Ti ;)`

کادر محاوره‌ای بلوک دیاگرام افزون بر این اجازه تعریف پارامترهای محدود سازی و سیگنال‌های ورودی، و طبقه‌بندی مدل به عنوان یک مدل خطی و یا یک ماکروی *DSL* را امکان پذیر نموده است.

Model description (توصیف مدل): توصیف مدل، مدل *DSL* را بر اساس سیگنال‌های

تعریف شده در واسطه وصف کرده و تشریح می‌نماید. توصیف *DSL* شامل موارد زیر می‌باشد:

- توصیفات پارامتر: نام و واحد
- دامنه‌های مجاز پارامتر
- شرایط اولیه و توابعی که برای محاسبه مقادیر اولیه استفاده می‌شوند.
- روابط جبری که کنترل کننده را تعریف می‌کنند.

مثال (انتگرالگیر):

2. `[Ti] = ' s ' ! the unit of Ti is seconds`
3. `limits (Ti) = (0)! Ti > 0`
4. `inc (x1) = y ! initially x1 = y`
5. `inc (xe) = 0 ! initially xe = 0`
6. `x1. = xe / Ti ! equation 1 : &*1/ &t = xe / Ti`
7. `y = x1 ! equation 2 : y = x1`

۳-۲-۲ ویژگیهای پیشرفته

انتگرال گیری عددی مدل‌های 'DSL'، زمان بندی قطع و برنامه ریزی سیگنال ورودی- خروجی بطور خودکار بوسیله برنامه مرکزی اداره می‌شود. به علاوه اگر خروجی یک مدل 'DSL' یک جریان الکتریکی است که به جریان باس کلی مناسب اضافه می‌شود در این حالت اگر یک بار یا ژنراتور ایجاد شود، همه عناصر لازم جاکوبین برای روش همانند سازی تکرار بطور خودکار محاسبه می‌شوند.

ویژگی مفید دیگر 'DSL' الگوریتم انجام شده برای تنظیم عددی ماتریس شبکه برای اهداف محاسبه مقدار 'eigen' می‌باشد. در نتیجه وقتی مقادیر 'eigen' شبکه محاسبه شده یا زمانی که روش کاهش شبکه مدل به کار می‌رود (MRT) هر مدل انجام شده در سطح 'DSL' بطور خودکار در نظر گرفته می‌شود. البته هر تابع محدود سازی سیگنال بطور خودکار برای این روش محاسبه غیر فعال می‌شود. به علاوه

ورودیها و خروجیهای پارامترهای مدل، تشکیلات آن را از طریق منوهای پنجره ها و غیره هم بطور خودکار از این مدل 'DSL' گرفته می شود.

3.3 Defining DSL Models

یک مدل *DSL* جدید را هم می توان به وسیله وارد کردن کد *DSL* در قسمت معادله عنصر *Block definition* یعنی *BlkDef* ساخت و هم می توان یک بلوک دیاگرام گرافیکی جدید ایجاد کرد. هر دو روش یک عنصر تعریف بلوک را ایجاد خواهند نمود که در برگزیده تعریف مدل *DSL* است.

بنابراین عناصر تعریف بلوک در فرآیند ساختن یک مدل *DSL* دو هدف زیر را دنبال می کنند :

- آنها لوازم و مقدماتی را ایجاد می نمایند که در آن می توان یک *DSL primitive* جدید را تعریف نمود.
- آنها تعاریف و بخش هایی از بلوک دیاگرام مرکب ساخته شده به صورت گرافیکی و گرافیک نموداری را که برای تعریف مدل استفاده شده است، نگهداری می کنند.

DSL ابتدایی و اصلی، بلوکهایی هستند که از آنها نمودارهای کنترل کننده های بسیار پیچیده ساخته می شود. برای مثال، یک *DSL* ابتدایی، ممکن است از یک فیلتر پایین گذر استفاده کرده باشد، که می تواند بعداً برای ساخت کنترل کننده های بسیار پیچیده تر که دارای چنین فیلتری هستند، استفاده شود. برخلاف ترانسفورماتورها یا سایر اجزاء سیستم های قدرت، که ممکن است شبیه به بلوک های ابتدایی شبکه قدرت باشند، یک *DSL* ابتدایی است فقط بوسیله بلوک دیاگرام های پیچیده تر ارجاع داده می شود و علاوه بر این می تواند در یک زمان در بیش از یک مدل *DSL* پیچیده نیز استفاده گردد.

عناصر تعریف بلوک پیچیده را می توان با "*Grid Folder*" (پوشه های شبکه) در درخت پایگاه داده های *DIgSILENT* مقایسه کرد. آنها از طریق تعریف گرافیکی یک بلوک دیاگرام کنترل کننده که در آن اطلاعات گرافیکی و تمام قسمت های منطقی را ذخیره می کند، مشخص می گردند. این قسمت ها شامل سیگنال ها، مولفه های استاندارد کوچک (آدرس، ضرب کننده ها و غیره) یا *DSL* ابتدایی می باشند. گرچه یک عنصر تعریف بلوک پیچیده به صورت گرافیکی ایجاد می شود، این امکان نیز مهیاست تا سایر معادلات *DSL* را که وارد کردن آنها در روش گرافیکی دشوار است، جداگانه تعریف نمود. در این جا به محیط های گرافیکی که در آنها یک بلوک دیاگرام پیچیده ساخته می شود، پرداخته نشده است. برای کسب اطلاعات بیشتر به *User's Manual* مراجعه کنید.

3.4 The DIgSILENT Simulation Language (DSL)

زبان 'DSL' برای برنامه ریزی مدلهایی برای کنترل کننده های الکتریکی و سایر مولفه های استفاده شده در شبکه های قدرت الکتریکی، استفاده شده است. مانند هر شبیه سازی و زبان برنامه نویسی دیگر، یک گرامر خاص برای فرمول نویسی مدل فراهم شده است.

این گرامر به ترتیب زیر توضیح داده شده است :

أ : اصطلاحات و اختصارات

ب : گرامر عمومی *DSL*

ج : متغیرهای *DSL*

د : ساختار *DSL*

ه : گرامر *DSL*

و : پیش فرض ها

۳-۴-۱ اصطلاحات و اختصارات

اصطلاحات و اختصارات زیر برای توصیف گرامر *DSL* استفاده شده اند :

expr : عبارت محاسباتی ریاضی که نباید به علامت '؛' ختم شود.

• عملگرهای محاسباتی / , * , - , + .

• ثوابت : همه اعدادی که به عنوان اعداد حقیقی عمل می کنند.

• توابع استاندارد :

$\sin(x)$, $\cos(x)$, $\tan(x)$, $\text{asin}(x)$, $\text{acos}(x)$, $\text{atan}(x)$, $\sinh(x)$, $\cosh(x)$, $\tanh(x)$, $\exp(x)$, $\ln(x)$, $\log(x)$ [basis 10] , $\text{sqrt}(x)$ [square root] , $\text{sqr}(x)$ [power of 2] , $\text{pow}(x, y)$, $\text{abs}(x)$, $\text{min}(x,y)$, $\text{max}(x,y)$, $\text{modulo}(x, y)$, $\text{trunc}(x)$, $\text{frac}(x)$, $\text{round}(x)$, $\text{ceil}(x)$, $\text{floor}(x)$.

این توابع استاندارد بطور مفصل در بخش بعدی تشریح شده اند.

• پرانتزها : (عبارات محاسباتی)

همه توابع مثلثاتی بر اساس رادیان ها می باشند (*RAD*).

مثال : $A = x1 + 2.45 * T1 / \sin(3.14 * y)$

boolexpr : عبارت منطقی که نباید به علامت '؛' ختم شود.

• روابط منطقی : = , <= , >= , < > (inequality) , < , > .

• عملکردهای یکانی : .not

• عملکردهای باینری (دوتایی) : .eor .nor .nand .or .and

• پرانتزها : (عبارت منطقی).

مثال : $A = \{ x1 > 0 . \text{and} . . \text{not} . x2 \leq 0.7 \} . \text{or} . T1 = 0.0$

string : هر چیز که بین علامت ' ' قرار گیرد.

مثال : $A = ' \text{this is a string} '$

۳-۴-۲ گرامر عمومی *DSL*

Line Length (طول خط) : حداکثر طول خط ۸۰ کاراکتر می باشد. خطوط طولانی تر را باید با

استفاده از علامت & در ستون اول به دو یا چند بخش شکست و تقسیم نمود. علامت & در ابتدای ستون باعث می گردد تا سطر جاری و سطر پیشین به یکدیگر متصل گردند.

مثال :

```
y = lapprox ( x , 1.674 , 7.367 , 2.485 , 12.479 , 5.457 , 18.578
&          6.783 , 15.54 , 8.453 , 12.589 , 9 , 569 , 6.478 )
```

قاعده شکستن خط را نمی‌توان در خصوص نامها یا رشته‌ها استفاده کرد.

Case sensitivity (حساسیت به کوچکی و بزرگی حروف) : تمام واژه‌های کلیدی، نامها، توابع،

متغیرها، مدلها، ماکروها و .. نسبت به حروف حساس هستند.

Blanks (جای خالی) : وقتی کد *DSL* پردازش می‌شود، تمام جاهای خالی برداشته می‌شوند. بجز

در رشته‌ها که این جاهای خالی حفظ می‌شوند.

Comments (توضیحات) : علامت ! باعث می‌شود تا محتویات سطری که این علامت را دارد به

عنوان توضیح منظور شود. وقتی کد *DSL* پردازش شود، تمام توضیحات برداشته خواهند شد. مثال :

! comments may start at the beginning of a line

x1. = select (at <> 0 , ! comments may be used in broken lines

```
& (1 - sqrt(x1) / sqrt(at) / Tw , 0)
```

۳-۴-۳ متغیرهای *DSL*

یک مدل *DSL* می‌تواند از ۵ نوع متغیر استفاده کند :

Output signals (سیگنال‌های خروجی) : در مدل‌های پیچیده تر از سیگنال‌های خروجی به

عنوان ورودی استفاده می‌شود.

Input Signals (سیگنال‌های ورودی) : متغیرهای ورودی ممکن است از مدل‌های دیگر *DSL* یا

از یک عنصر سیستم قدرت بدست آمده باشند. در مورد دوم، می‌توان از ولتاژها و جریان‌ها، و هر

سیگنال موجود دیگری که از آنالیز شبکه قدرت بوجود آمده باشد، در مدل *DSL* استفاده نمود.

State variables (متغیرهای حالت) : متغیرهای حالت، سیگنال‌های وابسته به زمانی می‌باشند که

در خود مدل *DSL* ایجاد شده و مورد استفاده قرار می‌گیرند.

Parameters (پارامترها) : پارامترها اعداد فقط خواندنی هستند که به منظور تغییر دادن رفتار مدل

DSL تنظیم می‌شوند.

Internal variables (متغیرهای داخلی) : متغیرهای داخلی تعریف شده و در مدل *DSL* استفاده

شده‌اند و سبب تسهیل در ساختن مجموعه معادلات *DSL* می‌شوند.

نکات :

- یک متغیر حالت نمی‌تواند در یک زمان به عنوان متغیر خروجی هم باشد؛ اگر چنین امری لازم باشد استفاده از یک متغیر جانشین مثل $y = x1$ توصیه می‌شود.
- همه پارامترها اعداد واقعی هستند.
- یک پارامتر خاص 'آرایه - iiiii' (با حداکثر ۴ رقم i)، با تعداد $2 * iiiii$ عنصر به منظور تعریف منحنی‌ها در دسترس می‌باشد (به روش "lapprox" نگاه کنید).
- تنها به مشتق‌های متغیرهای حالت می‌توان عبارتی را اختصاص داد.

۳-۴-۴ ساختار *DSL*

مدلهای *DSL* از سه قسمت ساخته شده اند :

- قسمت واسطه، که نام مدل، عنوان، طبقه بندی و مجموعه متغیرها را بیان می کند. این قسمت در صفحه اول کادر محاوره ای ویرایش بلوک دیاگرام تنظیم شده است.
- کد تعریف
- کد معادله

کد معادله و کد تعریف، شبکه کنترل کننده واقعی را به وجود آورده و در بخش های بعدی مورد بررسی قرار خواهند گرفت.

۳-۴-۵ کد تعریف

یک **کد تعریف** در قسمت معادله یک مدل *DSL* برای تعریف مشخصه های پارامتر و شرایط اولیه استفاده می شود.

واحد و توصیف

`vardef (varnm) = unitstring;namestring` واحد و نام برای متغیر *varnm*

مثالها :

`vardef (Ton) = 's' ; ' pick up time for restart ' ! defines unit and name`

`vardef (Ton) = ; ' pick up time for restart ' ! only defines name.`

`vardef (Ton) = ' s ' ; ! only defines unit`

`[varnm] = unitstring` : واحد برای متغیر *varnm* حداکثر تا پهنای ۱۰ کاراکتر.

مثال :

`[Ton] = 's'` واحد را تعریف می کند!

تذکر : اگر واحد متغیرهای جانشین (جایگذاری شده) با واحدهای تعریف شده مطابقت نداشته باشند، بوسیله یک ماکرو فراخوانده شده می توان پیغام های خطا را ایجاد نمود.

محدوده مقادیر معتبر

`Limits (varnm) = [(minimum value , maximum value] /)` : فاصله مجاز برای

متغیر *varnm* را تعریف می کند. نوسانات حدود فاصله در طی عمل شبیه سازی گزارش خواهد شد :

`limits (yt) = (,1)`

هم ارز است با

`output (yt > 1 , 'Maximum exceeded: yt = yt>1')`

پارانتزهای "(" و ")" در هر سمت محدوده تعریف فاصله که قرار گیرند آن حد (مقدار حداقل یا حداکثر) را مستثنی می کنند. و در مقابل براکتهای '[' و ']' در برگیرنده مقادیر کرانه ای فاصله خواهند بود.

مثالها :

$\text{limits}(x) = [\text{min}, \text{max}] ! \text{min} \leq x \leq \text{max}$
 $\text{limits}(x) = [\text{min}, \text{max}] ! \text{min} < x \leq \text{max}$
 $\text{limits}(x) = (, \text{max}) ! x \leq \text{max}$
 $\text{limits}(x) = (\text{min},) ! \text{min} < x$

اگر لازم و ممکن باشد، برنامه بطور خودکار کوچکترین فاصله را در بین چندین فاصله از همان متغیرها را تعیین می‌کند. برای مثال: $\text{limits}(x)=(1,3)$ و $\text{limits}(x)=(2,4)$ به معنی $2 < x < 3$. مدلهای ماکرو اغلب حدود متغیرهای معینی را تعریف می‌نمایند. مدلی که از ماکرو استفاده می‌کند همچنین ممکن است حدود متغیرهایی را که در فراخوانی های ماکرو استفاده شده اند را نیز تعریف کند. روش "*Smallest interval*" (کوچکترین فاصله) به مدل فراخوانده شده این اجازه را می‌داد که حدود پارامتر را بدون ایجاد اختلال در تعاریف حدود ماکروی داخلی، مجدداً تعیین نماید.

شرایط اولیه

تنظیم مستقیم

$\text{Inc}(\text{varnm}) = \text{expr}$: تعریف شرایط اولیه متغیر *varnm*. اگر $\text{inc}(\text{varnm})$ تعریف نشده باشد، عبارت انتسابی عادی محاسبه خواهد شد. (این کار فقط وقتی ممکن است که *varnm* از نوع ورودی یا داخلی باشد). اگر incvarnm تعریف شده باشد، وقتی مدل دوباره راه اندازی می‌شود، مجدداً ارزیابی خواهد شد.

$\text{Inco}(\text{varnm}) = \text{expr}$: تعریف شرایط اولیه متغیر *varnm*، برای متغیرهای ورودی و خروجی وصل نشده می‌باشد. این نوع از دستور $\text{inc}()$ فقط وقتی استفاده می‌شود که متغیر *varnm* نتواند از طریق شرایط اولیه سیگنال خروجی و ورودی متصل، راه اندازی شود. بنابراین دستور $\text{inc}()$ برای باز کردن احتمالی ترمینالهای ورودی و خروجی استفاده می‌شود.

$\text{incfix}(\text{varnm}) = \text{expr}$ این گونه از دستور $\text{inc}()$ فقط وقتی اعتبار دارد که مقداردهی اولیه خودکار باشد معتبر می‌باشد و برای تعیین مقادیر اولیه در موقعیت های دوگانه استفاده می‌شود. با incfix یک یا چند متغیر مستقیماً مقداردهی اولیه شده درحالی که مقادیر اولیه سایر متغیرها می‌توانند بطور خودکار مقداردهی شوند.

مثال :

مدل *AVR* دارای دو ورودی (*upss*, *usetp*) و یک خروجی (*uerrs*) می‌باشد. با مقدار خروجی تنها، نمی‌توان بطور همزمان و خودکار به هر دو ورودی مقدار اولیه نسبت داد. بنابراین مقدار اولیه یکی از ورودیها را باید عدد ثابتی داد، به طور مثال $\text{incfix}(\text{upss})=0$ آنگاه مقدار اولیه *usetp* با استفاده از $\text{upss} = 0$ بطور خودکار تعیین می‌شود.

تنظیم تکرار

به منظور تعیین مقادیر اولیه بطور مکرر، سه تابع وجود دارد که عبارتند از: *newtonnic*, *intervalinc*, *loopinc* این توابع برای پیدا کردن مقدار اولیه یک مجموعه از پارامترها استفاده می‌شود، اگر مقادیر اولیه مجموعه دیگر پارامترها که در واقع توابعی از مجموعه اول هستند، معلوم

باشند. توابع تکرار برای پیدا کردن (تقریبی) مقادیر پارامترهای نا مشخصی که این پارامتر شناخته شده مقادیر اولیه آنها را می‌گیرد، استفاده می‌شوند.

loopinc(varnm, min, max, step, eps) : این تابع یک جستجوی خطی ساده را برای یافتن

مقدار منحصر بفردی انجام می‌دهد که در ازاء آن پارامتر *varnm* نزدیکترین مقدار به مقدار اولیه معلوم خود داشته باشد.

Varnm = متغیر نهایی، که مقدار اولیه معلوم باشد.

حد بالا = **max**

حد پایین = **min**

حداکثر خطا = **eps**

اندازه گام = **step**

مثال : **inc(a) = loopinc(b, -5, 5, 0.01, 0.001)** : یعنی :

مقدار اولیه متغیر "a" با محاسبه پارامتر "b" بدست می‌آید، از **a=-5** شروع شده و در **a=5** پایان می‌یابد و اندازه گام افزایشی ۰/۰۱ می‌باشد.

مقدار برگشتی : مقداری برای متغیر "a" خواهد بود که در ازاء کوچکترین انحراف متغیر "b" از مقدار اولیه اش، بدست آمده است. اگر کمترین انحراف از ماکزیمم خطای *eps* بزرگتر باشد، هشدار داده خواهد شد.

محدودیت : فقط در سمت راست دستور **inc()** می‌تواند استفاده شود.

intervalinc(varnm, min, max, iter, eps) : این تابع یک جستجوی بین فاصله ای (با

تقسیم فاصله به اجزاء ریزتر) را برای یافتن مقدار منحصر بفردی انجام می‌دهد که در ازاء آن پارامتر *varnm* نزدیکترین مقدار به مقدار اولیه معلوم خود داشته باشد.

Varnm = متغیر نهایی، که مقدار اولیه معلوم باشد.

حداکثر تعداد دفعات تکرار = **iter** حد بالا = **max** حد پایین = **min** حداکثر خطا = **eps**

مثال : **inc(a) = intervalinc(b, -5, 5, 40, 0.001)** :

توضیح : از طریق تقسیم متوالی فاصله [۵ و -۵] به فواصل کوچکتر سعی دارد تا مقدار اولیه متغیر "a" را بیابد و تا زمانی که انحراف متغیر "b" با مقدار اولیه اش از *eps* کمتر باشد، عملیات ادامه خواهد یافت. اگر تعداد تکرارها به حداکثر مقدار خود برسد، فرآیند تکرار متوقف خواهد شد و اگر کمترین انحراف بیش از *eps* باشد یک پیغام هشدار داده می‌شود.

محدودیت : فقط در سمت راست دستور **inc()** می‌تواند استفاده شود.

newtoninc(initexpr, start, iter, eps) : با هدف دستیابی به یک یا چند پارامتر، یک

جستجوی تکراری نیوتن را از طریق حداقل کردن خطا در یک مجموعه از زوج معادلات انجام می‌دهد.

initexpr = عبارتی که باید با پارامترهایی که مقدار اولیه آنها جستجو شده است، برابر باشد.

Start = مقدار راه انداز (اولیه) برای پارامتری که مقدار اولیه آن جستجو شده است.

iter = حداکثر تعداد مجاز تکرارها.

eps = حداکثر خطای مطلق مجاز بین *initexpr* و پارامتری که مقدار اولیه آن جستجو شده است.

مثال :

qto = 0.5
 eps = 0.000001
 maxiter = 100

inc (hedr) = newtoninc (hw - sqr (qedr) * (Rds + Rdr),hw,maxiter,eps)
 inc (qt1) = newtoninc (pt1 / (4*dh * eta1) , qto , maxiter , eps)
 inc (qt2) = newtoninc (pt2 / (4*dh * eta2) , qto , maxiter , eps)
 inc (qt3) = newtoninc (pt3 / (4*dh * eta3) , qto , maxiter , eps)
 inc (qt4) = newtoninc (pt4 / (4*dh * eta4) , qto , maxiter , eps)

این مثال قسمتی از تعاریف مقدار اولیه را برای مدلی نشان می‌دهد که در آن مقادیر اولیه ۵ پارامتر ('hedr','qt1',..., 'qt4') با تشکیل سیستمی از زوج معادلات و حل آن با روش تکراری نیوتن، بطور همزمان حل شده است. در نهایت خواهیم داشت:

hedr ≈ hw - sqr (qedr) * (Rds + Rdr)
 qt1 ≈ pt1 / (4*dh * eta1)
 qt2 ≈ pt2 / (4* dh * eta2)
 qt3 ≈ pt3 / (4 * dh * eta 3)
 qt4 ≈ pt4 / (4 * dh * eta4)

۳-۴-۶ کد معادله

در کد معادله، همه معادلات لازم برای ساختن مدل‌های شبیه سازی گنجانده شده است. مجموعه معادلات، دسته ای از زوج معادلات دیفرانسیل را تعریف می‌کند که توصیف کننده توابع تبدیل بین سیگنال‌های ورودی و خروجی است. توابع تبدیل می‌توانند از توابع ساده خطی با یک سیگنال خروجی و یک سیگنال ورودی تا توابع غیرخطی پیچیده ناپیوسته با چند ورودی و خروجی، متغیر باشند. *DSL* برای توصیف ارتباط مستقیم بین سیگنالها و دیگر متغیرها استفاده می‌شود. ممکن است عبارات به خود متغیر یا به مشتق اول متغیر حالت آن اختصاص داده شوند. بنابراین معادلات دیفرانسیل با درجه بالاتر را باید با معرفی متغیرهای حالت دیگر به دسته ای از معادلات دیفرانسیل مرتبه اول تبدیل نمود.

۳-۴-۷ دستور معادله

دستورات معادله برای اختصاص دادن فرمولها به پارامترها استفاده می‌شود، بنابراین ارتباط تمام پارامترها را در یک دسته از معادلات دیفرانسیل نشان می‌دهد.
 گرامر :

$\text{Varnm} = \text{expr}$: فرمول expr را به متغیر varnm اختصاص می‌دهد.

مثالها :

$y = \sin(a) + 3 * x1$
 $y = . \text{not. } x1 > 2. \text{or. } a \leq 3$

$\text{varnm.} = \text{expr}$: فرمول expr را به اولین مشتق متغیر varnm اختصاص می‌دهد.

مثالها :

$x1. = (xe - x1) / T1$
 $x2. = x1$

توضیحات :

- معادل قرار دادن *DSL* در هر توالی ممکن است رخ دهد. این ترتیب و توالی بر محاسبه فرمول ها تأثیری ندارد.
- همه متغیرها از نوع اعشاری هستند، حتی اگر به یک عبارت دودویی تخصیص داده شده باشند، که در آن حالت مقدارش 0.000 یا 1.000 خواهد بود.
- وقتی یک متغیر *z* در یک عبارت منطقی استفاده می شود (مثلاً $y = \text{not. } z$)، آنگاه تعبیر عدد منطقی 1 از *z* با توجه به رابطه $(z > 0.5)$ بوده و به شکل زیر تعبیر می گردد :
 $y1 = \text{not. } z$ معادل است با $(z < 0.5)$.
- به هنگام ترکیب متغیرهای منطقی و ناگسسته در فرمول ها، اختطاری داده نمی شود بنابراین ادغام روابط زیر :

$$y = \text{not. } x1 > 2 \text{ or } a <= 3$$

$$z = 4.0 * y + x1$$

منجر به هیچ پیغامی نخواهد شد : بسته به *z* ، *y* مقدار $x1 + 4/0$ یا فقط *x1* را خواهد گرفت.

- اختصاص دادن مقدار به یک متغیر به گونه ای است که ارتباطات بین متغیرها تشخیص داده خواهد شد. برای مثال زیر :

1. $a = b + 5$
2. $b = x1$
3. $x1 = 1$

ابتدا سطر دوم محاسبه شده و سپس از نتیجه بدست آمده آن، سطر اول محاسبه خواهد گردید.

- تشکیل حلقه از عبارات ریاضی مجاز نمی باشد. در مثال زیر یک پیغام خطا ظاهر خواهد شد :

$$a = b + 5$$

$$b = 2 * a$$

- اگر هیچ مقداری به متغیر *varnm* نسبت داده نشود، مقدار اولیه خود را حفظ خواهد کرد.
- عبارت سمت راست نباید شامل مشتق های عبارات باشد، مشتق ها باید فقط در سمت چپ معادله ظاهر شوند :

$$a = \sin(x1) : \text{غلط و } x1 = \sin(a) : \text{صحیح است.}$$

ماکروهای DSL

یک ماکروی *DSL* مدل از پیش تعریف شده، تابع ابتدایی یا تابع پیچیده *DSL* است، که منظور از ابتدایی این است که باید در مدلهای سطح بالاتر *DSL* گنجانده شود. کادر محاوره ای ویرایش بلوک یک گزینه طبقه بندی "Macro" را ارائه می کند که می تواند در تنظیم علامت گذاری مدل به عنوان ماکرو به کار رود. لحاظ نمودن یک ماکروی *DSL* در یک مدل سطح بالاتر، یا با ایجاد یک مرجع بلوکی در گرافیکهای بلوک دیاگرام امکان پذیر است یا با ارتباط واضح و منطقی در یک معادله *DSL*.
گرامر :

$$\text{varnm1, varnm2, ...} = \text{macroname}(i1, i2, \dots ; s1, s2, \dots ; p1, p2, \dots ; i1, i2, \dots)$$

ماکروی *DSL* با نام *macroname* سیگنال های خروجی را به متغیرهای *varnm1*، *varnm2* و... و سیگنال های ورودی را به متغیرهای *i1* و *i2* اختصاص داده است. این ماکرو از متغیرهای حالت 's2' و 'p1'، پارامترهای 'p1' و 'p2'، و متغیرهای ورودی *i1* و *i2* نیز استفاده نموده است.

مثال: $P1, P2 = \backslash \text{User} \backslash \text{I.BlkDef}(i1, i2; s1, s2; T1, T2)$

در این مثال به *P1* و *P2* خروجی مدل 'User\I.BlkDef' از *DSL* تخصیص داده شده است. امکان فراخوانی های ماکرو در داخل فرمولها وجود ندارد، حتی اگر آنها فقط دارای یک متغیر خروجی باشند:

$y = \text{my_macro}(x1, s1, p1, i1)$: مثالی صحیح می باشد،

اما

$y = 3 * \text{my_macro}(x1, s1, p1, i1) + 4$ نادرست است که باید به صورت زیر جایگزین

شود:

$y1 = \text{my_macro}(x1, s1, p1, i1)$

$y = 3 * y1 + 4$

کارکردن با ماکروی داخلی *DSL*

یک پیش تحلیلگر وظیفه دارد تا هر فراخوانی ماکرو را، با کد معادله ماکرو جانشین کند. متغیرهای مدل *DSL* ماکرو بوسیله متغیرهای استفاده شده در فراخوانی ماکرو، جایگزین شده اند. نامهای بومی متغیر ماکروها بعد از مرحله پیش تحلیلی، از بین خواهند رفت.

مدلهای *DSL*

به طور کلی، دو نوع مدل اصلی *DSL* وجود دارد که عبارتند از:

۱- مدل‌های دستگاه‌های الکتریکی مانند ژنراتورها، بارها یا سیستم های *HVDC*. این مدلها بوسیله سیگنال خروجی اصلی خودشان 'complex device current' که در یک باس بار معین به شبکه الکتریکی تزریق شده است، مشخص شده اند. به هر حال، علاوه بر جریان های ادوات الکتریکی ممکن است هر متغیر دیگر را نیز به عنوان سیگنال خروجی تعریف نمود. توجه: این ادوات سیستم قدرت که توسط کاربر تعریف شده اند، زمانی که این کتاب راهنما نوشته می شده، در نرم افزار وجود نداشته است.

۲- مدل‌هایی با سیگنال‌های خروجی که مستقیماً به شبکه الکتریکی (دستگاه‌های کلی) تزریق نشده اند. از جمله این نوع مدلها واحدهای محرک اولیه، کنترل کننده های ولتاژ، رله ها، روشهای محاسبه و غیره را می توان نام برد.

حوادث و پیغامها

زبان *DSL* روش هایی را برای ایجاد یک رخداد وقفه یا برای ارسال یک پیغام خطا به پنجره خروجی فراهم نموده است که عبارتند از:

- روش $\text{Fault}(\text{boolexpr}, \text{event_string})$ که یک رخداد را ایجاد نموده و در آغاز هر پله زمانی، آن را محاسبه می کند.

- روش `output(boolexpr , message_string)` یک پیغام خروجی ایجاد می‌کند و در پایان هر پله زمانی محاسبه می‌شود.

روشهای `output` و `fault` در هر پله زمانی از فرایند شبیه‌سازی مدل، محاسبه می‌شوند. مرتبه اولی که مشخص شود `boolexpr` صحیح است، این رشته پردازش خواهد شد و پیغامی به پنجره خروجی فرستاده می‌شود یا رخدادی به صف حادثه `DIGSILENT` کشیده خواهد شد. بعد از این، روشهای `output` و `fault` به منظور جلوگیری از ارسال سیل آسای پیغامها و حوادث تا زمانی که مدل 'DSL' دوباره راه اندازی شود، غیرفعال خواهند شد. هر دو روش به تفصیل در بخشهای بعدی تشریح شده اند.

Output(boolexpr , message_string)

`message_string` ممکن است شامل متغیرها و تابع خاص `num(boolexpr)` یا `num(expr)` باشد :

- اسامی متغیرهایی که بلافاصله بعد از علامت '=' ظاهر می‌شوند بوسیله مقادیر واقعی خودشان جانشین خواهند شد. برای مثال :

`output(yymax : 'maximum excooded : y = yt > ymax = ymax')`

که ممکن است منجر به تولید پیغام " از مقدار حداکثر تجاوز کرده است : `yt=1.2 > ymny=10` گردد.

- `num(expr)` یا `num(bool expr)` با مقدار محاسبه شده فرمول، جایگزین خواهند شد. مثال :

`value = num (a + b)` که ممکن است مقدار `value = 3.5000` را ایجاد نماید.

Fault(boolexpr , event_string)

هر مدل `DSL` دارای قابلیت انتقال حوادث به محلی جهت ذخیره سازی می‌باشد. به طور مثال در مدل `DSL` یک رله دیستانس، می‌تواند کلید قدرت یک خط را از طریق تعریف صحیح رخداد کلید باز نمود. "Throwing an event" (ایجاد یک رخداد) با اجرای یک رخداد موجود در پایگاه داده‌های `DIGSILENT` اجرا می‌شود.

در نتیجه، همه رخدادهایی که ممکن است بوسیله مدل `DSL` استفاده شده باشند باید بطورهمزمان با مدل `DSL` ایجاد گردند. همه آنها باید در داخل مدل مشترک `ElmDSL`، دکمه `event` ذخیره شوند. بنابراین `DSL events` یک قسمت پیوسته از مدل `DSL` را تشکیل خواهند داد.

`event_string` در فرمول خطا بایستی به نام یکی از این رخدادها ارجاع داده شود. در محاسبه، اگر `boolexpr` درست باشد، آنگاه رخداد به درون پشته حادثه کشیده می‌شود. به محض این که فرایند شبیه سازی به این رخداد رسید، آن را اجرا خواهد نمود. در نتیجه یک رخداد تأخیری را ممکن است با تنظیم زمان اجرا جلوتر از زمان جاری و بوسیله مدل `DSL` اجرا نمود.

پارامترهای رخداد در این رشته خطا را می‌توان با اختصاص دادن یک مقدار جدید اصلاح کرد. روش کار همان است که در فوق و در مورد روش خروجی توصیف شد.

مثال :

Fault (u > 1.1 , 'name = MyswitchEvent1 dtime = 0.15 ')

اگر متغیر u از 1.1 بیشتر گردد، آنگاه رخداد نامیده شده با MySwitchEvent1 به درون پشته رخداد کشیده خواهد شد و متغیر $dtime$ (زمان نسبی رخداد) در ۱۵ میلی ثانیه تنظیم خواهد گردید. بنابراین، این رخداد به آن مقدار زمانی به تأخیر خواهد افتاد، که در این مورد زمان لازم برای باز کردن یک سویچ را شبیه سازی می کند. و سویچی که باز می شود در رخداد "MySwitchEvent1" تعریف شده است.

۳-۴-۸ مثالها

توربین حرارتی دوباره گرم شده با بخار ذخیره شده

مدل کنترل کننده :

```

model pt , ptmw
'pmu - 1 ' ( at , sgn , cosn , ngnum ; x1 , x2 , x3 , x4 ;Thp , Tip , Tlp , alfhp , Tspi )
    [T1] = 's'
    limits (T1) = [0 , )
    limits (alfhp) = [0 , 1 ]
    vardef (alfhp) = ; 'High pressure turbine ratio ' ;
    limits (alfhp) = [0 , 1 - alfhp ]
    vardef (alfhp) = ; ' Low pressure turbine ratio ' ;
    vardef (Tspi) = 's' ; 'Boiler capacity time constant ' ;
    limits (Tspi) = ( 0 , )
    vardef (Thp) = 's' ; 'High pressure turbine time constant ' ;
    vardef (Tip) = 's' ; ' First reheater time constant ' ;
    vardef (Tlp) = 's ' ; ' second reheater time constant ' ;

    inc (x1) = y / k
    inc (xe) = y / k
    inc (x4) = 1.0
    inc (at) = pt
    inc (steamflow0) = pt
    inc (ylp) = pt
    x1. = select ( T1 > 0 , ( xe - x1 ) / T1. 0 )
    y = k * select ( T1 > 0 , x1 , xe ) ! if T1 = 0 => y = xe
    steamflow = at * x4
    x4. = ( steamflow0 - steamflow ) / Tspi ! boiler
    yhp = PT1 (steamflow ; x1 ; Thp) ! high pressure part
    yip = PT1 ( yhp ; x2 ; Tip ) ! medium pressure part
    ylp = PT1 ( yip ; x3 ; Tlp ) ! low pressure part
    pt = yhp * alfhp + ylp * alfhp + yip * ( 1. 0 - alfhp - alfhp )
    ptmw = pt * sgn * cosn * ngnum ! only for output purposes

```

ماکروی استفاده شده 'PT1' به صورت زیر تعریف می شود:

```
model y = 'PT1' ( xe ; x1 ; k , T1 ; )
  x1. = Select ( T1 > 0 , ( xe - x1 ) / T1 , 0)
  y = k* select ( T1 > 0 , x1 , xe )      ! if T1 = 0 => y = xe
  inc (x1) = y/k
  inc (xe) = y / k
  [T1] = 's'
  limits (T1) = [ 0 , )
```

3.5 DSL Funtions

۱-۵-۳ توابع استاندارد

sin(x)	sine	sin(1.2)=0.93203
cos(x)	cosine	cos(1.2)=0.36236
tan(x)	tangent	tan(1.2)=2.57215
asin(x)	arcsine	asin(0.93203)=1.2
acos(x)	arccosine	acos(0.36236)=1.2
atan(x)	arctangent	atan(2.57215)=1.2
sinh(x)	hyperbolic sine	sinh(1.5708)=2.3013
cosh(x)	hyperbolic cosine	cosh(1.5708)=2.5092
tanh(x)	hyperbolic tangent	tanh(0.7616)=1.0000
exp(x)	exponential value	exp(1.0)=2.718281
ln(x)	natural logarithm	ln(2.718281)=1.0
log(x)	log10	log(100)=2
sqrt(x)	square root	sqrt(9.5)=3.0822
sqr(x)	power of 2	sqr(3.0822)=9.5
pow (x,y)	power of y	pow(2.5, 3.4)=22.5422
abs(x)	absolute value	abs(-2.34)=2.34
min(x,y)	smaller value	min(6.4, 1.5)=1.5
max(x,y)	larger value	max(6.4, 1.5)=6.4
modulo(x,y)	remainder of x/y	modulo(15.6, 3.4)=0.58823
trunc(x)	integral part	trunc(-4.58823)=-4.0000
frac(x)	fractional part	frac(-4.58823)=-0.58823
round(x)	closest integer	round(1.65)=2.000
ceil(x)	smallest larger integer	ceil(1.15)=2.000
floor(x)	largest smaller integer	floor(1.78) = 1.000
pi()	3.141592...	p=3.141592...
twopi()	6.283185...	2p=6.283185...
e()	2,718281...	e=2,718281...

آرگومان همه توابع مثلثاتی رادیان می باشد.

۲-۵-۳ توابع خاص

$Lim(u, min, max)$ = تابع محدود کننده غیر خطی.

returns $\left\{ \begin{array}{l} \min \text{ if } x < \min \\ \max \text{ if } x > \max \\ x \text{ if } \min \leq x \leq \max \end{array} \right\}$

$Limstate(x1, min, max)$ = تابع محدود کننده غیر خطی برای ایجاد انتگرال گیرهای محدود

شده.

مثال :

$x1. = xe/Ti ;$

$y = limstate(x1, min, max) ;$

این مورد قبلاً بوسیله استفاده از توابع "Selecte" و "lim" بررسی شده است.

$x1. \& = \& select(\{ x1\$>\$ = max. \text{ and. } xe\$ > \$0 \}$

$\& \text{ . or. } \{ x1 \$ < \$ = min. \text{ and. } xe\$ < \$0 \} , 0 , xe / Ti) ;$

$y \ \& = \& lim(x1, min, max) ;$

$delay(x, Tdelay)$: تابع تأخیری. مقدار $x(Tnow)$ را ذخیره کرده و مقدار

$x(Tnow, Tdelay)$ را بر می گرداند. $Tdelay$ برحسب ثانیه بوده و از 0.0 بزرگتر می باشد.

$Tdelay$ باید به صورت یک ضریب مستقل از زمان محاسبه شود و بنابراین باید فقط شامل ضرایب ثابت و متغیرهای پارامتری باشد. عبارت $x(t)$ ممکن است شامل توابع دیگری باشد.

مثال : $y = delay(xe + delay(x1, 1.0), 2.0)$

تنظیم مجدد یک مدل DSL توابع تأخیری آن را با $x(Treset)$ مقداردهی اولیه می نماید.

$selecte(boolepr, x, y)$: اگر $boolepr$ درست باشد x را بر می گرداند و در غیر این صورت

y برگردانده می شود.

مثال : برای جلوگیری از تقسیم شدن بر صفر

$x1. = select(T1 > 0, xe / T1, o/o)$

$Lapprox(x, array_iiii)$: تقریب خطی $y = f(x)$ را در جایی که f بوسیله $array_iiii$

تعریف شده است، بر می گرداند.

مثال :

$y = lapprox(1.8, array - valve)$

$invlapprox(y, array - iiiii)$: تابع $Lapprox$ را با $array_iiiiii$ معکوس می کند.

$file(ascil-parm, expr)!OBSOLETE$: از یک عنصر $ElmFile$ به جای مدل مرکب

استفاده می کند.

$Picdro(boolepr, Tpick, Tdrop)$: تابع منطقی 'pick - up - drop - off' برای رله ها

مفید می باشد. این تابع حالت منطقی داخلی را بر می گرداند یعنی : صفر یا یک. حالت داخلی عبارت است

از :

- اگر برای مدت زمانی معادل حداقل $Tpick$ ثانیه، $boolexpr = 1$ باشد، از صفر به یک تغییر پیدا می کند.
- اگر پس از $Tdrop$ ثانیه، $bollexpr = 0$ باشد، از یک به صفر تغییر پیدا می کند.
- در وضعیت های دیگر بدون تغییر باقی می ماند.

flip flop(boolset , boolreset): تابع فلیپ فلاپ منطقی. این تابع حالت منطقی داخلی را

بر می گرداند یعنی: صفر یا یک. حالت داخلی عبارت است از:

- اگر $boolset = 1$ و $boolreset = 0$ باشد. از صفر به یک تغییر می یابد (SET).
- اگر $boolset = 0$ و $boolreset = 1$ باشد. از یک به صفر تغییر می یابد (RESET).
- در شرایط دیگر بدون تغییر باقی می ماند (HOLD).

Boolset: مقدار اولیه

شرایط اولیه $boolset = boolreset = 1$ باعث ایجاد پیغام خطا می شود.

aflipflop(x , boolset , boolreset): یک تابع فلیپ فلاپ پیوسته. اگر وضعیت داخلی مساوی

۱ باشد مقدار (قبلی) را برای x در SET-time برگردانده و در غیر این صورت مقدار کنونی x را بر می گرداند.

این وضعیت داخلی عبارت است از:

- اگر $boolset = 1$ و $boolreset = 0$ باشد. از صفر به یک تغییر می یابد (SET).
- اگر $boolset = 0$ و $boolreset = 1$ باشد. از یک به صفر تغییر می یابد (RESET).
- در موقعیت های دیگر بدون تغییر باقی می ماند (HOLD).

Time(): زمان شبیه سازی جاری را بر می گرداند.

مثال:

$t = \text{time} () , y = \sin (t) \text{ or } y = \sin (\text{time} ())$