Tagged-Token Dataflow Architecture Simulation Facility User's Manual

Version 1.0

CSG Memo 250

March 1, 1985

Stephen A. Brobst

This research was supported in part by Advanced Research Projects Agency of the Department of Defense contract N00014-75-C-0661, in part by IBM Yorktown.

Table of Contents

List of Figures

1. Introduction

The Tagged Token Dataflow Project underway at the Massachusetts Institute of Technology is an effort to realize the model of dataflow computation embodied in the U-Interpreter. The Tagged Token Dataflow Architecture (TTDA) is a radical divergence of from traditional von Neumann machines. As a result, current benchmarks of computation derived from the study of sequential processors are not applicable for making implementation decisions in the design of our machine. Thus, to facilitate construction of the TTDA, a detailed simulation of the machine architecture has been developed.

The primary purpose of the simulation facility is to provide a prototype and test bed for the Tagged Token Dataflow Architecture. All components in the Tagged Token Dataflow Architecture are modeled in such a way that there exists a direct translation between functional modules in the simulation program and hardware components in the machine architecture. The architecture of a processing element in the TTDA is essentially an asychronous pipeline with stages connected via finite buffers. This modularity is reflected in the design of the simulator. The functional behavior of each stage is modeled by a Pascal procedure which receives an input packet and the state of a station and produces a list of result packets with a new state.

Machine configuration, hardware components, and resource management policies are all parameters of the Tagged Token Dataflow Architecture that can be determined by the user of the simulation facility. By experimenting with different parameterizations of the architecture, on can study design tradeoffs and their relative affects upon the performance of the machine. In addition to providing detailed modeling of the TTDA, the simulation facility has tools for debugging programs in a multiprocessor environment and for analyzing performance data generated by a simulation experiment.

The TTDA Simulation Facility is approximately 50,000 lines of Pascal code. It runs on an IBM

4341 in either batch or interactive mode, depending on user preference. If you do not have an account on MIT-BIGBLUE and would like to use the simulation facility, contact Bart Blaner (blaner@xx) to obtain your very own virtual machine. If you have an account on our IBM 4341, but do not have the TTDA EXEC installed, please send a message to me (sabrobst@xx) and I will set you up.

2. Porting Numeric Machine Code to the IBM 4341

The first step necessary to run an ID program on the TTDA Simulator is to compile the program into numeric machine code. The TTDA Simulator takes the numeric machine code generated by the ID Compiler as one of its inputs for a simulation run. If you have not yet compiled your ID program into numeric machine code please refer to the *ID Compiler User's Manual* for assistance in achieving this task.

Once you have compiled your ID program, it is necessary to port the numeric machine code to the IBM 4341 for use by the simulator. The following sequence of commands will achieve this task. In the example given, I assume you are porting from XX. However, you will need to specify a different machine and directory syntax if your numeric machine code exists on a Lisp machine or some other host. After logging on to the IBM 4341, you will want to invoke the FTP program to facilitate the file transfer of your numeric machine code. Text in **bold** represents prompts or messages from the IBM 4341 and text in *italics* are user responses.

```
ftp

Wisconsin WM Ftp version 1.3
Command:

connect xx

Connecting to xx 10.0.0.44, port 21
220 MIT-XX.ARPA FTP Server Process ...
Command:

login <login-name> <password>

>>>USER <login-name>
331 User name ok.  Password, please.
>>>PASS ********
230 User <login-name> logged in at Sat 21-Sep-85 ...
Command:

get <directory-path>progname.nmc progname.nmc.a

>>>RETR <directory-path>progname.nmc
150 ASCII retrieve of <directory-path>progname.nmc.1 started
226 Transfer completed.
5013 bytes transfered in 5.527 seconds ...
```

```
Command:

quit

>>>QUIT
221 QUIT command received.  Goodbye.
R; T=0.19/0.47 11:31:38
```

Now that the numeric machine code has been transfered to the A disk of your IBM Virtual Machine, you will want to send this file to the simulation server. The simulation server resides on a virtual machine called *simbatch*. Thus, you will want to send the numeric machine code to the simulation server as follows:

```
sendfile progname nmc a simbatch

File progname nmc a sent to SIMBATCH at BIG-BLUE ...
R; T=0.29/0.41 11:35:38
```

After you have ported the machine code for each of your ID programs and sent them to the simulation server, you will be ready to construct simulation experiments. The next two chapters will assist you in this task.

3. Using the Simulation Facility Interface

The user interface for constructing simulation experiments is built into the ISPF menu structure. To invoke the ISPF menu driver which includes definitions for the TTDA simulation, type *TTDA*. The result of this command will be a display of the following menu.

```
OPTION ===>
                                                             USERID    - SABROBST
   0 ISPF PARMS  - Specify terminal and user parameters    TIME      - 12:10
   1 BROWSE      - Display source data or output listings  TERMINAL  - 3277
   2 EDIT        - Create or change source data            PF KEYS   - 12
   3 UTILITIES   - Perform utility functions
   4 FOREGROUND  - Invoke language processors in foreground
   5 BATCH       - Submit batch job
   6 COMMAND     - Enter CMS command or EXEC
   7 DIALOG TEST - Perform dialog testing
   C CHANGES     - Display summary of changes for this release
   S SIMULATOR   - Display TTDA simulator entry panel
   T TUTORIAL    - Display information about ISPF/PDF
   X EXIT        - Terminate using console, log, and list defaults


Enter END command to terminate ISPF.
```

Figure 3-1: Top-level ISPF Menu

Since you are interested in simulation experiments, press S followed by the ENTER key to select the *simulator* menu choice. You have now entered the world of the Tagged Token Dataflow Architecture Simulation facility. The top-level menu of the simulation facility will be displayed.

```
OPTION ===>
                                                             USERID    - SABROBST
  D   DEFINE     - Define an experiment                        TIME      - 12:11
  E   EXTEND     - Extend an experiment
  B   BATCH RUN  - Run an experiment in batch mode
  U   USER RUN   - Run an experiment in user's virtual machine (interactive)
  M   MONITOR    - Monitor the progress of a simulation experiment
  V   VIEW       - View performance data from a simulation experiment


  PF1 = Help                                                 PF3 = Exit
```

Figure 3-2: Top-level Simulation Facility Menu

From this menu you can choose the desired function by typing the letter to the left of the menu item, followed by pressing the ENTER key. The DEFINE function allows you to define a new experiment, EXTEND provides a means for extending an existing experiment, BATCH RUN and USER RUN will allow you to run an experiments in batch or interactive mode respectively, and VIEW provides tools for examining and analyzing performance data from a simulation experiment.

3.1. Terminology

It is useful at this point to establish a common ground by associating specific terminology with concepts supported within the TTDA Simulation Facility.

1. **Code-block Name:** A code-block name, in the context that we will be interested in for running simulation experiments, is the name of an ID procedure[1] definition. For example, I might write a procedure to calculate the factorial of its input argument. If I named this procedure *factorial*, then its code-block name would also be factorial. Of course, there may be many code-block definitions within a single ID source file.

2. **Numeric Machine Code File:** This is the file which contains the numeric machine code generated by the ID compiler from an ID source code file. An executable program may consist of one or many numeric machine code files.

3. **Simulation Run:** A simulation run consists of a single parameterization of the architecture, data collection specification, and code-block specification. For example, a simulation run might consist of a machine configuration with 16 processing elements, round-robin resource management policies, default hardware timings, no data collection, and an invocation of a code-block named *matmul* with arguments (10, 10, 10).

4. **Simulation Experiment:** A simulation experiment is a collection of one or more simulation runs which are grouped together in some way which is meaningful to the user. For example, if one wanted to experiment with the scalability of the architecture, a reasonable grouping of simulation runs might be to have 32 simulation runs all of which are identical except that they range in the number of processing elements from 1 to 32.

5. **Simulation Job:** A simulation job is a run which has been submitted to the batch simulation server. Runs are submitted in groupings known as experiments. The simulation server will execute each of the runs in an experiment in the order that they appear in its queue. There may be many runs from different experiments and users in the simulation server's job queue at one time.

[1] Loops are also implemented as code-blocks, but this is not of interest to us here.

3.2. Command Keys

There are a number of keys which have consistent meanings throughout the simulation facility interface. This section enumerates these keys and their associated meanings.

1. **ENTER:** The ENTER key is used as a general affirmation key in the simulation interface. After you have typed in a command or specifications, pressing the ENTER key will indicate that the data entered is correct and should be processed.

2. **TAB:** There are two tab keys on the IBM 4341 keyboard. These are used to move from selection to selection on the various specification and selection menus. The *forward tab* takes you to the next field and the *backward tab* takes you to the previous field. Tab keys have wrap around capability.

3. **PF1:** The PF1 key will bring up a help screen pertaining to the function in which it was pressed. The help screen will elaborate on the menu choices or functions from which it was invoked.

4. **PF3:** The PF3 key will exit from the current menu or display to the previous menu in the simulation facility hierarchy.

5. **PF7:** The PF7 key will scroll backwards when there exists more than one screen for a display or specification entity.

6. **PF8:** The PF8 key will scroll forward when there exists more than one screen for a display or specification entity.

4. Constructing a Simulation Experiment

As indicated previously in section 3.1 of this manual, a simulation experiment is a collection of one or more simulation runs which are grouped together in some way which is meaningful to the user. There are two ways to construct a simulation experiment. The first way is to define a new experiment and specify a collection of runs within this experiment definition. Alternatively, one can choose to extend an already existing experiment. These two alternatives correspond to the DEFINE and EXTEND options, respectively, on the top-level menu of the simulation facility. To *define* a new experiment, type D followed by the **ENTER** key. To *extend* an existing experiment, type E followed by the **ENTER** key.

4.1. Defining an Experiment

The first thing you will be asked to to when defining a new experiment is to assign a name to the new experiment. This allows you to treat all runs within an experiment as a single entity referenced by the experiment name you choose. The following screen will be displayed.

```
                    Tagged Token Dataflow Architecture
                           Simulation Facility

COMMAND ===>

   Please enter a name for this experiment: (default is Dataflow)
     ===> NAME


   The user interface part of the simulator is still under development.
   If you find certain aspects of this interface awkward or confusing,
   send complaints to sabrobst@xx.


  PF1 = Help                                                PF3 = Exit
```

Figure 4-1: Naming an Experiment

At this point, you should use the forward tab key to place the cursor at the experiment name prompt. Now type in the name of your experiment. If an experiment with that name already exists, you will receive an error message and should try a different name. The **PF3** key will allow you to exit from the Define an Experiment function without creating a new experiment. The **PF3** key will display a help screen pertaining to the Define an Experiment function.

After you have given a name to the new experiment, the following menu will be presented. This menu allows you to ADD, DELETE, or MODIFY program runs within the new experiment. Since the experiment is initially empty, you will want to ADD a run to the new experiment.

```
                         Tagged Token Dataflow Architecture
                              Simulation Facility

 OPTION ===>

   A   ADD       - Add a run to experiment NAME
   D   DELETE    - Delete a run from experiment NAME
   M   MODIFY    - Modify a run in experiment NAME


   PF1 = Help                                            PF3 = Exit
```

Figure 4-2: Add, Delete, Modify Menu

To select a given function, merely press the letter (*e.g.*, A for ADD) to the left of the choice followed by the **ENTER** key. The **PF3** key will exit from the DEFINE function and **PF1** key will provide a help screen regarding the menu choices available at this level.

4.1.1. Add a Run to an Experiment

To add a run to an experiment, specifications for buffer sizes between stages in the pipeline of a processing element, the system and processing element configuration, timing parameters, the code-block to be invoked, and data collection. The following screen will be presented.

```
                    Tagged Token Dataflow Architecture
                           Simulation Facility
                     Adding Run to Experiment NAME

Buffer size specification ......................... STANDARD

System and PE configuration specification ......... STANDARD

Timing parameter specification .................... STANDARD

Code-block specification .......................... EXAMPLE

Data collection specification ..................... STANDARD


To create a new specification type, in the appropriate field, a
name for the specification and press the ENTER key.

  PF1 = Help
  PF3 = Exit without adding run
    ? = Display predefined specifications for current field.
ENTER = Affirm specifications for the new run.  Define new specifications.
```

Figure 4-3: Add a Run to an Experiment

By default, the specifications that you used in your last creation of a simulation run will be entered into the specification fields. There exist standard specifications for all aspects of a simulation run except code-blocks. The standard specifications may be used by entering the name STANDARD in the appropriate fields as indicated in the example above. To examine the set of predefined specifications available for a particular field, press the ? key followed by an ENTER in the desired field. Section 4.1.1.6. gives more information on this option. For general help, pressing the PF1 key will display a help screen with information relevant to the Add a Run to an Experiment function.

The ENTER key is used to affirm the specification data that has been entered for the new run. After this key has been pressed, the simulation facility will verify that all specification names are defined and then create a new run and add it to the experiment. If all specification names have not yet been defined, you will be placed in the entry screen for the first undefined specification. After typing in the desired specifications and pressing the ENTER key, you will be brought back out to the screen shown above. This process will continue until the experiment run is fully specified. To exit from the Add a Run to an Experiment function without creating a new run, press the **PF3 key.**

4.1.1.1. Buffer Size Specification

This specification allows you to define the size of buffers between each of the different stages of a processing element in the Tagged Token Dataflow Architecture. The figure on the next page illustrates the simulation analogue of a processing element in the TTDA. Refer to this figure to determine the meaning of the various buffers between stages in the pipeline of a processing element. The standard default for buffer size specification is given below. To obtain these defaults, merely type **STANDARD** in the buffer size specification field when adding a run to an experiment.

```
                    Tagged Token Dataflow Architecture
                           Simulation Facility
             Buffer Specification STANDARD for Experiment NAME

    Size of buffers between CEstations and INstations in bytes ...... 100
    Size of buffers between INstations and Istations in bytes ....... 1024
    Size of buffers between INstations and WMstations in bytes ...... 100
    Size of buffers between INstations and IFstations in bytes ...... 1024
    Size of buffers between INstations and MANstations in bytes ..... 1024
    Size of buffers between INstations and PeConstations in bytes ... 1024
    Size of buffers between WMstations and IFstations in bytes ...... 1024
    Size of buffers between Istations and OUTstations in bytes ...... 100
    Size of buffers between MANstations and OUTstations in bytes .... 100
    Size of buffers between PeConstations and OUTstations in bytes .. 100
    Size of buffers between IFstations and ALUstations in bytes ..... 100
    Size of buffers between IFstations and CTAGstations in bytes .... 100
    Size of buffers between ALUstations and FTstations in bytes ..... 100
    Size of buffers between CTAGstations and FTstations in bytes .... 100
    Size of buffers between FTstations and OUTstations in bytes ..... 100
    Size of buffers between OUTstations and INstations in bytes ..... 100
    Size of port0 buffers in bytes .................................. 100
    Size of port1 buffers in bytes .................................. 100

    PF1 = Help                 ENTER = Affirm                PF3 = Exit
```

Figure 4-4: Standard Buffer Size Specification

It is extremely important to note that these buffer size specifications are in terms of bytes, **not** packets. A single packet may take up to 50 bytes in a buffer, although the average is about half this number. Notice that there are five buffers which default to 1024 bytes rather than 100. This essentially makes the buffers between these stations infinite. These buffers of "infinite" size

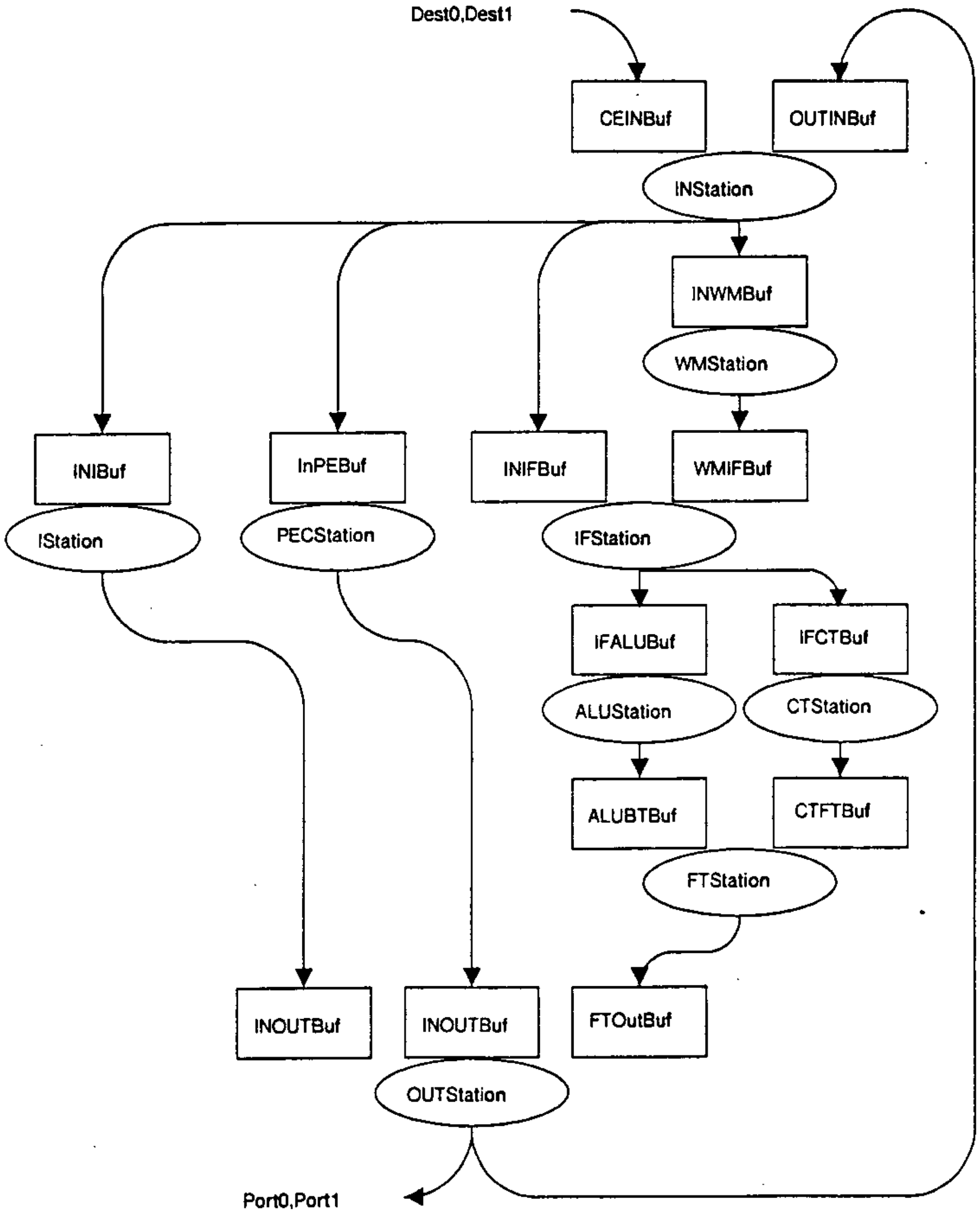


Figure 4-5: Simulation Analogue of a Processing Element

prevent buffer deadlock from occurring in the Tagged Token Dataflow Architecture. There is one infinite buffer for each path in the pipeline of a processing element.

4.1.1.2. System and PE Configuration Specification

Resource management policies and the configuration of processing elements within the Tagged Token Dataflow Architecture must be specified for each run of an experiment. This field allows you to make those specifications. You may either make your own specifications or use those that have already been defined. The standard default specifications are illustrated below. The first screen allows you to specify resource management policies as well as the number of processing elements and domains.

```
                    Tagged Token Dataflow Architecture
                           Simulation Facility
        System and PE Specification STANDARD for Experiment NAME

           Domain Policy ............. RR
           I-Structure Policy ........ RR
           Color Policy .............. RR

           Total Number of PEs ....... 16
           Total Number of Domains ... 8


    PF1 = Help                                               ENTER = Affirm
    TAB = Next Field                                         PF3 = Exit
```

Figure 4-6: Standard System and PE Configuration Specification

The resource management policies currently available are *round-robin* (abbreviated as RR), *random*, and *interactive*. The default policies are all round-robin. However, you may select either of the other two by typing **RANDOM** or *INTERACTIVE* as desired in the appropriate fields. Note that interactive management policies are meaningful only when running a TTDA simulation experiment in user interactive mode.

The number of processing elements and domains in the TTDA configuration must also be specified. The default number of processing elements is sixteen. However, a TTDA simulation run may be specified with anywhere between one and thirty-two processing elements. The default configuration of these sixteen processing elements is eight domains of two PEs each. The second screen of the default System and PE Specification will appear as follows.

```
                  Tagged Token Dataflow Architecture
                         Simulation Facility
          System and PE Specification STANDARD for Experiment NAME

                  Domain Number           Domain Size

                        1                      2
                        2                      2
                        3                      2
                        4                      2
                        5                      2
                        6                      2
                        7                      2
                        8                      2

   PF1 = Help             PF7 = Previous Screen        ENTER = Affirm
   TAB = Next Field       PF8 = Next Screen            PF3 = Exit
```

Figure 4-7: Standard System and PE Configuration Specification

When you enter this screen during the creation of a new System and PE Specification, the user interface will enter default domain sizes based on the last domain sizes you used. These defaults will often be incorrect for your PE specification and should be changed as desired. Domain sizes in the Tagged Token Dataflow Architecture must be a power of two. Furthermore, when entering these domain sizes you should be sure that the sum of the sizes does not exceed the number of processing elements specified on the previous screen. Pressing the ENTER key after entering these domain sizes will complete the System and PE Specification. Press the **PF3** key to exit without creating a new System and PE Specification.

4.1.1.3. Timing Specification

The timing of the hardware components in the Tagged Token Dataflow Architecture are selected using the following Timing Specification data entry screen. The basic unit of time in these specifications is the transit time τ of the FIFO buffers. With current technology, this should be approximately 50 nanoseconds. A full description of these timings and the exact calculations used to derive timing information in the TTDA simulator can be found in a paper entitled *Temporal*

Behavior in the TTDA Simulation, available as a TTDA Simulation Design Note.

```
                    Tagged Token Dataflow Architecture
                           Simulation Facility
             Timing Specification STANDARD for Experiment NAME

Minimum buffer time constant ........... 1   Form token time constant .... 1
Program memory access time constant ..... 3   InStation time constant ..... 1
Program memory latency time constant .... 1
Waiting matching access time constant ... 3   Hash time constant ......... 1
Waiting matching latency constant ....... 1   Match time constant ........ 2
PE controller time constant ............. 50
Integer Add time constant ............... 3
Integer Multiply time constant .......... 15
Floating Point Add time constant ........ 10
Floating Point Multiply time constant ... 20
I-Structure Operation time constant ..... 6
I-Structure access time constant ........ 3
I-Structure latency time constant ....... 1
I-Structure overhead time constant ...... 1
OutStation look-up time constant ........ 2
OutStation overhead time constant ....... 1
Arbiter time constant ................... 2

  PF1 = Help              ENTER = Affirm               PF3 = Exit
```

Figure 4-8: Standard Timing Specification

Varying these timing parameters allows one to experiment with different hardware realizations of the Tagged Token Dataflow Architecture. Sensitivity of the architectural performance to the implementation of a particular component can be easily assessed using different timing models for the component.

4.1.1.4. Code-block Specification

The Code-block Specification for a simulation run requires that you enter (1) the name of the numeric machine code file(s) where the compiled code for your program can be found, (2) the name of the ID code-block to be invoked, and (3) the arguments to that code-block. An example is given below.

```
                    Tagged Token Dataflow Architecture
                            Simulation Facility
            Code-block Specification EXAMPLE for Experiment NAME

      Name of the numeric machine code files where program can be found:

        ===> FACT   NMC
        ===>

      Name of the ID code-block and its arguments:

        ===> FACTORIAL   ===> 10     ===>          ===>          ===>


      PF1 = Help                  ENTER = Affirm                 PF3 = Exit
```

Figure 4-9: Example Code-block Specification

Press the **ENTER** key when you have finished entering the specifications to create a new Code-block Specification. The **PF3** key will exit without creating a new specification. The **TAB** keys will move you from field to field. **PF1** provides a basic help facility.

4.1.1.5. Data Collection Specification

The simulator maintains a complete profile of the Tagged Token Dataflow Machine during the execution of a program run. The Data Collection Specification for a program run indicates which performance data should be saved in the simulation facility database for future reference and analysis. A certain amount of very high level data about a program run such as station utilization, buffer fill levels, waiting-matching token storage fill levels, and so on. However, much more detailed information is also available. The dynamic behavior of any station or buffer in the architecture can be recorded as a series of "snapshots". The granularity of this information is determined by the sampling period chosen for data collection. There are two Data Collection Specification screens to be filled in. The screens shown below illustrate the *standard* Data Collection Specification.

```
                 Tagged Token Dataflow Architecture
                        Simulation Facility
     Data Collection Specification STANDARD for Experiment NAME


Simulator version ................ 1             (default is 1)
Domain version ................... 1             (default is 1)
Color version .................... 1             (default is 1)
I-Structure version .............. 1             (default is 1)
Sampling period in nanoseconds .... 10000        (default is 10000)

Collect statistics globally? ........................... YES
Collect statistics in IN stations? ..................... NO
Collect statistics in Waiting-Matching stations? ........ YES
Collect statistics in ALU stations? .................... YES

PF1 = Help              PF7 = Previous Screen          ENTER = Affirm
TAB = Next Field        PF8 = Next Screen              PF3   = Exit
```

Figure 4-10: Standard Data Collection Specification

The version specifications in the upper portion of the screen define the simulation software release number and the resource management policy versions implemented. As the Tagged Token Dataflow Architecture evolves, these version numbers will serve to distinguish between different implementations of the machine. The second screen shown below allows you to specify those buffers for which you desire profile information to be collected.

The standard data collection specification does not collect any profile information on the buffers in the TTDA. However, there are certainly meaningful experiments that would find this information essential. There is a limitation on the amount of experimental data one can collect before exceeding the capacity of the disk on which this data is stored. Although this capacity is quite large (currently 250 cylinders), data from the experiments of all users reside on this simulation facility data storage disk. Therefore, some prudence is called for when deciding on what data to collect for each experimental run.

```
                    Tagged Token Dataflow Architecture
                            Simulation Facility
          Data Collection Specification STANDARD for Experiment NAME

   Collect statistics for buffers between CE and IN stations? ......... NO
   Collect statistics for buffers between IN and I-Struct stations? ... NO
   Collect statistics for buffers between IN and WM stations? ......... NO
   Collect statistics for buffers between IN and IF stations? ......... NO
   Collect statistics for buffers between IN and MAN stations? ........ NO
   Collect statistics for buffers between IN and PeCon stations? ...... NO
   Collect statistics for buffers between WM and IF station? .......... NO
   Collect statistics for buffers between I-Struct and Out stations? .. NO
   Collect statistics for buffers between MAN and Out stations? ....... NO
   Collect statistics for buffers between PeCon and Out stations? ..... NO
   Collect statistics for buffers between IF and ALU stations? ........ NO
   Collect statistics for buffers between IF and BTAG stations? ....... NO
   Collect statistics for buffers between ALU and FT stations? ........ NO
   Collect statistics for buffers between BTAG and FT stations? ....... NO
   Collect statistics for buffers between FT and Out stations? ........ NO
   Collect statistics for buffers between Out and IN stations? ........ NO
   Collect statistics for Port 0 buffers? ............................. NO
   Collect statistics for Port 1 buffers? ............................. NO

   PF1 = Help              PF7 = Previous Screen          ENTER = Affirm
   TAB = Next Field        PF8 = Next Screen              PF3   = Exit
```

Figure 4-11: Standard Data Collection Specification

4.1.1.6. Examine Predefined Specifications

To examine the set of predefined specifications available for a particular field in the Add A Run to an Experiment function, press the **?** key followed by an **ENTER** in the desired field. After you have done this, a list of predefined specifications for the particular field will be displayed. Suppose that we examine predefined timing specifications; the following screen is an example of what might be displayed.

```
                    Tagged Token Dataflow Architecture
                           Simulation Facility
                  Listing of All Timing Specification Names

COMMAND ===>

 Browse an experiment by typing a B in front of its name.
 Select an experiment by typing a S in front of its name.
 PF1 for help.
 PF3 to exit.
 PF7/PF8 to scroll.

         Timing Specification Names

         STANDARD
         FASTWM
         HASHWM
         SLOWALU
******************************BOTTOM OF DATA******************************
```

Figure 4-12: Examining Predefined Timing Specifications

You can either select or browse specifications from this function. To select a particular specification, use the **TAB** key to move the cursor in front its name. Then type an S followed by the **ENTER** key to invoke the selection. When you return to the Add a Run to an Experiment screen, the selected specification will appear in the appropriate field.

To browse a particular specification for more information about its components, use the **TAB** key to move the cursor in front of its name. Then type a **B** followed by the **ENTER** key to invoke the browse function. The browse function will display the specification indicated by your cursor positioning. To invoke the help function or to exit, press the **PF1** or **PF3** keys, respectively. The **PF7** and **PF8** keys allow you to scroll through the names when the list takes more than one page.

4.1.2. Delete a Run from an Experiment

After selecting the delete a run function by typing **D** followed by the **ENTER** key at the Add, Delete, Modify Menu a list of all runs in the current experiment will be displayed. You can now choose to delete one or more runs from the experiment.

```
                         Tagged Token Dataflow Architecture
                                Simulation Facility
                         Deleting Run from Experiment NAME

COMMAND ===>

Type a d before the run that you wish to delete.
Type a 1 to see the the Code-block Specification of a run.
Type a 2 to see the the Configuration Specification of a run.
Type a 3 to see the the Buffer Size Specification of a run.
Type a 4 to see the the Timing Specification of a run.
Type a 5 to see the the Data Collection Specification of a run.

PF1 = Help     TAB = Next Run      PF7/PF8 = Scroll Screen      PF3 = Exit

  Code-block   Configuration   Buffer Size      Timing      Data Collection

  EXAMPLE      STANDARD        STANDARD         STANDARD    STANDARD
  MATRIX       ONEPE           INFINITE         STANDARD    NONE
  MYPROG       FOURPE          STANDARD         FASTWM      WMSTATION

*******************************BOTTOM OF DATA*******************************
```

Figure 4-13: Delete a Run from an Experiment

By tabbing to the appropriate experiment run and then typing **D**, a deletion can easily be made from the experiment. Note, however, that each of specification names remain in the simulation facility database and thus can be used for future specifications even though they have been deleted from all existing experiments. You may wish to examine a program run before deleting it. To do this, tab to the desired program run and then type 1, 2, 3, 4, or 5 depending on which of the code-block, configuration, buffer size, timing, or data collection specifications (respectively) of the run that you wish to see.

4.1.3. Modify a Run in an Experiment

The Modify a Run in an Experiment menu choice allows you to changes specifications for a program run without having to delete the run and then type in new specifications. Unfortunately, this function is not yet implemented.

4.2. Extending an Experiment

Extending an Experiment is very similar to Defining an Experiment except that you will be adding program runs to an already existing collection of runs rather than starting from scratch. You must first select the experiment to extend. This is achieved through the following selection screen.

```
                    Tagged Token Dataflow Architecture
                           Simulation Facility
                         Extending an Experiment

COMMAND ===>

 Browse an experiment by typing a B in front of its name.
 Select an experiment by typing a S in front of its name.
 PF1 for help.
 PF3 to exit.
 PF7/PF8 to scroll.

   Experiment      User        Creation Date   Last Update

   NAME            SABROBST     09/21/85        03/01/86
   LVRLOOPS        SABROBST     01/09/85        03/19/85
   PDES            ROBK         07/19/85        07/19/85

******************************BOTTOM OF DATA******************************
```

Figure 4-14: Extending an Experiment

To select a particular experiment for extension, use the TAB key to move the cursor in front of its name. Then type an S followed by the ENTER key to invoke the selection. From this point on, the extend an experiment function will provide the exact same functionality as defining a new experiment. You will be presented with the add, delete, modify menu as shown in Figure 4-2 of this chapter. Refer to section 4.1 of this chapter for explanations and instructions for using the the functions for adding, deleting, and modifying program runs.

To browse a particular experiment for more information about its components, use the **TAB** key to move the cursor in front of its name. Then type a B followed by the ENTER key to invoke the browse function. The browse function will be discussed in more detail in the next section. To invoke the help function or to exit, press the **PF1** or **PF3** keys, respectively. The **PF7** and **PF8** keys allow you to scroll through the names when the list takes more than one page.

4.3. Browsing an Experiment

It is often useful to re-examine the specifications and program runs that make up an experiment. The Browse function provides a facility to do exactly that. Invoking the Browse function on a particular experiment will cause a listing of all program runs and their associated specifications to appear on the screen.

```
                         Tagged Token Dataflow Architecture
                                Simulation Facility
                            Browsing Experiment NAME

COMMAND ===>

Type a 1 to see the the Code-block Specification of a run.
Type a 2 to see the the Configuration Specification of a run.
Type a 3 to see the the Buffer Size Specification of a run.
Type a 4 to see the the Timing Specification of a run.
Type a 5 to see the the Data Collection Specification of a run.

PF1 = Help     TAB = Next Run     PF7/PF8 = Scroll Screen     PF3 = Exit

  Code-block  Configuration  Buffer Size    Timing      Data Collection

  EXAMPLE     STANDARD       STANDARD       STANDARD    STANDARD
  MATRIX      ONEPE          INFINITE       STANDARD    NONE
  MYPROG      FOURPE         STANDARD       FASTWM      WMSTATION

******************************BOTTOM OF DATA******************************
```

Figure 4-15: Browsing an Experiment

By tabbing to the appropriate experiment run and then typing 1, 2, 3, 4, or 5 depending on which of the code-block, configuration, buffer size, timing, or data collection specifications (respectively) of the run that you wish to see, the

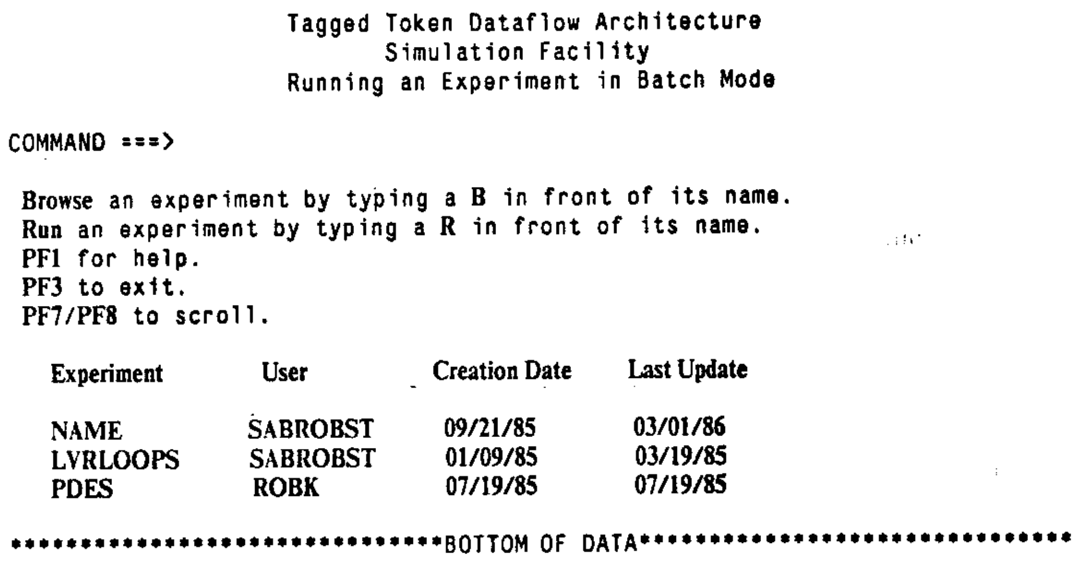
5. Running a Simulation Experiment

The ultimate goal in constructing a simulation experiment is to run it on the simulator. This section will instruct you in this task. There are two modes in which the simulator can be run. Most users will be making use of the batch-mode simulation server. There are cases, however, when someone may want to use the simulation facility in an interactive mode. These methods of running an experiment as well as monitoring the progress of a simulation experiment will be discussed in the ensuing sections.

5.1. Running in Batch Mode

Running in Batch Mode allows you to submit an experiment to the batch simulation server for processing without requiring any further user interaction. An experiment submitted to the batch simulation server will be treated as a collection of simulation jobs. Each simulation job corresponds to a simulation run within the experiment. There may be many runs from different experiments and users in the simulation server's job queue at one time. These jobs will be serviced on a first-come first-serve basis. Monitoring the current activities of the simulation server is explained in section 5.3.

After selecting the Batch Run menu option, you will be presented with the following screen. From this screen you can browse through the contents of an experiment as well as select an experiment for submission to the batch simulation server (run an experiment).



```
                    Tagged Token Dataflow Architecture
                           Simulation Facility
                    Running an Experiment in Batch Mode

COMMAND ===>

 Browse an experiment by typing a B in front of its name.
 Run an experiment by typing a R in front of its name.
 PF1 for help.
 PF3 to exit.
 PF7/PF8 to scroll.

   Experiment      User         Creation Date    Last Update

   NAME            SABROBST     09/21/85         03/01/86
   LVRLOOPS        SABROBST     01/09/85         03/19/85
   PDES            ROBK         07/19/85         07/19/85

*******************************BOTTOM OF DATA*******************************
```

Figure 5-1: Running in Batch Mode

To run a particular experiment in batch mode, use the TAB key to move the cursor in front of its name. Then type an R followed by the ENTER key to submit the batch run to the simulation server. Each simulation run in the experiment will be sent to the batch simulation server as a simulation job. A summary of the results of the simulation job will be printed under the user name SIMBATCH.

To browse a particular experiment for more information about its components, use the TAB key to move the cursor in front of its name. Then type a B followed by the ENTER key to invoke the browse function. The browse function is discussed in detail in section 4.3. To invoke the help function or to exit, press the PF1 or PF3 keys, respectively. The PF7 and PF8 keys allow you to scroll through the names when the list takes more than one page.

5.2. Running in Interactive Mode

There are only two reasons that one would want to run in interactive mode. The first is to facilitate experimentation with resource management policies that requires very fine grained control of resource allocation. Interactive mode allows one to use an interactive resource management policy whereby each resource request will cause the simulation user to be prompted regarding how that request should be satisfied.

The second reason for wanting to run in interactive mode is to allow debugging of the simulator. There are a number of tools for setting breakpoints, examining simulation data structures, etc. that are meaningful only when the simulation facility is running in interactive mode.

A major disadvantage of running in interactive mode is that data collection will necessitate usage of your disk space rather than the global data storage disk. This severely constrains the amount of data collection possible using interactive mode. Furthermore, your virtual machine will be tied up for the duration of the simulation experiment run time. Note that the batch simulation server has highest priority for machine resources on the IBM 4341. Therefore, one is likely to receive a quicker turnaround time on simulation experiments through the batch server than using the interactive facility unless the simulation job queue is very long.

Browsing through experiments and selecting them for interactive runs on the simulation facility is identical to the process described above in section 5.3. Please refer to that section for instructions on how to run an experiment.

5.3. Monitoring the Progress of a Simulation Experiment

The TTDA Simulation Facility provides a means for you to monitor the progress of a simulation experiment while it is running in batch mode. This function is invoked by typing an **M** followed by the ENTER key at the top-level menu in the simulation facility. You will then be presented with the following status screen.

```
                    Tagged Token Dataflow Architecture
                          Simulation Monitor

COMMAND ===>

  Status of the simulator is RUNNING

  Code-Block is FACTORIAL                 arguments are 10

  Started at 12:18:27 on March 1, 85

  Current time is 12:20

  Simulated time is 410850 nanoseconds

  Total number of ALU operations is now up to 2000

  Total number of station events is now up to 12339

  PF1 = Help        ENTER to see most recent status        PF3 = Exit
```

The monitor provides insight into the current activities of the batch simulation server. The simulator can be in one of three states: INACTIVE, LOADING, or RUNNING. The status of the simulation server is updated upon each transition between these three states and every 1000 ALU operations when the simulator is RUNNING a job. The most recent status of a simulation experiment will be displayed every time you press the ENTER key. Note that the status screen will **not** refresh itself; you **must** press the ENTER key. To exit the simulation monitor, simply press the **PF3** key. For help, press the **PF1** key.

The INACTIVE state indicates that the simulation server is currently idling and waiting for the submission of a new simulation job. The last job that the simulation server ran will be displayed along with the times that it ran. Do not be alarmed if the simulation monitor indicates an INACTIVE status even though you have submitted a job. It sometimes takes a couple of minutes for the server to wake up and initialize itself.

The LOADING state indicates that a simulation job has been received by the server and is currently being loaded into the simulation environment. Note that the code-block and arguments displayed do not reflect the parameters of the job being loaded. If the status of the simulation server changes immediately from the LOADING state to the INACTIVE state, then the numeric machine code for the numeric machine code file specified by the simulation job was not found or the code-block definition was not found within the numeric machine code file.

The RUNNING state of the simulation server indicates that the simulation program is currently executing a simulation run. The simulated time, number of ALU operations executed, and station events that have been processed are displayed near the bottom of the monitor screen.