CDF/DOC/ONLINE/CDFR/1242
Version 1.3
July 19, 1990

Estimation of Event Builder Execution Time for the 1991 Run

Robert Harris, Jim Patrick, Kurt Schurecht, Theresa Shaw

Abstract

In the 1991 run, the rate out of level 2 will be limited by the rate capability of the Event Builder. Consequentially, we need a rough estimation of the Event Builder execution time immediately; a precise projection requires more of the final hardware and software in place. During the 1989 run the Event Builder rate capability was about 7 Hz in one engine mode. During the 1991 run, we estimate the rate capability of a single event builder will be roughly 23 Hz in two engine mode. This rate includes the effect of redistributing the scanners on four fanout cable segments; the proposed redistribution is presented.

We have also estimated $f_{scan}f_{evb}$, the product of the scanner and event builder livetime fractions. Using the single event builder execution time of 43 milliseconds and an event scan time of 3 milliseconds, we estimate from queueing theory that if the level 2 trigger rate is 23 Hz, then $f_{scan}f_{evb}$ will be only about $0.94 \times 0.83 = 0.78$. To obtain CDF's stated goal of 90% total livetime fraction, it would be wise to maintain $f_{scan}f_{evb} = 0.95$, for which a one event builder system could only tolerate a level 2 trigger rate of about 10 Hz.

If we modify the CDF data acquisition system to accommodate two event builders we can have a larger level 2 trigger rate. For a two event builder system we estimate an event builder rate capability of 43 Hz. This system has significantly reduced deadtime. At a level 2 trigger rate of 23 Hz the livetime has increased to $f_{scan}f_{evb} = 0.94 \times 0.96 = 0.90$. However, to obtain $f_{scan}f_{evb} = 0.95$, even this system can only tolerate a level 2 output rate of only 14 Hz. For a two event builder system, the dead time at these level 2 trigger rates is dominated by the 3 millisecond scan time.

Contents

List of Tables

List of Figures

1 Introduction

The CDF hardware Event Builder[1, 2] **pulls** data from the front end scanners, **reformats** the data into YBOS bank structure, and **pushes** the data to level 3. The **pull** and **push** are initiated by FASTBUS messages sent from the Buffer Manager[1, 3] to the Event Builder's crate controller. The single crate controller board communicates with multiple reformatter boards and cable controller boards via a front panel message bus. In 1989 there were two reformatter boards, corresponding to two cable controller boards controlling two cable segments (FANOUT_CABLE_1 and FANOUT_CABLE_2). In 1991 a single event builder will contain one crate controller, four reformatters, and four cable controllers connected to four fanout cable segments.

The hardware Event Builder execution time can be divided into the time required to perform the three main Event Builder functions: The **pull time**, the **reformat time**, and the **push time**. These times can be measured on a digital oscilloscope when the event builder is running in *one engine mode*: using only one of the two buffers per reformatter. We measured these times in May 1988 with the Buffer Manager running on a μVAX II, and we repeated the measurement in June 1990 with the Buffer Manager running on a VAX 3200. The following estimation comes from those measurements, personal recollections, and some educated guesses. For those who don't have time to read this estimate, the essential results are displayed in figure 2.

2 Pull Time

The time to pull events from the scanners (MXs and SSPs) was an execution time bottleneck during the 1989 run, but will be significantly reduced for the 1991 run. The pull time can be subdivided into a **prepare to pull** time and an actual **data pull** time.

2.1 EVB Overhead: Prepare to Pull

The **prepare to pull** sequence is illustrated in figure 1a. The Event Builder crate controller receives a FASTBUS message (PULL_EVENT) from the Buffer Manager and sends a front panel message (PULL_EVENT) to each cable controller. Each cable controller then sends a message (PREP_PULL) to its reformatter, which returns a message when it is prepared for new data (PREP_COMPLETE); allowing the cable controller to begin the **pull**. The complete sequence, from receipt of the FASTBUS message (PULL_EVENT) to beginning the **pull**, presently takes about 3.5 ms. With additional reformatter boards, this time will increase to roughly 5 ms in 1991.

2.2 Data Pull Time

The **data pull** sequence is illustrated in figure 1b. The data pull time can be subdivided into two stages: **scanner communication** time and **data transfer** time.

2.2.1 Scanner Communication

During the 1989 run each scanner received four FASTBUS messages, each within a separate **AS-AK** lock[5]. These messages required approximately 0.55 ms per scanner. In 1989 FANOUT_CABLE_1 had 63 scanners (see table 1), which all went to a single reformatter, and gave a total **scanner communication** time of $63 \times 0.55 \approx 35$ ms. This time dominated the **pull** time.

Two improvements to the scanner communication time are being made for the 1991 run. First, the messages can be sent in a reduced number of **AS-AK** locks, which should be able to reduce the scanner communication time to a few hundred μs. These improvements, already begun, have reduced the MX communication time to 0.33 ms and the SSP communication time to 0.38 ms. Second, the number of cable segments that the scanners are distributed over, will be doubled from two to four. With a more sensible distribution of the 83 scanners anticipated in 1991, we should be able to reduce the maximum number of scanners on the slowest cable segment to 30 MXs (see appendix A). With these two improvements we should be able to reduce the scanner communication time on the slowest segment to roughly $30 \times 0.33 \approx 10$ ms.

2.2.2 Data Transfer

During the 1989 run the data was transferred in *Block Transfer Mode*[5]: a handshake was required for the transmission of a single data word. First the Event Builder asserted Data Sync (DS) and Read (RD), then it waited for the SSP to assert Data Acknowledge (DK) and place the data word on the Address/Data line, before proceeding with another DS and RD. The Block Transfer Mode required approximately 0.9 μs per word, and there was roughly 2.2×10^4 words on FANOUT_CABLE_1 (see table 3), resulting in a data transfer time of about 20 ms during the 1989 run.

The data transfer time per word will be speeded up in the 1991 run by transferring the data in *Pipeline Mode*[5], in which no handshake is required. The Event Builder will assert DS and RD and then almost immediately assert another DS and RD, without waiting for a DK response. Pipeline mode has already been implemented, and has reduced the data transfer time to about 0.37 μs per word. With the introduction of two new cable segments, and a sensible redistribution of scanners (see appendix A) we should be able to reduce the number of words being transferred on the slowest cable segment to about 7000 words, corresponding to a data transfer time of about 2.5 ms in 1991. Other cable segments would carry more words, but they would have much fewer scanners, so their **pull** plus **reformat** time would not be the dominant one.

2.3 Pull Completion

The pull completion sequence is illustrated in figure 1c. After scanner data on a single cable segment has been pulled into a reformatter the cable controller sends a front panel message to the reformatter (START_PROC) and the crate controller (PULL_EVENT_ACK). The crate controller updates the scoreboard and waits for all cable controllers to return front panel messages (PULL_EVENT_ACK) before it sends a FASTBUS message (PULL_COMPLETE)

to the Buffer Manager. Since reformatting begins immediately on receipt of the data, the pull completion sequence time of around 2.5 ms does not affect the total pull time.

2.4 Total Pull Time Summary

Summing the **prepare to pull** time, the **scanner communication** time and the **data transfer** time gives a total pull time of about $3.5 + 35 + 20 \approx 59$ ms in 1989. With the addition of two new cable segments, reduced scanner communication time, and reduced data transfer time, the total **pull** time should only be about $5 + 10 + 2.5 \approx 18$ ms in 1991. This is the **pull** time on the cable segment with the largest total **pull** plus **reformat** time (see appendix A).

3 Reformat Time

The reformatting sequence is illustrated in figure 1d. The reformatter does not actually rearrange the data, it merely builds YBOS bank headers and a table of block pointers to the event data. During the push stage the table of block pointers is used to push the data in the order prescribed for the given detector component YBOS bank.

Reformatting begins on a segment of the data as soon as a cable controller finishes pulling data from all its scanners and sends a front panel message to its reformatter (START_PROC). After all reformatters have finished reformatting the event, and have each sent a front panel message to the crate controller (PROC_COMPLETE), the crate controller sends a FASTBUS message (PROCESSING_COMPLETE) to the Buffer Manager indicating that reformatting is done.

The reformatting time depends mainly on the number of scanners read and the number of bank headers constructed. The reformatting code[4] calls REF_BUILD_DBANKS, which loops over scanners constructing block pointers, and then calls REF_BUILD_HEADERS, which loops over banks constructing bank headers. As discussed in appendix A, it appears that the reformatting time is linearly proportional to the number of banks and also linearly proportional to the number of scanners. This isn't unreasonable, because there is a strong correlation between the number of scanners and the number of block pointers constructed. During the 1989 run, with 63 scanners and 24 banks on the busiest cable segment, the reformat time was roughly 30 ms with the most optimized code. In 1991, with the scanners redistributed across four cable segments, the slowest cable segment could have only 30 scanners and 12 banks, which will reduce the reformatting time to roughly 14 ms with the most optimized code (see appendix A).

4 Push Time

The push sequence can be subdivided into four stages: **initial push messages** between the Event Builder and the Buffer Manager, the **prepare to push** in the Event Builder, the actual **push to level 3**, and **final push messages** between the Event Builder, the buffer manager and level 3.

4.1 Initial Push Messages

After the Event Builder finishes reformatting, it sends a message (PROCESSING_COMPLETE) to the Buffer Manager, which checks that level 3 has a free node to accept the data, and then returns a message (PUSH_EVENT) to the Event Builder. This sequence took about 8 ms with the Buffer Manager running on a μVAX II, during the first half of the 1989 run, and takes about 6 ms with the Buffer Manager running on a VAX 3200, during the second half of the 1989 run and as planned for the 1991 run.

4.2 Prepare to Push

The prepare to push sequence is illustrated in figure 1e. After receipt of the PUSH_EVENT message, the crate controller sends a front panel message (PREP_PUSH) to each reformatter. The reformatters then copy the table of block pointers to an internal Direct Memory Access (DMA) table, where it is used to push the data to level 3 in the correct YBOS order. In 1991 the pointers will be copied to a DMA table as they are calculated, virtually eliminating the prepare to push time. The prepare to push time was about 15 ms during the 1989 run, and should be reduced to less than 5 ms in 1991.

4.3 Push to Level 3

The push sequence is illustrated in figure 1f. During the 1989 run the push took about 10 ms, corresponding roughly to the Branch Bus bandwidth of 20 MBytes/sec (200 ns/word) and about 42,000 words per event (see table 3). Currently the path from the Event Builder to Level 3 involves a Branch Bus and a VME to Silicon Graphics interface called IO2. This combination has a bandwidth of only 7 MBytes/sec, but the IO2 will be replaced with a device called IO3, which should allow use of the full Branch Bus bandwidth of 20 MBytes/sec. This should allow the same bandwidth to level 3 as in the 1989 run. Including multiple interactions (see appendix B and table 5), and extra words from new detector components (see table 6), we expect about 15,000 extra words pushed to level 3 in 1991. This is roughly 35% more words than in 1989. Assuming we can achieve the same bandwidth to level 3 that we did in 1989, the push should only take about 35% more time in 1991 than in 1989, so we estimate about 13 ms for the push to level 3.

4.4 Final Push Messages

After pushing the data to Level 3, the Event Builder sends a message (PUSH_COMPLETE) to the Buffer Manager, which in turn sends a message (START_PROCESSING) to Level 3. After sending the START_PROCESSING message, the Buffer Manager completes the cycle by awakening its internal Event Manager[3] assigned for the new event. The Buffer Manager then sends a message (PULL_EVENT) to the Event Builder for the new event. This sequence took about 18 ms on a μVAX II, during the first half of the 1989 run, and takes about 10 ms with the Buffer Manager running on a VAX 3200, during the second half of the 1989 run and as planned for the 1991 run.

4.5 Total Push Time Summary

Summing the four components of the push time gives a total push time of about $8+15+10+18 = 51$ ms during the first half of the 1989 run. In 1991, initial and final message times will be about what they were during the second half of the 1989 run, with the Buffer Manager running on a VAX 3200. The prepare to push time should be greatly reduced, however the amount of data pushed will probably increase. We estimate the total push time, including message overhead, will be roughly $6 + 5 + 13 + 10 = 34$ ms in 1991.

5 One and Two Engine Mode

During the 1989 run the Event Builder used only one reformatting engine for each reformatter. During the 1991 run it will be necessary to use both reformatting engines. A single event builder's two engines can simultaneously perform any two of the three functions: **pull**, **reformat** and **push**. For example, while the data from one event on a cable segment is being pushed from one engine's buffer to level 3, the data from the next event on the same cable segment can be pulled into the second engine's buffer and reformatted. This should greatly speed up the Event Builder rates, however, a factor of 2 cannot be expected because at times one engine will have to wait for the other engine to complete. Also, there is only a single crate controller, so a two engine event builder does not correspond to two independent servers in queueing theory[6]. We have measured for a single cable segment (with 5 SSPs and 6 MXs) a rate in two engine mode which is $\frac{5}{3}$ times the rate in one engine mode. The ratio of the two engine rate to the one engine rate depends sensitively on how long each of the individual stages takes, and is likely to be smaller than $\frac{5}{3}$ in the full system. We assume, somewhat arbitrarily, that the two engine rate will be $\frac{3}{2}$ times the one engine rate.

6 Summary

The Event Builder execution time in one engine mode during the 1989 run is summarized in figure 2a. Clearly the total pull time was the longest single time, however the other times were not negligible. The total Event Builder execution time was roughly 140 ms, corresponding to a rate capability of 7 Hz.

As described in the text, the following improvements to the data acquisition system will increase the Event Builder rate capability in 1991:

- Two additional reformatters, cable controllers, and fanout cable segments, coupled with an optimized redistribution of scanners on the cable segments, will reduce the **pull** time and **reformat** time.
- More efficient scanner communication, and data transfer in pipeline mode, will reduce the **pull** time.
- Directly loading the block pointers as they are calculated will virtually eliminate the **prepare to push** time.

The execution time in one engine mode estimated for the 1991 run, summarized in figure 2b, is roughly 65 ms corresponding to a rate capability of 15 Hz. Multiplying by a factor of $\frac{3}{2}$ we find that the estimated Event Builder rate capability, in two engine mode during the 1991 run, is roughly 23 Hz. The livetime fraction while running at this rate, and at other level 2 trigger rates, is discussed in appendix C. For a combined scanner and event builder livetime fraction of $f_{scan}f_{evb} = 0.90$, a single event builder can only tolerate a level 2 trigger rate of 15 Hz.

If this rate is unacceptably low, we can consider running with two event builders. An estimate of the rate capability using two event builders is discussed in appendix D and the livetime is discussed in appendix C. For $f_{scan}f_{evb} = 0.90$, a two event builder system can only tolerate a level 2 output rate of 23 Hz. To obtain CDF's stated goal of 90% total DAQ system livetime fraction, we had better strive for $f_{scan}f_{evb} = 0.95$, for which a one event builder system can only tolerate a level 2 trigger rate of about 10 Hz, and a two event builder system can only tolerate a level 2 trigger rate of about 14 Hz.

Appendix

A Proposed Redistribution of Scanners

During the 1989 run the scanners were distributed on FANOUT_CABLE_1 and FANOUT_CABLE_2 as indicated in table 1 and table 2 respectively. Table 3 gives the total number of datawords pulled into the event builder from each cable segment, and table 4 shows the sources of all words within the data acquisition system. The data in all four tables comes from early in run 20007, when the luminosity was roughly $1.5 \times 10^{30} s^{-1} cm^{-2}$. This run used a standard trigger table (28$6).

Compared to FANOUT_CABLE_2, FANOUT_CABLE_1 carried 6 times the number of scanners, 30% more banks, and 20% more data words. This resulted in FANOUT_CABLE_1 setting the pull and reformat time.

During the 1991 run we will have two additional cable segments which I will call FANOUT_CABLE_3 and FANOUT_CABLE_4. We want to redistribute the existing scanners over the four cable segments, and assign the new scanners listed in table 6, in a way that minimizes the event builder execution time. The push time does not play a part in this minimization, since the event builder waits for each reformatter to complete before pushing to level 3. We want to minimize the sum of the pull time and reformat time, and we accomplish this by minimizing the sum for the slowest cable segment.

As discussed in section 2, the time to pull data from the scanners into the event builder on a given cable segment during the 1991 run will be approximately:

$$T_{pull}(ms) = 5 + (0.33 \times N_{MX}) + (0.38 \times N_{SSP}) + (0.37 \times N_{KWords}) \quad (1)$$

where N_{MX} and N_{SSP} and N_{KWords} are the number of MXs and SSPs and Kilowords (1000 words) on the cable segment respectively.

As discussed in section 3, the time to reformat data on a cable segment should depend mainly on the number of scanners and the number of banks. Measurements of the reformatting time in June 1990, using non-optimized code, indicate the reformatting time on a given cable segment is roughly estimated by the simple relation:

$$T_{reformat} \approx (T_{bank} \times N_{bank}) + (T_{scan} \times N_{scanner}) \quad (2)$$

where T_{bank} and $T_{scanner}$ are the reformatting time per bank and per scanner respectively, and N_{bank} and $N_{scanner}$ are the number of banks and scanners on the cable segment respectively. Measurements indicate that the reformat time for MXs and SSPs are roughly the same and that

$$T_{bank} \approx 1.5 \times T_{scanner} \quad (3)$$

The reformat time on FANOUT_CABLE_1, using the most optimized code during the 1989 run, was roughly 30 ms for 63 scanners and 24 banks. Using this information in equations (2) and (3) we estimate the reformatting time per scanner will be $T_{scanner} \approx .3ms$. Using this time and equations (1), (2) and (3), we can estimate the pull and reformat time for

any distribution of scanners. However, we are not free to make any possible distribution, because the event builder requires that all scanners that contain data for a single bank (such as CEMD) must be on the same cable segment. This immediately determines the indivisible **groups** of scanners shown in table 1 and table 2. Note that in order to redistribute some of the MXs on FANOUT_CABLE_1 to another cable, we have had to violate this rule for the single bank TMXD (timing information for all MXs), which will have to be split into two banks: MX1D and MX2D. There is one other consideration guiding our redistribution: it would be convenient, though not absolutely necessary, if we could keep the central, wall, and plug MXs on the same cable segment in order for us not to have to introduce any new FASTBUS crates for MEPs. With that consideration in mind we have constructed strawman proposal 1, shown in table 7, which has a **pull** plus **reformat** time of roughly 39 ms. If we are allowed to split the plug and central MX's over two cable segments we can cut this down to roughly 32 ms, as shown in table 8, which is strawman proposal 2. Notice also that if we can split the plug and central MXs we can roughly equalize the sum of the pull and reformat time over all four segments. I have used strawman proposal 2 for all the 1991 timing estimates in this paper.

B Multiple Interactions

The number of data words in the DAQ pipeline will depend on the average number of extra minimum bias interactions per hard collision. Using Poisson statistics and a minimum bias cross section of 44 mb, Chris Wendt has calculated that at a luminosity of $10^{31} s^{-1} cm^{-2}$, roughly 22% of our standard triggers will be single interactions, 33% will be double, 25% will be triple, 12% will be quadruple, 5% will have quintuple interactions, and 2% will have more than 5 interactions. Thus on average there will be roughly $0.33 + 2(.25) + 3(.12) + 4(.05) + 5(.02) = 1.5$ extra minimum bias interactions in a standard event.

To estimate how many extra data words are produced by an extra minimum bias interaction, we subtract the number of data words per event in a level 0 query run from the number of data words per event in a minimum bias run. This subtracts off the detector noise data words which would already be present in the event, and gives a rough estimate of the extra words per extra interaction. Using run 20445, which was acquired with a level 0 query trigger, we select those events which pass the BBC_INTIME_YMON level 1 trigger, and calculate the mean number of datawords per event averaged over the run. We do the same for run 20445 without any additional trigger requirement, and subtract the two. The distributions are not gaussian, and the RMS deviations are large. The results shown in table 5, need to be multiplied by 1.5 to get an estimate of the additional number of words expected in 1991. We need to subtract VTPD data, since the VTX has no pads. During the 1991 run we can expect an extra 15,000 words being pulled by the event builder and pushed to level 3, and an extra 10,000 words being pulled by the VAX and written to tape.

C Livetime

The livetime fraction f of the complete CDF data acquisition system, including the trigger, can be factorized into individual livetime fractions:

$$f = f_{L2}(R_{L1}, T_{L2}) f_{scan}(R_{L2}, T_{scan}) f_{evb}(R_{buf}, T_{evb}) f_{L3}(R_{evb}, T_{L3}) f_{tape}(R_{L3}, T_{tape}) \quad (4)$$

In this note we shall only consider two of these. First, the livetime fraction of the scanners is

$$f_{scan} = \frac{1}{1 + R_{L2} T_{scan}} \quad (5)$$

where R_{L2} is the rate of level 2 triggers and T_{scan} is the time required to scan a single event. Second, the livetime fraction of the event builder depends on whether we are considering a one or two event builder system. For both cases we define the dimensionless variable $x = R_{buf} T_{evb}$, where $R_{buf} = R_{L2} f_{scan}$ is the rate of events flowing into the scanner buffers, and T_{evb} is the time required for a single event builder to **pull**, **reformat** and **push** a single event in two-engine mode. From queueing theory[6] we can derive the livetime fraction of a system with N event buffers and one server:

$$f_1 = \frac{1 - x^N}{1 - x^{N+1}} \quad (6)$$

In our case there are four buffers per scanner so $N = 4$.

With $T_{evb} = 43$ ms, and $T_{scan} = 3$ ms, we obtain the first four columns of table 9, which gives the scanner and event builder livetime fractions as a function of Level 2 trigger rate. In this approximation we have considered a single event builder running in two engine mode as a single server, which is a reasonable assumption since both engines cannot perform identical operations at the same time, they can only perform non-identical operations at the same time and there is considerable waiting time. The level 3 input rate is given from the level 2 output rate times $f_{scan} f_{evb}$. We do not achieve level 3 input rate equal to the event builder capability until the level 2 trigger rate is infinite, and the livetime fraction is 0%. To obtain $f_{scan} f_{evb} = 0.9$ we must run the level 2 trigger at $R_{L2} \approx 15$ Hz. This will not achieve CDF's stated goal of 90% total livetime unless every other stage in the DAQ pipeline has a livetime of 100%, which is unlikely indeed. If we wish to obtain the goal of 90% total livetime, it would be wise to require $f_{scan} f_{evb} = 0.95$, which allows $R_{L2} \approx 10$ Hz.

With two event builders acting as independent servers of four buffers the livetime fraction is given from queueing theory [6]:

$$f_2 = \frac{1 + x + \frac{x^2}{2} + \frac{x^3}{4}}{1 + x + \frac{x^2}{2} + \frac{x^3}{4} + \frac{x^4}{8}} \quad (7)$$

Using equation (7) with $T_{evb} \approx 46ms$, as discussed in appendix D, gives the last two columns of table 9. For a system with two event builders, to obtain a combined scanner and event builder livetime of $f_{scan} f_{evb} = 0.9$, the level 2 trigger rate should be $R_{L2} \approx 23$ Hz. To realistically attempt a total livetime fraction of 90%, we should try and obtain $f_{scan} f_{evb} = 0.95$, for which $R_{L2} \approx 14$ Hz.

D Two Event Builders

The system throughput can be increased if we consider using two event builders. Since each event builder would be connected separately to half of the level 3 three processors, the **reformat** and **push** could proceed in parallel. An extra **pull** time would be incurred during the fraction of the time when the two event builders collided, which is approximately the single event builder output rate (15 Hz) times the event builder **data pull** time on the cable segment (13 ms), which gives a cable segment *duty cycle* of $15 \times .013 \approx 0.2$. Then the time to process two events, using two event builders, would be the single event builder execution time (43 ms) plus the product of the cable segment duty cycle (0.2) and the **data pull** time (13 ms), which is just $43 + 0.2 \times 13 \approx 46$ ms. This is the effective server time, T_{evb}, for a single event builder in a two event builder system. Thus we estimate an execution time of 23 ms per event, or a rate capability of 43 Hz, for a system with two event builders. Estimates of the livetime are discussed in appendix C and tabulated in table 9.

It should be noted that a two event builder system does not come for free. To implement two event builders would require non-trivial changes to the Buffer Manager. Also, managing a total of 18 error and control windows, for 18 event builder boards, might require improving the existing control and error structure. We have yet to make and debug even a single event builder in the nine board configuration.

References

[1] E. Barsotti et al., "FASTBUS Data Acquisition for CDF", Nucl. Instr. and Meth. **A269**(1988)82.

[2] A. W. Booth et al., "The CDF Event Builder", IEEE Trans. on Nucl. Sci. **NS-34**(1987)790.

[3] C. Day, "Buffer Manager Software Design", CDF Note 326, July 1985.

[4] A. W. Booth and P. K. Sinervo, "Software Specification for the Hardware Event Builder", Event Builder Note S1, September 1987.

[5] IEEE standard FASTBUS, ANSI/IEEE Std 960-1986.

[6] Lee Holloway, "An Application of Queueing Theory to the CDF Data Acquisition Deadtime Problem", CDF Note 536, August 1987.

Group	Bank	Crate	Scanner	Blocks	Bank Words	Data Words	RMS
CTC	CTCD	3A-3F	SSP 00-05	10	6906	6889	2566
	CFHD	3A	SSP 00	10	385	368	0
	CFWD	3A	SSP 00	1	16	8	6
CEN	CEMD	11-12	MX 00-23	24	185	167	61
	CEGD	11-12	MX 00-23	24	426	408	65
	CESD	11-12	MX 00-23	24	1188	1170	382
	CHAD	11-12	MX 00-23	24	108	90	42
	CHTD	11-12	MX 00-23	24	30	12	7
	CCRD	11-12	MX 00-23	16	363	349	44
	CMUD	11	MX 00-23	24	136	118	271
WALL	WHAD	12	MX 24-25	24	103	85	39
	WHTD	12	MX 24-25	24	24	6	5
PLUG	PEMD	11-12	MX 26-37	24	911	893	173
	PESD	11-12	MX 26-37	24	247	229	44
	PHAD	11-12	MX 26-37	24	740	722	118
	PEAD	11-12	MX 26-37	8	209	199	40
	PHWD	11-12	MX 26-37	24	226	208	38
MX	TMXD	11-13	MX 00-59	60	204	168	0
CDT	CDTD	13	MX 38-41	6	639	630	205
FOR	FEMD	13	MX 42-53	8	900	890	319
	FEAD	13	MX 42-53	8	731	721	109
	FHXD	13	MX 42-53	8	397	387	103
	FHAD	13	MX 42-53	8	526	516	192
	FMSD	13	MX 54-55	8	178	168	180
Sum	24	5	63	439	15789	15412	3650
Extra TDC words (before L3 reformatting)						6889	2566
Pulled	24	5	63	439	22678	22302	6112

Table 1: Banks, scanners, and words on FANOUT_CABLE_1 during the 1989 run.

Group	Bank	Crate	Scanner	Blocks	Bank Words	Data Words	RMS
VPTC	VTWD	21-22	SSP 08-09	8	4846	4831	2162
	VTPD	23	SSP 13-14	8	4940	4925	1823
	FMCD	29	SSP 19-20	8	31	16	0
FMU	FMTD	29	SSP 19-20	2	521	512	0
	FMUD	29	SSP 19-20	16	520	497	247
	TAGC	07	SSP 24-27	0	964	950	0
	TL2D	07	SSP 24	13	234	214	19
	TL1D	07	SSP 25	3	82	72	0
	TCSD	07	SSP 24	64	878	807	208
	TRCD	07	SSP 25	10	369	352	0
TRIG	TCMD	07	SSP 27	3	101	91	6
	SCLD	07	SSP 26	15	566	544	0
	LATD	07	SSP 26	3	19	9	0
	BFLD	07	SSP 26	1	16	8	0
	BBCD	07	SSP 27	2	145	136	0
	BBLD	07	SSP 27	1	10	2	0
	TFRD	07	SSP 26	1	13	5	0
	TODD	07	SSP 27	1	15	7	0
Sum	18	5	10	159	14272	13980	4030
Extra TDC words (before L3 reformatting)						4831	2162
Pulled	18	5	10	159	19103	18811	6160

Table 2: Banks, scanners, and words on FANOUT_CABLE_2 during the 1989 run.

DAQ Stage	Longwords	RMS
Pulled on FANOUT_CABLE_1	22302	6112
Pulled on FANOUT_CABLE_2	18811	6160
Pushed to L3	41983	11774
Pulled to VAX	31493	7552

Table 3: The mean and standard deviation of the number of words per event during the 1989 run. The number pushed to level 3 equals the number pulled on the two cables plus block pointers and bank headers.

DAQ Source of words	Longwords	RMS
Unreformatted TDC banks from FANOUT_CABLE_1	13777	5132
Other banks from FANOUT_CABLE_1	8883	1570
Unreformatted TDC banks from FANOUT_CABLE_2	9740	4365
Other banks from FANOUT_CABLE_2	9426	1971
Level 3 Reformatted TDC banks	11753	4475
L3 Filter Module Output banks	1275	295
Total Event Record	55011	16423
Recorded on Tape (no unreformatted TDC data)	31493	7552

Table 4: Sources of words in the event record during the 1989 run. LeCroy 1879 TDC data is reformatted in level 3 (one word out for every two words in) and only the reformatted data is written to tape.

DAQ Stage	Longwords	RMS
Extra Pulled on FANOUT_CABLE_1	6000	5000
Extra Pulled on FANOUT_CABLE_2	6000	5000
Extra Pushed to L3	12000	10000
Extra Pulled to VAX	8000	6000
Banks	**Longwords**	**RMS**
Extra CTCD	2100	2000
Extra VTWD	1700	1600
Extra VTPD	1600	1600
Extra FEMD	360	250
Extra FMUD	260	220
Extra FHAD	170	120
Extra FEAD	160	110
Extra FHXD	150	90
Extra CDTD	90	130
Extra PEMD	60	70
Extra PEMD	60	70
Extra CESD	50	150
Extra CEMD	30	30

Table 5: Rough estimates of the mean and RMS deviation of the number of extra data words per extra minimum bias interaction in an event (detector noise subtracted). To obtain the extra number of words due to multiple interactions in 1991 you need to subtract the number of VTPD words and multiply the result by 1.5 interactions. For extra words pulled, CTCD and VTWD data need to be multiplied by a factor of 2 to account for unreformatted TDC data.

Group	Extra Banks	Extra Scanners	Extra Words
SVX	SVXD	4 SSPs	4800
VTX	-	2 SSPs	0
CPT	CPTD	1 SSP	400
CPR	CPRD	-	200
CMX	CMXD	2 MXs	200
WALL	-	2 MXs	-
Total	4	11	5600

Table 6: **Extra banks, scanners, and words in 1991 over that in 1989.**

Cable Segment	Groups	Banks	Scanners	Words	Pull	Ref	Sum
FANOUT_CABLE_1	CEN	CEMD,CEGD CESD,CHAD CHTD,CCRD CPRD,CMUD	24 MXs	2600			
	WALL	WHAD,WHTD	4 MXs	100			
	PLUG	PEMD,PEAD PESD,PHAD PHWD	12 MXs	2400			
	MX1	MX1D		100			
	Total	16 Banks	40 MXs	5200	20 ms	19 ms	39 ms
FANOUT_CABLE_2	VTX	VTWD	6 SSPs	15000			
	FMU	FMCD,FMTD FMUD	2 SSPs	1400			
	TRIG	TAGC,TL2D TL1D,TCSD TRCD,TCMD SCLD,LATD BFLD,BBCD BBLD,TFRD TODD	4 SSPs	3000			
	Total	17 Banks	12 SSPs	19000	17 ms	12 ms	29 ms
FANOUT_CABLE_3	CTC	CTCD,CFHD CFWD,CPTD	7 SSPs	21000			
	SVX	SVXD	4 SSPs	4800			
	CMX	CMXD	2 SSPs	200			
	Total	6 Banks	13 SSPs	26000	20 ms	7 ms	27 ms
FANOUT_CABLE_4	FOR	FEMD,FEAD FHAD,FHXD FMSD	14 MXs	3900			
	CDT	CDTD	4 MXs	700			
	MX2	MX2D		50			
	Total	7 Banks	18 SSPs	4700	14 ms	9 ms	23 ms

Table 7: **Strawman proposal 1 for redistribution of scanners for the 1991 run. FANOUT_CABLE_1 would make the sum of the pull and reformat time 39 ms. The estimated number of words pulled by the event builder includes multiple interactions and new components.**

Cable Segment	Groups	Banks	Scanners	Words	Pull	Ref	Sum
FANOUT_CABLE_1	CEN	CEMD,CEGD CESD,CHAD CHTD,CCRD CPRD,CMUD	24 MXs	2600			
	WALL	WHAD,WHTD	4 MXs	100			
	MX1	MX1D		100			
	Total	11 Banks	28 MXs	2800	15 ms	13 ms	28 ms
FANOUT_CABLE_2	VTX	VTWD	6 SSPs	15000			
	FMU	FMCD,FMTD FMUD	2 SSPs	1400			
	TRIG	TAGC,TL2D TL1D,TCSD TRCD,TCMD SCLD,LATD BFLD,BBCD BBLD,TFRD TODD	4 SSPs	3000			
	Total	17 Banks	12 SSPs	19000	17 ms	12 ms	29 ms
FANOUT_CABLE_3	CTC	CTCD,CFHD CFWD,CPTD	7 SSPs	21000			
	SVX	SVXD	4 SSPs	4800			
	CMX	CMXD	2 SSPs	200			
	Total	6 Banks	13 SSPs	26000	20 ms	7 ms	27 ms
FANOUT_CABLE_4	FOR	FEMD,FEAD FHAD,FHXD FMSD	14 MXs	3900			
	PLUG	PESD,PHAD PEMD,PEAD PHWD	12 MXs	2400			
	CDT	CDTD	4 MXs	700			
	MX2	MX2D		100			
	Total	12 Banks	30 MXs	7100	18 ms	14 ms	32 ms

Table 8: **Strawman proposal 2 for redistribution of scanners for the 1991 run. FANOUT_CABLE_4 would make the sum of the pull and reformat time 31 ms. The estimated number of words pulled by the event builder includes multiple interactions and new components.**

Level 2 Output (Hz)	Scanner Livetime Fraction	One EVB System		Two EVB System	
		Livetime Fraction	Level 3 Input (Hz)	Livetime Fraction	Level 3 Input (Hz)
10	0.971	0.981	9.5	0.997	9.7
11	0.968	0.975	10.4	0.996	10.6
12	0.965	0.967	11.2	0.994	11.5
13	0.962	0.958	12.0	0.992	12.4
14	0.960	0.948	12.7	0.990	13.3
15	0.957	0.937	13.4	0.988	14.2
16	0.954	0.925	14.1	0.985	15.0
17	0.951	0.912	14.8	0.982	15.9
18	0.949	0.899	15.4	0.979	16.7
19	0.946	0.885	15.9	0.975	17.5
20	0.943	0.871	16.4	0.971	18.3
21	0.941	0.856	16.9	0.967	19.1
22	0.938	0.841	17.4	0.963	19.9
23	0.935	0.826	17.8	0.958	20.6
24	0.933	0.811	18.1	0.953	21.3
25	0.930	0.796	18.5	0.947	22.0
26	0.928	0.781	18.8	0.942	22.7
27	0.925	0.766	19.1	0.936	23.4
28	0.923	0.751	19.4	0.930	24.0
29	0.920	0.737	19.7	0.924	24.6
30	0.917	0.723	19.9	0.918	25.3
31	0.915	0.709	20.1	0.911	25.8
32	0.912	0.695	20.3	0.904	26.4
33	0.910	0.682	20.5	0.898	27.0
34	0.907	0.669	20.7	0.891	27.5
35	0.905	0.657	20.8	0.884	28.0
36	0.903	0.645	20.9	0.877	28.5
37	0.900	0.633	21.1	0.870	29.0
38	0.898	0.622	21.2	0.863	29.4
39	0.895	0.610	21.3	0.856	29.9
40	0.893	0.600	21.4	0.849	30.3

Table 9: **The estimated livetime fraction of the scanners, and a one (or two) event builder system in 1991, is shown as a function of the level 2 trigger rate. The estimate assumes a scan time of 3 ms and a single event builder execution time of 43 ms. For a two event builder system, the effective single event builder execution time used was 46 ms. Also shown is the level 3 input rate, equal to the level 2 output rate times the product of the scanner and event builder livetime fractions.**

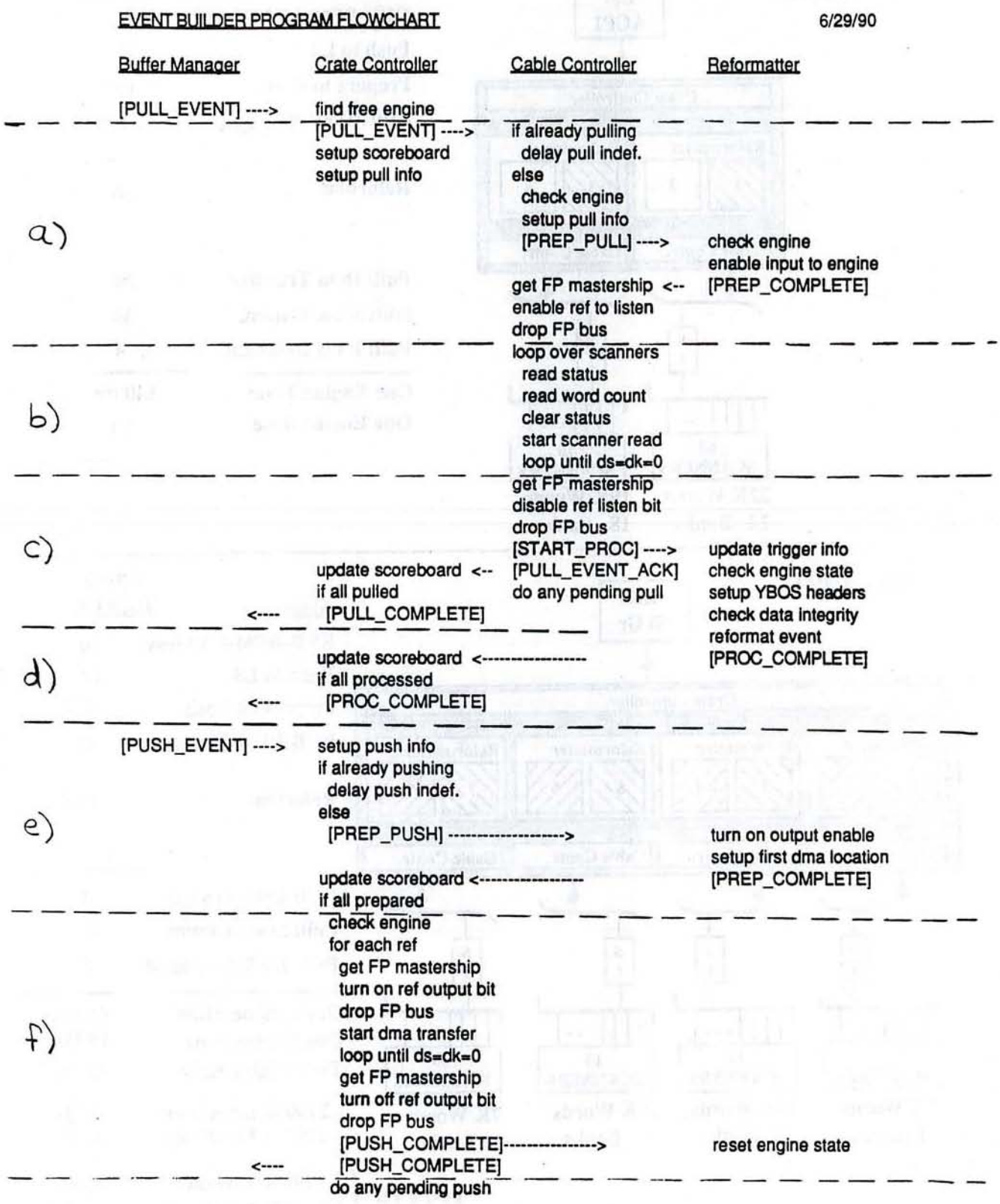
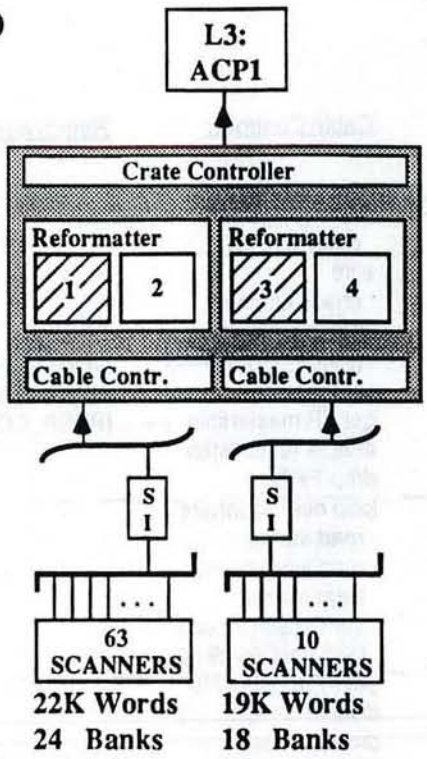


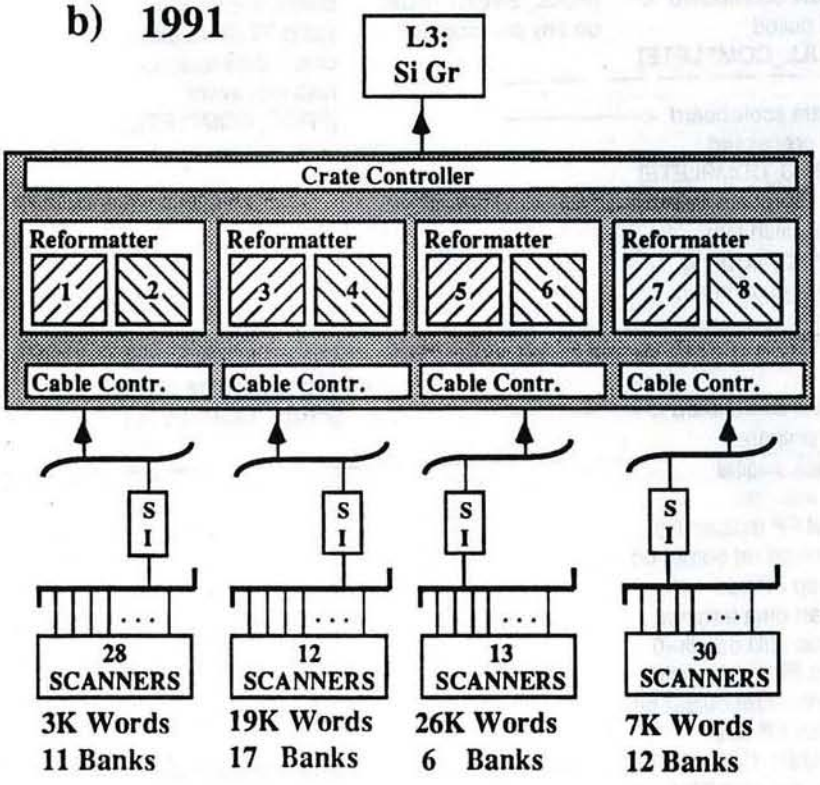
Figure 1: Event Builder execution stages are illustrated by this flow chart of Event Builder code. **a) Prepare to pull, b) data pull, c) pull completion, d) reformat, e) prepare to push, f) push to level 3. The Event Builder waits for messages from the Buffer Manager between d) and e), and** between **f) and a).**

a) 1989



Stage	Approx Time (ms)
EVB-BFM-L3 Messages	18
Push to L3	10
Prepare to Push	15
EVB-BFM Messages	8
Reformat	30
Pull: Data Transfer	20
Pull: Scan. Comm.	35
Pull: EVB Overhead	4
One Engine Time	140 ms
One Engine Rate	7 Hz

b) 1991



Stage	Approx. Time (ms)
EVB-BFM-L3 Mess.	10
Push to L3	13
Prepare to Push	5 ?
EVB-BFM Messages	6
Reformat	14 ?
Pull: Data Transfer	3
Pull: Scan. Comm.	10
Pull: EVB Overhead	5
One Engine Time	66 ms
One Engine Rate	15 Hz
Two Engine Rate	23 Hz ?
L2 (90% Livetime)	15 Hz
L2 (95% Livetime)	10 Hz
Two EVB: L2 (90% Livetime)	23 Hz
Two EVB: L2 (95% Livetime)	14 Hz

Figure 2: Sources of event builder execution time, in a) 1989 and b) 1991, are listed and totalled. Data flow is schematically illustrated: from the scanners, over crate and cable segments, into the event builder (shaded), and on up to level 3.